TOPS–20 Monitor Calls Reference Manual

AA–4166E–TM, AD–4166E–T1

December, 1982

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.

Information in these pages updates the manual of the same name and order no. AA–4166E–TM.

OPERATING SYSTEM: TOPS–20 V5 (KS/KL Model A)
TOPS–20 V5.1 (KL Model B)

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

Northeast/Mid–Atlantic Region	Central Region	Western Region
Digital Equipment Corporation PO Box CS2008 Nashua, New Hampshire 03061 Telephone:(603)884–6660	Digital Equipment Corporation Accessories and Supplies Center 1050 East Remington Road Schaumburg, Illinois 60195 Telephone:(312)640–5612	Digital Equipment Corporation Accessories and Supplies Center 632 Caribbean Drive Sunnyvale, California 94086 Telephone:(408)734–4915

digital equipment corporation • marlboro, massachusetts

UPDATE NOTICE

TOPS–20
Monitor Calls
Reference Manual
AD–4166E–T1

December, 1982

Insert this Update Notice in the *TOPS–20 Monitor Calls Reference Manual* to maintain an up-to-date record of changes to the manual.

Changed Information

The changed pages contained in this update package reflect changes to the monitor calls for TOPS–20, Version 5.1.

The instructions for inserting this update start on the next page.

digital software

INSTRUCTIONS
AD–4166E–T1

The following list of page numbers specifies which pages are to be placed in the *TOPS–20 Monitor Calls Reference Manual* as replacements for, or additions to, current pages.

Title page Copyright page	3–285 3–294	Index–1 Index–2
vii viii	3–415 3–416	Index–11 Index–12
xiii xiv	3–469 3–470	Index–15 Index–16

KEEP THIS UPDATE NOTICE IN YOUR MANUAL TO MAINTAIN AN UP-TO-DATE RECORD OF CHANGES.

TYPE AND IDENTIFICATION OF DOCUMENTATION CHANGES.
Five types of changes are used to update documents contained in the TOPS–20 software manuals. Change symbols and notations are used to specify where, when, and why alterations were made to each update page. The five types of update changes and the manner in which each is identified are described in the following table.

The Following Symbols and/or Notations	Identify the Following Types of Update Changes
1. Change bar in outside margin; version number and change date printed at bottom of page.	1. Changes were required by a new version of the software being described.
2. Change bar in outside margin; change date printed at bottom of page.	2. Changes were required to either clarify or correct the existing material.
3. Change date printed at bottom of page.	3. Changes were made for editorial purposes but use of the software is not affected.
4. Bullet (●) in outside margin; version number and change date printed at bottom of page.	4. Data was deleted to comply with a new version of the software being described.
5. Bullet (●) in outside margin; change date printed at bottom of page.	5. Data was deleted to either clarify or correct the existing material.

CONTENTS

			Page
PREFACE			
CHAPTER 1		INTRODUCTION	
	1.1	CALLING CONVENTIONS	1-1
	1.2	MONITOR CALL ARGUMENTS	1-2
	1.2.1	Addresses	1-2
	1.2.2	Page Numbers	1-3
	1.2.3	Section Numbers	1-3
	1.2.4	Byte Pointers	1-4
	1.2.5	File Handles and File Designators	1-5
	1.2.6	Source/Destination Designators	1-6
	1.2.6.1	File Designator	1-7
	1.2.6.2	Byte Pointers and ASCII Strings	1-7
	1.2.6.3	Special Designators	1-8
	1.2.6.4	Numeric Designators	1-8
	1.2.7	Device Designator	1-8
	1.2.8	Process Handles	1-9
	1.2.8.1	Process/File Handle	1-9
	1.3	SYSTEM DATE AND TIME	1-9
	1.4	PROCESSING ERRORS	1-10
	1.5	CONVENTIONS USED IN THIS MANUAL	1-11
	1.5.1	Number Bases	1-11
	1.5.2	Abbreviations	1-12
	1.5.3	Symbols	1-12
	1.5.4	Unimplemented Features	1-12
CHAPTER 2		FUNCTIONAL ORGANIZATION OF JSYS'S	
	2.1	ACCOUNTING FUNCTIONS	2-1
	2.2	REFERENCING FILES	2-1
	2.2.1	File Specifications	2-1
	2.2.2	Logical Names	2-2
	2.2.3	File Handles	2-3
	2.2.4	File References	2-5
	2.2.4.1	Files and Devices	2-5
	2.2.5	Sample Program	2-5
	2.2.6	File Access	2-8
	2.2.7	Directory Access	2-9
	2.2.8	File Descriptor Block	2-10
	2.2.9	Primary Input and Output Files	2-20

CONTENTS (Cont.)

		Page
2.2.10	Methods of Data Transfer	2-20
2.2.11	File Byte Count	2-20
2.2.12	EOF Limit	2-21
2.2.13	Input/Output Errors	2-21
2.2.13.1	Testing for End-of-File	2-22
2.3	OBTAINING INFORMATION	2-24
2.3.1	Error Mnemonics and Message Strings	2-24
2.3.2	System Tables	2-24
2.4	COMMUNICATING WITH DEVICES	2-32
2.4.1	Physical Card Reader (PCDR:)	2-33
2.4.2	Spooled Card Reader (CDR:)	2-34
2.4.3	Physical Card Punch (PCDP:)	2-35
2.4.4	Spooled Card Punch (CDP:)	2-35
2.4.5	Physical Line Printer (PLPT:)	2-36
2.4.5.1	PLPT: Status Bits	2-38
2.4.6	Spooled Line Printer (LPT:)	2-38
2.4.7	Physical Magnetic Tape (MTA:)	2-39
2.4.7.1	Buffered I/O	2-40
2.4.7.2	Unbuffered I/O	2-41
2.4.7.3	Magnetic Tape Status	2-41
2.4.7.4	Reading a Tape in the Reverse Direction	2-41
2.4.7.5	Hardware Data Modes	2-42
2.4.8	Logical Magnetic Tape (MT:)	2-45
2.4.9	Terminal (TTY:)	2-45
2.4.9.1	JFN Mode Word	2-45
2.4.9.2	Control Character Output Control	2-48
2.4.9.3	Character Set	2-48
2.4.9.4	Terminal Characteristics Control	2-51
2.4.9.5	Terminal Linking	2-53
2.4.9.6	Terminal Advising	2-53
2.5	SOFTWARE DATA MODES	2-54
2.6	SOFTWARE INTERRUPT SYSTEM	2-57
2.6.1	Software Interrupt Channels	2-57
2.6.2	Software Interrupt Priority Levels	2-58
2.6.3	Software Interrupt Tables	2-59
2.6.4	Terminating Conditions	2-59
2.6.5	Panic Channels	2-60
2.6.6	Terminal Interrupts	2-60
2.6.6.1	Terminal Interrupt Modes	2-62
2.6.7	Dismissing an Interrupt	2-62
2.7	PROCESS CAPABILITIES	2-63
2.7.1	Assigned Capabilities	2-64
2.7.2	Access Control	2-65
2.7.3	Processes and Scheduling	2-67
2.7.3.1	Process Freezing	2-67
2.7.3.2	Execute-Only Files and Execute-Only Processes	2-68
2.8	SAVE FILES	2-70
2.8.1	Format for Nonsharable Save Files	2-70
2.8.2	Format of Sharable Save Files	2-71
2.8.3	Entry Vector	2-74
2.8.4	Program Data Vector	2-75
2.9	INPUT/OUTPUT CONVERSION	2-75
2.9.1	Floating Output Format Control	2-76
2.9.1.1	Free Format	2-76
2.9.1.2	General Format Control	2-76

CONTENTS (Cont.)

			Page
2.9.2		Date and Time Conversion Monitor Calls	2-79
2.10		ARCHIVE/VIRTUAL DISK SYSTEM	2-81
2.11		PRIVILEGED MONITOR CALLS	2-82
CHAPTER 3		TOPS-20 MONITOR CALLS	
ACCES	(552)	Specifies access to a directory	3-2
ADBRK	(570)	Controls address breaks	3-5
AIC	(131)	Activates software interrupt channels	3-8
ALLOC	(520)	Allocates a device	3-9
ARCF	(247)	Archive/virtual disk operations	3-11
ASND	(70)	Assigns a device	3-15
ASNSQ	(752)	Assigns ARPANET special message queue	3-16
ATACH	(116)	Attaches a terminal to a job	3-17
ATI	(137)	Assigns a terminal code to a software interrupt channel	3-19
ATNVT	(274)	Creates ARPANET Network Virtual Terminal Connection	3-20
BIN	(50)	Performs byte input	3-21
BKJFN	(42)	Backs up the source designator's pointer by one byte	3-22
BOOT	(562)	Performs functions required for loading front-end software	3-23
BOUT	(51)	Performs byte output	3-37
CACCT	(4)	Changes account designator	3-38
CFIBF	(100)	Clears the input buffer	3-39
CFOBF	(101)	Clears the output buffer	3-40
CFORK	(152)	Creates an inferior process	3-41
CHFDB	(64)	Changes a File Descriptor Block	3-43
CHKAC	(521)	Checks access to a file	3-45
CIS	(141)	Clears the interrupt system	3-47
CLOSF	(22)	Closes a file	3-48
CLZFF	(34)	Closes the process' files	3-50
COMND	(544)	Parses a command	3-52
CRDIR	(240)	Creates, changes, or deletes a directory	3-75
CRJOB	(2)	Creates a job	3-81
CRLNM	(502)	Defines or deletes a logical name	3-87
CVHST	(276)	Converts ARPANET host number to primary name	3-89
CVSKT	(275)	Converts ARPANET local socket to absolute form	3-90
DEBRK	(136)	Dismisses current software interrupt	3-91
DELDF	(67)	Expunges deleted files	3-92
DELF	(26)	Deletes files	3-94
DELNF	(317)	Retains specified number of generations of a file	3-96
DEQ	(514)	Removes request from resource queue	3-97
DEVST	(121)	Translates a device designator to a string	3-99
DFIN	(234)	Inputs double-precision floating point number	3-100

CONTENTS (Cont.)

			Page
DFOUT	(235)	Outputs double-precision floating point number	3-101
DIAG	(530)	Reserves or releases hardware channels	3-102
DIBE	(212)	Dismisses until input buffer is empty	3-106
DIC	(133)	Deactivates software interrupt channels	3-107
DIR	(130)	Disables software interrupt system	3-108
DIRST	(41)	Translates a directory number to a string	3-109
DISMS	(167)	Dismisses the process	3-110
DOBE	(104)	Dismisses until output buffer is empty	3-111
DSKAS	(244)	Assigns or deassigns disk addresses	3-112
DSKOP	(242)	Specifies disk transfers in hardware terms	3-113
DTACH	(115)	Detaches a terminal from a job	3-115
DTI	(140)	Deassigns a terminal code	3-116
DUMPI	(65)	Reads data in unbuffered data mode	3-117
DUMPO	(66)	Writes data in unbuffered data mode	3-119
DVCHR	(117)	Retrieves device characteristics	3-121
EIR	(126)	Enables software interrupt system	3-123
ENQ	(513)	Places request in resource queue	3-124
ENQC	(515)	Obtains status of resource queue	3-130
EPCAP	(151)	Enables process capabilities	3-134
ERSTR	(11)	Converts error number to string	3-135
ESOUT	(313)	Outputs an error string	3-136
FFFFP	(31)	Finds first free page in file	3-137
FFORK	(154)	Freezes processes	3-138
FFUFP	(211)	Finds first used page in file	3-139
FLHST	(277)	Flushes an ARPANET host	3-140
FLIN	(232)	Inputs floating-point number	3-141
FLOUT	(233)	Outputs floating-point number	3-142
GACCT	(546)	Gets current account designator	3-143
GACTF	(37)	Gets account designator of file	3-144
GCVEC	(300)	Gets entry vector of compatibility package	3-145
GDSKC	(214)	Gets disk count	3-146
GDSTS	(145)	Gets device's status	3-147
GDVEC	(542)	Gets entry vector of RMS	3-148
GET	(200)	Gets a save file	3-149
GETAB	(10)	Gets a word from a monitor table	3-152
GETER	(12)	Returns the last error in a process	3-153
GETJI	(507)	Gets specified job information	3-154
GETNM	(177)	Returns the program name currently being used	3-156
GETOK%	(574)	Requests access to a protected resource	3-157
GEVEC	(205)	Gets entry vector	3-162
GFRKH	(164)	Gets process handle	3-163
GFRKS	(166)	Gets process structure	3-164
GFUST	(550)	Returns author and last writer name strings	3-166
GIVOK%	(576)	Grants access to a protected resource	3-167

CONTENTS (Cont.)

			Page
GJINF	(13)	Gets current job information	3-168
GNJFN	(17)	Gets the next JFN	3-169
GPJFN	(206)	Gets the primary JFNs	3-170
GTAD	(227)	Gets current date and time	3-171
GTDAL	(305)	Gets disk allocation of a directory	3-172
GTDIR	(241)	Gets information of directory entry	3-173
GTFDB	(63)	Gets a File Descriptor Block	3-175
GTHST	(273)	Obtains ARPANET host information	3-176
GTJFN	(20)	Gets a JFN	
		Short Form	3-179
		Long Form	3-187
GTRPI	(172)	Gets trap information	3-194
GTNCP%	(272)	Obtains information about the NCP	3-195
GTRPW	(171)	Gets trap words	3-197
GTSTS	(24)	Gets a file's status	3-198
GTTYP	(303)	Gets the terminal type number	3-199
HALTF	(170)	Halts the current process	3-200
HFORK	(162)	Halts a process	3-201
HPTIM	(501)	Returns values of high precision clocks	3-202
HSYS	(307)	Halts the system	3-203
IDCNV	(223)	Inputs date and time conversion	3-204
IDTIM	(221)	Inputs date and time	3-205
IDTNC	(231)	Inputs date/time without converting	3-207
IIC	(132)	Initiates software interrupts on specified channels	3-209
INLNM	(503)	Lists job's logical names	3-210
JFNS	(30)	Translates a JFN to a string	3-211
KFORK	(153)	Kills a process	3-214
LGOUT	(3)	Kills a job	3-215
LNMST	(504)	Converts a logical name to a string	3-216
LOGIN	(1)	Logs in a job	3-217
LPINI	(547)	Loads VFU or translation RAM	3-218
MDDT%	(777)	Enters MDDT	3-219
METER%	(766)	Returns EBOX/MBOX clock values	3-220
MRECV	(511)	Receives an IPCF message	3-222
MSEND	(510)	Sends an IPCF message	3-224
MSFRK	(312)	Starts a process in monitor mode	3-229
MSTR	(555)	Performs structure-dependent functions	3-230
MTALN	(774)	Associates magnetic tape drive with logical unit number	3-247
MTOPR	(77)	Performs device-dependent functions	3-248
MTU%	(600)	Performs various functions for MT: devices	3-277
MUTIL	(512)	Performs IPCF control functions	3-279
NIN	(225)	Inputs an integer number	3-285
NODE	(567)	Performs network utility functions	3-286
NOUT	(224)	Outputs an integer number	3-292.1
NTMAN%	(604)	Performs network management functions	3-292.2
ODCNV	(222)	Outputs date and time conversion	3-294
ODTIM	(220)	Outputs date and time	3-295
ODTNC	(230)	Outputs date/time without converting	3-297
OPENF	(21)	Opens a file	3-298
PBIN	(73)	Inputs the next byte	3-303
PBOUT	(74)	Outputs the next byte	3-304

CONTENTS (Cont.)

			Page
PDVOP%	(603)	Manipulates program data vectors	3-305
PEEK	(311)	Obtains monitor data	3-308
PLOCK	(561)	Locks physical pages	3-309
PMAP	(56)	Maps pages	3-310
PMCTL	(560)	Controls physical memory	3-315
PPNST	(557)	Translates project-programmer number to string	3-318
PRARG	(545)	Reads/sets process argument block	3-319
PSOUT	(76)	Outputs a string	3-321
RCDIR	(553)	Translates string to directory number	3-322
RCM	(134)	Reads the channel word mask	3-326
RCUSR	(554)	Translates string to user number	3-327
RCVIM	(751)	Retrieves message from ARPANET special message queue	3-329
RCVOK%	(575)	Retrieves access request from GETOK queue	3-330
RDTTY	(523)	Reads data from primary input designator	3-332
RELD	(71)	Releases a device	3-335
RELSQ	(753)	Deassigns ARPANET special message queue	3-336
RESET	(147)	Resets/initializes the current process	3-337
RFACS	(161)	Reads process' ACs	3-338
RFBSZ	(45)	Reads file's byte size	3-339
RFCOC	(112)	Reads file's control character output	3-340
RFMOD	(107)	Reads a file's mode	3-341
RFORK	(155)	Resumes a process	3-342
RFPOS	(111)	Reads terminal's position	3-343
RFPTR	(43)	Reads file's pointer position	3-344
RFRKH	(165)	Releases a process handle	3-345
RFSTS	(156)	Reads a process' status	3-346
RFTAD	(533)	Reads file's time and dates	3-349
RIN	(54)	Performs random input	3-351
RIR	(144)	Reads software interrupt table addresses	3-352
RIRCM	(143)	Reads inferior reserved channel mask	3-353
RLJFN	(23)	Releases JFNs	3-354
RMAP	(61)	Obtains a handle on a page	3-355
RNAMF	(35)	Renames a file	3-356
ROUT	(55)	Performs random output	3-358
RPACS	(57)	Reads a page's accessibility	3-359
RPCAP	(150)	Reads process capabilities	3-360
RSCAN	(500)	Accepts a new string or uses the last string as input	3-361
RSMAP%	(610)	Reads a section map	3-363
RTFRK	(322)	Returns the handle of a process suspended because of a monitor call intercept	3-364
RTIW	(173)	Reads terminal interrupt word	3-365
RUNTM	(15)	Returns runtime of process or job	3-366
RWM	(135)	Reads waiting channel interrupt word mask	3-367
RWSET	(176)	Releases the working set	3-368
SACTF	(62)	Sets account designator of file	3-369

CONTENTS (Cont.)

			Page
SAVE	(202)	Saves a file as nonsharable	3-370
SCTTY	(324)	Changes controlling terminal	3-371
SCVEC	(301)	Sets entry vector of compatibility package	3-373
SDSTS	(146)	Sets device's status	3-375
SDVEC	(543)	Sets entry vector of RMS	3-376
SETER	(336)	Sets the last error in a process	3-377
SETJB	(541)	Sets job parameters	3-378
SETNM	(210)	Sets program name	3-381
SETSN	(506)	Sets system name for a process	3-382
SEVEC	(204)	Sets entry vector	3-383
SFACS	(160)	Sets process' ACs	3-384
SFBSZ	(46)	Sets file's byte size	3-385
SFCOC	(113)	Sets file's control character output	3-386
SFMOD	(110)	Sets a file's mode	3-387
SFORK	(157)	Starts a process	3-388
SFPOS	(526)	Sets terminal's position	3-389
SFPTR	(27)	Sets file's pointer position	3-390
SFRKV	(201)	Starts process using its entry vector	3-391
SFTAD	(534)	Sets file's time and dates	3-392
SFUST	(551)	Sets author and last writer name strings	3-394
SIBE	(102)	Skips if input buffer is empty	3-395
SIN	(52)	Performs string input	3-396
SINR	(531)	Performs record input	3-398
SIR	(125)	Sets software interrupt table addresses	3-400
SIRCM	(142)	Sets inferior reserved channel mask	3-401
SIZEF	(36)	Gets the size of a file	3-402
SJPRI	(245)	Sets job's priority	3-403
SKED%	(577)	Performs services relating to the class scheduler	3-404
SKPIR	(127)	Tests the state of the software interrupt system	3-409
SMAP%	(767)	Maps one or more contiguous sections of memory	3-410
SMON	(6)	Sets monitor flags	3-415
SNDIM	(750)	Sends a message to ARPANET special message queue	3-417
SNOOP	(516)	Performs system analysis	3-418
SOBE	(103)	Skips if output buffer is empty	3-422
SOBF	(175)	Skips if output buffer is full	3-423
SOUT	(53)	Performs string output	3-424
SOUTR	(532)	Performs record output	3-426
SPACS	(60)	Sets a page's accessibility	3-428
SPJFN	(207)	Sets the primary JFNs	3-429
SPLFK	(314)	Splices a process structure	3-430
SPOOL	(517)	Defines and initializes input spooling	3-431
SPRIW	(243)	Sets the priority word	3-433
SSAVE	(203)	Saves a file as sharable	3-434
STAD	(226)	Sets system date and time	3-436
STCMP	(540)	Compares two strings	3-437

CONTENTS (Cont.)

			Page
STDEV	(120)	Translates string to device designator	3-438
STI	(114)	Simulates terminal input	3-439
STIW	(174)	Sets terminal interrupt word	3-440
STO	(246)	Simulates terminal output	3-442
STPAR	(217)	Sets terminal parameters	3-443
STPPN	(556)	Translates string to project-programmer number	3-444
STSTS	(25)	Sets a file's status	3-445
STTYP	(302)	Sets the terminal type number	3-446
SWJFN	(47)	Swaps two JFNs	3-447
SWTRP%	(573)	Traps for arithmetic underflow or overflow conditions	3-448
SYERR	(527)	Writes data to the system error file	3-450
SYSGT	(16)	Returns information for a system table	3-451
TBADD	(536)	Adds entry to command table	3-452
TBDEL	(535)	Deletes entry from command table	3-453
TBLUK	(537)	Looks up entry in command table	3-454
TEXTI	(524)	Reads input from a terminal or a file	3-457
TFORK	(321)	Sets and removes monitor call intercepts	3-461
THIBR	(770)	Blocks the current process	3-464
TIME	(14)	Returns time system has been up	3-465
TIMER	(522)	Sets time limit for a job	3-466
TLINK	(216)	Controls terminal linking	3-468
TMON	(7)	Tests monitor flags	3-470
TTMSG	(775)	Sends a message to a terminal	3-472
TWAKE	(771)	Wakes a specified job	3-473
UFPGS	(525)	Updates file pages	3-474
USAGE	(564)	Writes entries into the accounting data file	3-475
USRIO	(310)	Places program in user I/O mode	3-478
UTEST	(563)	Tests monitor routines	3-479
UTFRK	(323)	Resumes a process suspended because of a monitor call intercept	3-481
VACCT	(566)	Validates an account	3-482
WAIT	(306)	Dismisses process until interrupt occurs	3-483
WFORK	(163)	Waits for processes to terminate	3-484
WILD%	(565)	Compares wild and non-wild strings	3-485
XGSEV%	(614)	Gets an extended entry vector	3-487
XGTPW%	(612)	Returns the page fail words	3-488
XGVEC%	(606)	Returns an entry vector	3-489
XRIR%	(601)	Reads the addresses of the channel and priority level tables	3-490

CONTENTS (Cont.)

				Page
	XRMAP%	(611)	Acquires a handle on a page	3-491
	XSFRK%	(605)	Starts a process in a non-zero section of memory	3-493
	XSIR%	(602)	Sets the addresses of the channel and priority level tables	3-494
	XSSEV%	(613)	Allows setting of extended entry vector	3-495
	XSVEC%	(607)	Sets or clears the entry vector	3-496

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

APPENDIX B MONSYM

APPENDIX C MACSYM

APPENDIX D ACTSYM

TABLES

TABLE	1-1	P-Field Values for One-word Global Byte Pointers	1-5
	1-2	Source/Destination Designators	1-6
	2-1	File Descriptor Block (FDB)	2-11
	2-2	System Tables	2-25
	2-3	Device Types	2-33
	2-4	PCDR: Status Bits	2-34
	2-5	PCDP: Status Bits	2-35
	2-6	PLPT: Control Characters	2-37
	2-7	PLPT: Status Bits	2-38
	2-8	MTA: Status Bits	2-39
	2-9	JFN Mode Word	2-46
	2-10	Wake-up Classes/CCOC Word Bits	2-49
	2-11	Terminal Characteristics	2-51
	2-12	Software Interrupt Channels	2-58
	2-13	Terminal Interrupt Codes	2-60
	2-14	Process/Job Capabilities	2-64
	2-15	Floating-Point Format Control	2-77
	2-16	Time Zones	2-80
	A-1	ASCII and SIXBIT Collating Sequence and Conversion to EBCDIC	A-1
	A-2	EBCDIC Collating Sequence and Conversion to ASCII	A-3

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. For a more complete description of the monitor calls that can be used to perform ARPANET functions, refer also to the TOPS-20AN 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 is a listing of the system file MONSYM.MAC, which defines many of the symbols used in this manual. Appendix C is a listing of the system file MACSYM.MAC, which contains symbols and macros useful in assembly-language programming. Appendix D is a listing of the system file ACTSYM.MAC, which defines the macros and symbols used with the USAGE monitor call.

REFERENCES

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

Referenced as	Title and Order Number
Monitor Calls User's Guide	TOPS-20 Monitor Calls User's Guide
ARPANET Manual	TOPS-20AN Monitor Calls User's Guide
ARPANET Handbook	ARPANET Protocol Handbook Available from: Network Information Center SRI International Menlo Park, California 94025

Referenced as	Title and Order Number
DECnet Manual	TOPS-20 DECnet-20 Programmer's Guide and Operations Manual for DECnet-20 Version 2, and DECnet-20 User's Guide for DECnet-20 Version 3
Assembler Manual	MACRO Assembler Reference Manual
Link Manual	TOPS-20 LINK Reference Manual
Hardware Reference Manual	DECsystem-10/DECSYSTEM-20 Processor Reference Manual
Commands Reference Manual	TOPS-20 Commands Reference Manual
SPEAR Manual	TOPS-10/TOPS-20 SPEAR Manual
TOPS-20 User's Guide	TOPS-20 User's Guide
Installation Guide	TOPS-20 Software Installation Guide (for KS/KL Model A) or KL Model B Software Installation Guide
Network Management Spec	Network Management Architecture Specification

CHAPTER 1

INTRODUCTION

The TOPS-20 Monitor Calls Reference Manual describes every monitor call in the TOPS-20 system. Monitor calls for ARPANET systems and DECnet systems are also described. The use of these calls, however, is more completely described in the ARPANET Manual and the DECnet Manual.

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 calling location.

Next, there is an optional description of the action taken by the monitor call. Finally, a list of possible error mnemonics ends the monitor call definition.

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, up to a maximum of four accumulators.

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 AC's. If a call returns more data than can be held in four AC's, 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 the newer monitor calls (those defined in TOPS-20 Release 4 or later). Although older calls (those defined before TOPS-20 Release 4) 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:

- a word of data
- the memory address word that contains data
- a page number
- a section number
- a byte pointer
- a file handle
- a source (or destination) designator that defines where to obtain (or send) data
- a process handle
- 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 provides an address space of 32 (decimal) sections, each of which contains 256K words. Thus although 30 bits are used to contain a global address, the section number in such an address can be no longer than 6 bits, making the largest possible address a total of 23 bits long.

An 18-bit address is called a 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 bit 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 zero), it is a global address.

If the address is 18 bits (the section number is zero), 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 zero.

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 6 bits long. In a global address, a section number occupies bits 6 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 paragraphs below 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.

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:

```
-------------------------------------------------
!    P   !              address                 !
-------------------------------------------------
```

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
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. In addition, you cannot use one-word global byte pointers in section 0.

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 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
.CTTRM	0	777777	the process'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.)
	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.

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.

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.

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.

1.2.8 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.

Value	Symbol	Meaning
400000	.FHSLF	current process
400000+n	-	process n, relative to the current process
-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' 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,,fraction

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

fraction/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 the date and time conversion.

1.4 PROCESSING ERRORS

After execution of a monitor call, program control returns to the calling program at one of two locations. The +2 return indicates successful completion of the monitor call. The +1 return is often used to indicate failure of the monitor call to perform its intended function. (Refer to Chapter 3 for specifics on the returns possible from 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. In either case, you can obtain the message associated with the error by using the GETER or ERSTR calls.

Some monitor calls, however, have only a single return (+1), to the instruction following the call. This instruction is executed upon successful completion of the call.

When an error occurs during execution of single-return call, the monitor examines that next instruction. If it is a JUMP instruction, and the AC field is 16 or 17, the monitor transfers control to the address specified in the JUMP instruction.

If the instruction following the call is not a JUMP instruction, the monitor generates a software interrupt. The calling program can process the interrupt by means of the software interrupt system. If the program is not prepared to process the interrupt, the process is usually terminated, and a message is output. (Refer to Section 2.6.)

Instead of a JUMP instruction, you can use one of the following symbols as the instruction following the call:

```
ERJMP address
ERCAL address
```

These symbols correspond to JUMP 16, and JUMP 17, respectively, which are machine no-ops. Because ERJMP and ERCAL are symbols that are defined in MONSYM, you must place a SEARCH MONSYM statement at the top of your program. (See the Assembler Manual for a description of the SEARCH pseudo-op.)

When an ERJMP is used, the monitor simulates a

```
JRST address
```

instruction. This transfers control permanently to the effective address. The address should be the starting address of an error-processing routine. To return control to the program after processing the monitor call error, the error routine must include a JRST instruction.

When an ERCAL is used, the monitor simulates a

```
PUSHJ 17, address
```

instruction. This is a subroutine call. To return control to the code that follows the unsuccessful monitor call, the subroutine must include a

```
POPJ 17,
```

instruction. Note that ERCAL requires accumulator 17 to be set up as a pushdown pointer.

The ERJMP or ERCAL instruction can be used with all monitor calls independent of whether the call has one or two returns. These instructions allow you to process an error without using the software interrupt system. In fact, use of these symbols overrides the software interrupt system.

An ERJMP or ERCAL may also be used following a machine instruction, and will trap for the following conditions:

1. Illegal instruction
2. Illegal memory read
3. Illegal memory write
4. Pushdown list overflow

The ERJMP or ERCAL executes if it is either the next instruction following a monitor call that fails, or the next instruction following a machine instruction that generates the errors shown above; otherwise, it is a no-op.

NOTE

> If an ERJMP or ERCAL executes on an error from a monitor call, the contents of any AC's that would normally contain an error code may be unreliable. Using the GETER monitor call is the sure way to obtain the error code in such a case.

1.5 CONVENTIONS USED IN THIS MANUAL

1.5.1 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.

1.5.2 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.

1.5.3 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. Refer to Appendix B for a listing of MONSYM.

1.5.4 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 2

FUNCTIONAL ORGANIZATION OF JSYS'S

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 monitor calls that perform accounting functions are as follows:

LOGIN	Logs a job into the system
GACCT	Reads a job's account
SACTF	Sets a file's account
GACTF	Reads 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.

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. If the input string cannot possibly match any existing file specification, the system returns an error.

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.
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 <u>TOPS-20 User's Guide</u> 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 (i.e., 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:

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.

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 JSYS's will 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, 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 MTA1: 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.

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

        TITLE FILEIO            ;TITLE OF PROGRAM
        SEARCH MONSYM           ;SEARCH SYSTEM JSYS-SYMBOL LIBRARY

;*** 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

T1==1                           ;JSYS AC'S
T2==2
T3==3
T4==4
T5==5                           ;TEMPORARY AC'S
                                ....
P==17                           ;PUSH DOWN POINTER

;*** 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
        MOVE 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
        MOVE 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
        MOVEM T1,OUTJFN         ;SAVE THE JFN
```

```
;NOW, OPEN THE FILES WE JUST GOT

;    INPUT

        MOVE T1,INJFN           ;RETRIEVE THE INPUT JFN
        MOVE T2,[7B5+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
        MOVE T2,[7B5+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           ;SUPRESS 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.
        TLNN 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

CLOSOF: 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 ***

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.

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 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 (e.g., 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:

1. WRITE - write access

2. OWNER - owner access

3. W/OPR - WHEEL or OPERATOR capabilities enabled

In some cases, separate JSYS's 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 DEC UNCHANGEABLE B29-35(FB%LEN) Length of this file's FDB UNCHANGEABLE
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 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.

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning	
1	.FBCTL (Cont.)	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 DEC. 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. 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

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
1	.FBCTL (Cont.)	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. JSYS WRITE OWNER W/OPR CHFDB N 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

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
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. 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. UNCHANGEABLE RH(FB%DRN): monitor internal directory number of the file (only if B7 of .FBCTL is on). UNCHANGEABLE

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
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. 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

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
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. 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. UNCHANGEABLE
21	.FBBK2	Reserved for DEC UNCHANGEABLE

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
22	.FBBBT	The right half contains the number of pages in the file when the contents were deleted from disk. UNCHANGEABLE 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 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

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
22	.FBBBT (Cont.)	B7(AR%RFL) Restore failed. Set by ARCF to 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 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. UNCHANGEABLE
26	.FBNAM	Pointer to filename block. UNCHANGEABLE

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
27	.FBEXT	Pointer to file type block. UNCHANGEABLE
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) 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

Table 2-1 (Cont.)
File Descriptor Block (FDB)

Word	Symbol	Meaning
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

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

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 (e.g., 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 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 (e.g., 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 JSYS's, such as PMAP, ignore the EOF limit and can read any existing page of the file. However, page-mode JSYS's 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 (e.g., 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 JSYS's. For example, if a PMAP operation is 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 EOFTST
        .
        .                       ;Process byte
        .
EOFTST: MOVE    T1,INJFN        ;Get input JFN
        GTSTS%                  ;Get status of that JFN
        TLNN    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 monitor calls used in referencing files are:

GTJFN	Assigns a JFN to a file
GNJFN	Assigns a JFN to the next file
JFNS	Translates a JFN to a string
WILD%	Compares a wild file specification against a non-wild file specification. Also compares strings.
SPJFN	Sets primary JFN's
GPJFN	Returns primary JFN's
SWJFN	Transposes two JFN's
RLJFN	Releases a JFN
OPENF	Opens a file
CLOSF	Closes a file
CLZFF	Closes a process' files

FUNCTIONAL ORGANIZATION OF JSYS'S

BIN	Reads the next byte
BOUT	Writes the next byte
FLIN	Reads a floating-point number
FLOUT	Writes a floating-point number
NIN	Reads a number
NOUT	Writes a number
PSOUT	Writes string to primary output designator
PBIN	Reads byte from primary input designator
PBOUT	Output byte to primary output designator
SIN	Reads a string
SOUT	Writes a string
SINR	Reads a record
SOUTR	Writes a record
RIN	Reads a byte nonsequentially
ROUT	Writes a byte nonsequentially
DUMPI	Reads data in unbuffered data mode
DUMPO	Writes data in unbuffered data mode
PMAP	Maps pages
RSCAN	Reads and outputs rescan buffer
RDTTY	Reads data from primary input designator
TEXTI	Reads data from terminal or file
CRLNM	Creates a logical name
INLNM	Writes logical names
LNMST	Translates logical name to string
CHFDB	Changes a File Descriptor Block
GTFDB	Reads a File Descriptor Block
SFUST	Changes the author or last writer name string
GFUST	Reads the author or last writer name string
CHKAC	Checks access to a file
ACCES	Specifies access to a directory
DIRST	Translates directory or user number to a string
RCDIR	Translates directory name to number
RCUSR	Translates user name to number
SIZEF	Obtains file's length
SFBSZ	Sets file's byte size
RFBSZ	Reads file's byte size
SFPTR	Sets file's pointer
RFPTR	Reads file's pointer
BKJFN	Backspaces file's pointer
RNAMF	Renames a file
SFTAD	Sets file's time and dates
RFTAD	Reads file's time and dates
STSTS	Sets file's status
GTSTS	Reads file's status
UFPGS	Updates file's pages
DELF	Deletes a file
DELDF	Expunges deleted files
DELNF	Retains specified number of generations of file
FFFFP	Finds first free file page
FFUFD	Finds first used file page

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.

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
BLDTD		Date and time system was generated
DBUGSW		Debugging information
	0	state of operator coverage 0 = unattended 1 = attended 2 = debugging
	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 (e.g., DSK:)
DRMERR		Information on drum errors
	0 1 to n	number of recoverable errors varies depending on type of drum being used
DSKERR		Information on disk errors
	0 1 to n	number of recoverable disk errors 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

Table 2-2 (Cont.)
System Tables

Name	Index	Contents
IMPLT1	c(P2)	ARPANET - 1 fullword for each link: LH: internal connection number, index for: NETAWD NETBAL NETBTC NETBUF NETFSK NETLSK NETSTS or -1 if control link RH: B18-19 00 receive 10 send 11 free 01 delete B20-27 host number B28-35 link number c (index) is derived from bits 24-35 of NETAWD.
IMPLT2	c(P2)	ARPANET - 1 fullword for each link: LH: B0-9 flags B10-17 byte size of buffer RH: address of input buffer c (index) is derived from bits 24-35 of NETAWD.
IMPLT3	c(P2)	ARPANET - 1 fullword for each link: LH: address of output buffer RH: message saved for retransmission c (index) is derived from bits 24-35 of NETAWD.
IMPLT4	c(P2)	ARPANET - 1 full word for each link LH: address of current buffer RH: message allocation in bits c (index) is derived from bits 24-35 of NETAWD.
JBONT	Job #	Owning job for CRJOB-created jobs.

Table 2-2 (Cont.)
System Tables

Name	Index	Contents
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)
JOBTTY	Job #	LH: controlling terminal line number, or -1 if none (i.e., job is detached) RH: reserved for DEC
LOGDES		Logging information
	0	designator for logging information
	1	designator for job 0 and error information
LQLAV		Low queue load averages
NETHST	c(P2)	ARPANET - 1 full word for each internal connection: -1 if no foreign host, otherwise the same as IMPLT5. c (index) is derived from bits 24-35 of NETAWD.
NETAWD	c(P2)	ARPANET - 1 full word for each internal connection: B0-8 link number B9-17 unused B18-23 timeout countdown B24-35 index to link tables c (index) is internal connection (see IMPLT1).
NETBAL	c(P2)	ARPANET - number of bits allocated to each internal connection c (index) is internal connection (see IMPLT1).

Table 2-2 (Cont.)
System Tables

Name	Index	Contents
NETBTC	c(P2)	ARPANET - byte count statistics: the number of bits sent or received over each internal connection since the socket was created. c (index) is internal connection (see IMPLT1).
NETBUF	c(P2)	ARPANET - 1 fullword for each internal connection: LH: bytes per buffer RH: buffer location -1 c (index) is internal connection (see IMPLT1).
NETFSK	c(P2)	ARPANET - foreign socket number (32 bits) for each internal connection c (index) is internal connection (see IMPLT1).
NETLSK	c(P2)	ARPANET - local socket number for each internal connection c (index) is internal connection (see IMPLT1).
NETRDY		ARPANET operational status table
	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

Table 2-2 (Cont.)
System Tables

Name	Index		Contents
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.
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
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
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

Table 2-2 (Cont.)
System Tables

Name	Index	Contents
	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.)
	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	time spent on background tasks. These tasks include low-level data transfer between RSX20F and TOPS-20, and moving data from swapping space to file space.
	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

Table 2-2 (Cont.)
System Tables

Name	Index	Contents
	50	integral of number of wait time without swap waits
	51	count of working set loads
	52	count of runable processes removed from balance set
	53	number of pages removed from memory during process-wide garbage collection. NOTE This table is subject to change (usually additions) as measuring routines are added to the system.
SYSVER		An ASCIZ 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.

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:

GETER	Returns the last error condition
ERSTR	Translates an error number to a string
ESOUT	Returns an error string
SYSGT	Returns values for a system table
GETAB	Returns a word from a system table
GETNM	Returns the program name being used by the job
GETJI	Returns job information for specified job
GJINF	Returns job information for current job
GTAD	Returns the system's date
TIME	Returns the time since the system was restarted
RUNTM	Returns the runtime of a job or process
HPTIM	Returns the high-precision clock values
GTDAL	Returns the disk allocation of a directory
GTRPI	Returns the paging trap information
GTRPW	Returns the trap words

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 as follows:

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
NET:	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.

The following sections describe each 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.

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)
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 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.
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.
B13	MO%IOP	I/O in progress
B14	MO%SER	Software error (e.g., 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.

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 (e.g., 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 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, 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.

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 SIN 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 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 9	8	7	6	5	4	3	2	1	FRAMES
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
0	0	0	0	B32	B33	B34	B35	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 (i.e., bits 0 through 31) of the word are written in four frames on the tape. The rightmost four bits (i.e., 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 9	8	7	6	5	4	3	2	1	FRAMES
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

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 9	8	7	6	5	4	3	2	1	FRAMES
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 7	6	5	4	3	2	1	FRAMES
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.

TRACKS 9	8	7	6	5	4	3	2	1	FRAMES
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 JSYS's 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 TTY's.

Bit	Symbol	Meaning
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 NVT's will not accept a parity bit.

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
	TT%IGN		B19: ignore the other TT%WAK bits
	TT%WKF		B20: formatting control character
	TT%WKN		B21: non-formatting control character
	TT%WKP		B22: punctuation character
	TT%WKA		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	SFMOD	terminal data mode
	.TTBIN		00: no translation
	.TTASC		01: translate both echo and output
	.TTATO		10: translate output only
	.TTATE		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	STPAR	duplex mode
	.TTFDX		00: Full duplex
	.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.

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.

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):

Table 2-10
Wake-up Classes/CCOC Word Bits

ASCII Code	Wake-up 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	C	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		,

Table 2-10 (Cont.)
Wake-up Classes/CCOC Word Bits

ASCII Code	Wake-up Class	CCOC Word(bits)	Character or Control Character
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. ESC(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 SFMOD and .MOSBM set the same mask; however SFMOD computes wakeup classes from the mask while .MOSBM uses character-oriented wakeups.

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, 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

Table 2-11 (Cont.)
Terminal Characteristics

Number	Terminal	Symbol	Characteristics
10	Default	.TTDEF	no mechanical form feed or tab, lower case, full padding, page width 72, page length 66
11	Ideal	.TTIDL	has mechanical form feed and tab, lower case, no padding, no specified width and length
12	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
13	VT50	.TTV50	no mechanical form feed or tab, has upper case only, no padding, page width 80, page length 12, has cursor commands
14	LA30	.TTL30	no mechanical form feed or tab, has upper case only, full padding, page width 80, page length 66
15	GT40	.TTG40	no mechanical form feed or tab, lower case, no padding, page width 80, page length 30
16	LA36	.TTL36	no mechanical form feed or tab, lower case, no padding, page width 132, page length 66
17	VT52	.TTV52	no mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24
20	VT100	.TT100	no mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24, has cursor commands When used in VT52 mode, the terminal type should be set to .TTV52.
21	LA38	.TTL38	no mechanical form feed, has mechanical tab, lower case, no padding, page width 132, page length 66

Table 2-11 (Cont.)
Terminal Characteristics

Number	Terminal	Symbol	Characteristics
22	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 80, page length 24, has cursor commands and color graphics capabilities

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

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.

The following monitor calls are used for device control:

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

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 ARPANET devices 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
PCDP:	0	.GSNRM	CDP: 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.
	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.

Device	Mode	Symbol	Explanation
	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 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:	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.
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.

Device	Mode	Symbol	Explanation
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 (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.
NET:			For ARPANET systems only. Allows buffered or non-buffered byte and string I/O in 8, 32, or 36-bit bytes as follows:
	0		Non-buffered send mode with wait. Waits for state of connection to be other than "request for connection sent" before the OPENF returns. Data is transmitted on every JSYS.
	5		Buffered send mode with wait. Waits for state of connection to be other than "request for connection sent" before the OPENF returns. Data is sent a buffer-load (8000 bits) at a time. The concept of buffering does not apply to input.
	6		Non-buffered send mode with no wait.
	7		Buffered send mode with no wait. Data is sent a buffer-load (8000 bits) at a time. The concept of buffering does not apply to input.
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.

Device	Mode	Symbol	Explanation
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.
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).

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 DEC
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 DEC
15	.ICILI	Illegal instruction[1]
16	.ICIRD	Illegal memory read[1]
17	.ICIWR	Illegal memory write[1]
18		Reserved for DEC
19	.ICIFT	Inferior process termination or forced freeze
20	.ICMSE	System resources exhausted [1]
21		Reserved for DEC
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.)

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.

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 because

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), or

4. the channel has been "reserved" by the superior process (refer to the SIRCM call description),

then the 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 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	.TICCJ 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

Table 2-13 (Cont.)
Terminal Interrupt Codes

Terminal Code	Symbol Character or Condition
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	.TICCZ CTRL/Z
27	.TICES escape (altmode)
28	.TICRB delete (rubout)
29	.TICSP space
30	.TICRF dataset carrier off
31	.TICTI typein
32	.TICTO typeout
33-35	reserved for DEC

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 the job is guaranteed to receive it when the carrier state changes. 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 (i.e., 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 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 (e.g., 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 JSYS's, 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.

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 (i.e., 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.

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 monitor calls for controlling the software interrupt system are:

SIR	Sets the interrupt table addresses for a single-section process
XSIR%	Sets the interrupt table addresses for a multiple-section process
RIR	Reads the interrupt table addresses for a single-section program
XRIR%	Reads the interrupt table addresses for a multiple-section program
EIR	Enables the interrupt system
DIR	Disables the interrupt system
CIS	Clears the interrupt system
SKPIR	Skips if the interrupt system is enabled
AIC	Activates interrupt channels
IIC	Initiates interrupts on specific channels in a process
DIC	Deactivates interrupt channels
RCM	Reads activated channel word mask
RWM	Reads waiting channel word mask
SIRCM	Sets inferior reserved channel mask
RIRCM	Reads inferior reserved channel mask
DEBRK	Dismisses current interrupt
ATI	Assigns terminal code to channel
DTI	Deassigns terminal code
STIW	Sets terminal interrupt word
RTIW	Reads terminal interrupt word
GTRPW	Returns trap words
XGTPW%	Returns page-fail words
SCTTY	Changes source of terminal interrupts

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. 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.

Table 2-14 (Cont.)
Process/Job Capabilities

Bit	Symbol	Meaning
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

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

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 JSYS's allow each installation to write its own access-control program:

1. GETOK%
2. GIVOK%
3. RCVOK%
4. SMON
5. 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.

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 DEC-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% JSYS's 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. JSYS's 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.

2.7.3 Processes and Scheduling

These monitor calls deal with establishing and interrogating the process structure of a job. Refer to the TOPS-20 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 RFORK 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' 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 monitor calls associated with capabilities and processes are:

RPCAP	Returns process capabilities word
EPCAP	Enables process capabilities word
RESET	Resets and initializes current process
CFORK	Creates inferior process
SFORK	Starts a process in section zero
XSFRK%	Starts a process in a non-zero section
HFORK	Halts an inferior process
HALTF	Halts a process
DISMS	Dismisses process for specified amount of time
WAIT	Dismisses process until interrupt occurs
WFORK	Waits for process to terminate
KFORK	Kills one or more processes
FFORK	Freezes one or more processes
RFORK	Resumes one or more processes
TFORK	Sets and removes monitor call intercepts
RTFRK	Returns the handle of the process suspended because of a monitor call intercept
UTFRK	Resumes a process suspended because of a monitor call intercept
RFSTS	Returns process' status
SFACS	Sets process' accumulators
RFACS	Returns process' accumulators
PRARG	Sets or returns process argument block
RFRKH	Releases process handles
GFRKS	Gets current process structure
GFRKH	Gets process handle
SPLFK	Splices a process structure
RMAP	Obtains a handle on a page in a process
SPACS	Sets accessibility of page
RPACS	Returns accessibility of page
RSMAP%	Returns information about the mapping of one section of a process
RWSET	Releases working set
ADBRK	Controls address breaks

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, etc.).

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, AC's, 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.

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 can't 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 JSYS's 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.

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

1. 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.

2. The TOPS-10 compatibility package (PA1050). This has the same restriction that object-time systems have.

3. Any program that uses the TOPS-10 RUN or GETSEG UUO's. These UUO's require mapping into a non-virgin address space.

4. 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

.

.

.

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:

```
Directory        ========================
Area:            !  Directory Section   !
                 !                      !
                 !                      !
                 ------------------------
                 ! Entry Vector Section !
                 ------------------------
                 ! Program Data Vector  !
                 !       Section        !
                 ------------------------
                 ! Terminating Section  !
                 ========================
Data Area:       !     Data Section     !
                 !                      !
                 !                      !
                 !                      !
                 !                      !
                 !                      !
                 !                      !
                 !                      !
                 !                      !
                 ========================
```

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.

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:

```
 0            8 9         17 18                      35
!=======================================================!
!      Identifier code      !      Number of words      !
!           1776            !   (including this word)   !
!                           !   in directory section    !
!=======================================================!
!    Access    !  Page number in file, or 0 if group    !
!     bits     !          of pages is all zero          !
!=======================================================!
!    Repeat    !       Page number in the process       !
!    count     !                                        !
!=======================================================!
/          additional word pairs (as necessary)         /
/            to describe each group of pages            /
/             in the process address space              /
!=======================================================!
! Access bits  !        Page number in the file         !
!=======================================================!
! Repeat count !       Page number in the process       !
!=======================================================!
```

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.

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:

```
0                        17 18                        35
!=======================================================!
!      Identifier code      !      Number of words      !
!           1775            !   (including this word)   !
!                           !  in entry vector section  !
!=======================================================!
!            Number of words in entry vector            !
!=======================================================!
!                Address of entry vector                !
!=======================================================!
```

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 (PDVA's). 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:

```
0                        17 18                        35
!=======================================================!
!      Identifier code      !      Number of words      !
!           1774            !   (including this word)   !
!                           !  in data vector section   !
!=======================================================!
!               Address of data vector 1                !
!=======================================================!
!               Address of data vector 2                !
!=======================================================!
/                           .                           /
/                           .                           /
/                           .                           /
!=======================================================!
!               Address of data vector n                !
!=======================================================!
```

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 containing the group who last modified the program, B3-B11 containing major version number, B12-B17 containing minor version number, and B18-B35 containing edit number.) 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 location 137. This format is not recommended, but is maintained for compatability with older monitors.

The following monitor calls are used in conjunction with save files:

GET	Obtains a saved file
SAVE	Saves a process as nonsharable
SSAVE	Saves a process as sharable
SEVEC	Sets the entry vector for a single-section program
XSVEC%	Sets the entry vector for a multiple-section program
GEVEC	Gets process entry vector of a single-section program
XGVEC%	Gets process entry vector for a multiple-section program
SFRKV	Starts process using its entry vector
XSFRK%	Starts a process using a user-supplied, global PC
SCVEC	Sets compatibility package entry vector
GCVEC	Gets compatibility package entry vector
SDVEC	Sets RMS entry vector
GDVEC	Gets RMS entry vector

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	.PVSTR	Program starting address.
3	.PVREE	Program reenter address.
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 PDV's. When loading a program into memory, LINK executes a PDVOP% call to give the monitor the addresses of the PDV's for that program. The PDVA's are the only data regarding PDV's that the monitor keeps in its data base.

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

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

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 0's. 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.

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 (e.g., 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.
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 0's. 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.

Table 2-15 (Cont.)
Floating-Point Format Control

Bit	Symbol	Meaning
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 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 saving applied from 4AM on the next to last Sunday in April to 3:59:59AM on the next to last Sunday in October.

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 JSYS's. 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.

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 (e.g., months, time zones, AM-PM-NOON-MIDNIGHT) can be represented by any nonambiguous abbreviation (e.g., 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
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 accessable to programs (unless the GTJFN specifically sets bit G1%IIN) and EXEC commands. A visible file appears in a DIRECTORY listing and is accessable to programs and EXEC commands.

: 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 JSYS's are used to implement the archive/virtual disk system:

ARCF
CRDIR
DELDF
DELFN
GTJFN
GNJFN
JFNS
OPENF
RFTAD
SETJB
SFTAD
SMON
TMON

2.11 PRIVILEGED MONITOR CALLS

The following monitor calls are privileged and require the process to have WHEEL or OPERATOR capability enabled. The JSYS's marked with an asterisk ("*") require privileges for specific functions only.

ACCES*	Accesses a directory
ALLOC	Allocates a device to a particular job
ARCF*	Performs archive/virtual-disk operations
ASNSQ	Assigns ARPANET special message queue
ATACH*	Attaches job to new controlling terminal
BOOT	Performs functions required for loading front-end software
CRDIR*	Creates or modifies a directory
CRJOB*	Creates a new job
DELDF*	Expunges deleted files
DELF*	Deletes files
DIAG	Reserves and releases hardware channels
DSKAS	Assigns specific disk addresses

DSKOP	Allows hardware address specification in disk transfers
EFACT	Makes entries to the FACT file
ENQ*	Places a request in ENQ/DEQ resource queue
ENQC*	Returns status of a resource
FLHST	Flushes an ARPANET host
GACCT*	Returns job account information
GIVOK%	Allows/denies access to a protected system resource
GTDIR*	Returns directory information
HALTF*	Halts a process
HSYS	Halts the monitor
LGOUT*	Logs a job out
LPINI	Loads line-printer VFU
MDDT%	Enters MDDT program
MRECV*	Retrieves IPCF message
MSEND*	Sends IPCF message
MSFRK	Starts a process in monitor mode
MSTR*	Performs structure-related functions
MTALN	Associates magnetic tape drive with logical unit number
MTOPR*	Performs device-related functions
MTU%	Performs MT-device functions
MUTIL*	Performs IPCF functions
NODE*	Performs DECnet functions
PEEK	Reads monitor data
PLOCK	Locks physical pages
PMCTL	Controls physical memory
RCVOK%	Services GETOK% requests
SETJB*	Sets job parameters
SFTAD*	Sets file data/time
SFUST*	Sets file author
SJPRI	Sets job priority
SKED%*	Manipulates scheduler data base
SMON	Sets monitor flags
SNOOP	Performs system performance analysis
SPOOL	Performs spooling-related functions
SPRIW	Sets process priority
STAD*	Sets system date/time
STI*	Simulates terminal input
SYERR	Places information in the System Error file
TTMSG*	Sends a message to a terminal
USAGE	Makes entries in accounting file
USRIO	Places program in user I/O mode
UTEST	Monitors executed instructions

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

CHAPTER 3

TOPS-20 MONITOR CALLS

ACCES JSYS 552

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 capabilities 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.
B2(AC%REM)	relinquish the owner access (obtained with the AC%OWN function) to the specified directory and the 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.
B18-B35	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 (PS: on most systems) 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: 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.

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

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

ADBRK JSYS 570

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

RESTRICTIONS: Not available on 2020 hardware.

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 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.

Each function is described in the paragraphs below.

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.

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.

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

AIC JSYS 131

Activates specific software interrupt channels. (Refer to Section 2.6.)

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

ALLOC JSYS 520

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 capabilities 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.

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

ARCF JSYS 247

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 capabilities enabled.

ACCEPTS IN AC1: JFN

AC2: Function code. 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 capabilities 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 capabilities 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 privileges enabled to be used separately): B0 AR%CR1 Clear information for run 1. B1 AR%CR2 Clear information for run 2.

Code	Symbol	Function
5	.ARSST	Set tape information for a file. This function is used to set information for the first, second, or both tape runs. AR%O1 and AR%O2 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%O1)	Set information for run 1.
		B1(AR%O2)	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.	
6	.ARPSZ	Number of pages in the file. This word can be set only if AR%O1 and AR%O2 are set first.	

Code	Symbol	Function
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 capabilities are 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 capabilities.
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 capabilities 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, archived, or archive-pending files

ARCX17: FDB improper format for ARCF

ARCX18: Retrieval wait cannot be fulfilled for waiting process

ARCX19: Migration already pending

ASND JSYS 70

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

ASNSQ JSYS 752

Assigns a special message queue to a job. See ARPANET manual for more details.

RESTRICTIONS: for ARPANET systems only. Requires NET WIZARD capabilities enabled.

ACCEPTS IN AC1: mask

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

ATACH JSYS 116

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 capabilities 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-B35 (AT%JOB) job number of the desired job.

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 ASCIZ 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.

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

ATI JSYS 137

Assigns a terminal code to a software interrupt channel. (Refer to Section 2.6.) This call also sets the corresponding bit in the process' 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

ATNVT JSYS 274

Creates the Network Virtual Terminal (NVT) connection. See the ARPANET manual for more details.

RESTRICTIONS:		for use with ARPANET 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
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
B2	AT%NTP	If set, this bit indicates new TELNET protocol. If clear, this bit indicates old TELNET protocol.

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)

BIN JSYS 50

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.)

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

BKJFN JSYS 42

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

BOOT JSYS 562

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.

On 2040,2050, and 2060 hardware, the BOOT JSYS is used to load and dump a PDP-11 connected to a DTE20. On 2020 hardware, the BOOT JSYS loads and dumps the KMC11 microcode, dumps line counters, controls DDCMP on a line, performs multidrop functions, and loads and dumps front-end images.

RESTRICTIONS: requires WHEEL or OPERATOR capabilities 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. Note that in the discussion of each function, the applicable processor is indicated as follows:

Group	Processor
A	2020
B	2040,2050,2060

BOOT JSYS Functions:

Code	Symbol	Meaning
0	.BTROM	Puts the line in MOP (Maintenance-Operation Protocol) mode and activates the bootstrap ROM in the front end.

Applicable Hardware: processor group A, B

Processor Group A Argument Block

0	.BTPRT	Line number

Activates the hardware ROM bootstrap in the communications front end.

Processor Group B Argument Block

0	.BTDTE	DTE-20 number
1	.BTERR	Error status flags returned on failure of the call

Code	Symbol	Meaning
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. Applicable Hardware: processor group A,B Processor Group A Argument Block 0 .BTPRT Line number 1 Not used, must be zero 2 .BTSEC Address of bootstrap program to be loaded Processor Group B 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: the first two bytes of data supplied by the caller must be a count of the remaining number of data bytes. Applicable Hardware: processor group A,B Processor Group A Argument Block 0 .BTDTE line number 1 Not used and must be zero 2 Not used and must be zero 3 Not used and must be zero 4 .BTCNT Number of bytes to transfer 5 .BTLPT Pointer to data to be loaded

Code	Symbol	Meaning
2	.BTLOD (Cont.)	Processor Group B 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 B0(BT%BEL) Send a doorbell to the front end to indicate when the setup is complete and the transfer can begin.
4	.BTCNT	Number of bytes to transfer
5	.BTLPT	Pointer to data to be loaded

Code	Symbol	Meaning
3	.BTDMP	Dump the communications front-end memory using the ROM bootstrap program. The caller must activate the ROM bootstrap (function .BTROM) before dumping memory. Subsequent .BTDMP functions to dump memory start where the previous dump terminated unless the ROM bootstrap is activated again by a .BTROM function. The caller blocks until the transfer is complete. Applicable Hardware: processor group B 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	.BTDPT	Pointer to where the data is to be dumped in TOPS-20

Code	Symbol	Meaning
4	.BTIPR	Generates and links a DDCMP Station Table. Starts up lines and terminals not previously known to the system. (Must be issued once for each multidrop terminal being started up.) Applicable Hardware: Processor group A (See below for group B.) Processor Group A Argument Block 0 .BTPRT Drop,,line number 1 .BTPRV Version number of protocol to be used Processor Group A protocol types are: Symbol — Meaning .VNDDC (2) DDCMP protocol .VNMOP (3) MOP (DDCMP maintenance) mode .VNCNL (4) Controller loopback .VNCBL (5) Cable loopback
	.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. Applicable Hardware: processor group B Processor Group B Argument Block 0 .BTDTE DTE-20 number 1 .BTPRV Version number of the protocol to be used Processor Group B protocol types are: 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. After successful execution of this function, TOPS-20 ignores interrupts from the given DTE-20 or line. Applicable Hardware: processor group A,B Processor Group A Argument Block 0 .BTPRT Line number Processor Group B Argument Block 0 .BTDTE DTE-20 number

Code	Symbol	Meaning
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.

Applicable Hardware: processor groups A,B

Processor Group A Argument Block

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

Processor Group A protocol types are:

Symbol	Meaning
.VNDDC (2)	DDCMP protocol
.VNMOP (3)	MOP (DDCMP maintenance) mode
.VNCNL (4)	Controller loopback
.VNCBL (5)	Cable loopback

Processor Group B Argument Block

0	.BTDTE	DTE-20 number
1	.BTCOD	Returned protocol version type. If no protocol is running, this word contains -1.

Processor Group B protocol types are:

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

Code	Symbol	Meaning
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.

Applicable Hardware: processor group B

Argument Block

0	.BTDTE	DTE-20 number

Code	Symbol	Meaning
10	.BTRMP	Read data from the communications front end using the previously loaded secondary or tertiary bootstrap program.

Applicable Hardware: processor group A (See below for group B.)

Processor Group A Argument Block

0	.BTPRT	Line number
1		Not used and must be zero
2		Not used and must be zero
3		Not used and must be zero
4	.BTCNT	Number of bytes to transfer
5	.BTLPT	Pointer to where the data is to be dumped in TOPS-20

.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.

Applicable Hardware: processor group B

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

Code	Symbol	Meaning
11	.BTKML	Load a KMC11. This function will optionally load the CRAM, DRAM, and the four UNIBUS registers. Before the KMC11 is loaded, the system verifies that each bit in UNIBUS registers can be set and cleared. Before the DRAM is loaded, the system verifies that each bit in the entire DRAM can be set and cleared. After the CRAM, DRAM, and registers are loaded, they are verified to ensure that the data was properly loaded. If the register data is not supplied, the UNIBUS registers are cleared before the KMC11 is started.

Applicable Hardware: processor group A

Argument Block

0	.BTKMC	KMC11 address
1	.BTKER	Error flags returned in left half and bad data word (16 bit) returned in right half
		B0 (BT%CVE) CRAM verify error
		B1 (BT%DVE) DRAM verify error
		B2 (BT%RVE) Register verify error
2	.BTKCC	Count of CRAM data
3	.BTKCP	Pointer to CRAM data (16-bit data)
4	.BTKDC	Count of DRAM data
5	.BTKDP	Pointer to DRAM data (8-bit data)
6	.BTKRC	Count of register data
7	.BTKRP	Pointer to register data (16-bit data)
8	.BTKSA	If bit 0 of this word is set, then the right halfword contains the starting address; otherwise, this word is ignored.
		B0 (BT%KSA) Right halfword is set; start KMC11

Code	Symbol	Meaning
12	.BTKMD	Dump a KMC11. This function optionally dumps the CRAM, DRAM, and registers if space is provided. The registers are SEL0, SEL2, SEL4, SEL6, INDATA, OUTDATA, INBA, OUTBA, and MISC*400+NPR. Applicable Hardware: processor group A Argument Block 0 .BTKMC KMC11 address 1 Not used, must be zero 2 .BTKCC Count of CRAM data 3 .BTKCP Pointer to CRAM data (16-bit data) 4 .BTKDC Count of DRAM data 5 .BTKDP Pointer to DRAM data (8-bit data) 6 .BTKRC Count of register data 7 .BTKRP Pointer to area for storing register data (16-bit data)
13	.BTRLC	Return line counters. All counters are positive numbers. Applicable Hardware: processor group A Argument Block 0 .BTPRT Port number 1 .BTZTM Time since counters were last zeroed 2 .BTSCC Number of status counts to return 3 .BTSCP Pointer to area to receive status counters 4 .BTRCC Number of receive counts to return 5 .BTRCP Pointer to area to store receive counters 6 .BTTCC Number of transmit counts to return 7 .BTTCP Pointer to area to receive transmit counters

Code	Symbol	Meaning
14	.BTCLI	Convert line id to port number Applicable Hardware: processor groups A,B Argument Block 0 .BTPRT Port number 1 .BTLID Pointer to ASCIZ line id
15	.BTCPN	Convert NSP port number to line id Applicable Hardware: processor groups A,B Argument Block 0 .BTPRT Port number 1 .BTLID Pointer to ASCIZ line id
16	.BTSTA	Set the station's polling state to active to cause the terminal to be polled, or set it to idle to prevent the terminal from being polled. Applicable Hardware: VT62 on processor group A Argument Block 0 .BTPRT Drop,,Line number 1 .BTCOD Flags: 0 .BTACT Set line active 1 .BTIDL Set line idle
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. Applicable Hardware: DN60 on KL-10 Model B processor 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.

Code	Symbol			Meaning
16	.BTD60 (Cont.)			
		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.
		35	D6%NT6	The front end is not running the 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	The time TOPS-20 queued the header to the DTE
		10	.BT6TRD	The time
		11	.BT6TDD	The time
		12	.BT6TFR	The time TOPS-20 satisfied the request

Code	Symbol	Meaning
17	.BTSSP	Set the start-up priority value. This value specifies the relative frequency at which start-ups are attempted. That is, for a value of N, each active station is polled N times for each DDCMP start. This is used to prevent unresponsive stations from deteriorating performance of a multidrop line. N is a 4-bit field in which truncation occurs if the field size is exceeded. If N is zero, stations in start-up mode will be polled along with active stations. Applicable Hardware: VT62 on processor group A Argument Block 0 .BTPRT Line number 1 .BTSPR Start priority count
20	.BTSTP	Set the polling priority. This parameter is maintained in the Station Table to specify the relative polling priority of a station. If this feature is not used, all priority values default to 1 and polling proceeds in a round robin manner. Applicable Hardware: VT62 on processor group A Argument Block 0 .BTPRT Drop,,line number 1 .BTPRI Priority value Typical range: 1 (high) to 5 (low)
21	.BTSDD	Send a DDCMP message. A DDCMP message will be queued for transmission on the specified line. Applicable Hardware: Processor group A Argument Block 0 .BTPRT Drop,,line 1 .BTMSG Address of message or byte pointer to message 2 .BTLEN Byte count of message

Code	Symbol	Meaning
22	.BTRDD	Receive a DDCMP message. An item from the DDCMP input queue is returned or .BTLEN is set to zero if the queue is empty. Items on the queue will be data segments, completion postings, or status postings (station going up or down). Applicable Hardware: Processor group A Argument Block 0 .BTPRT Line number 1 .BTMSG Address of buffer or byte pointer to buffer 2 .BTLEN Size of user buffer For data messages, the byte count of the message is returned in .BTLEN. If the buffer is too small, the JSYS will fail. For completion postings, the following are returned in .BTLEN: BT%CTL (1B0) + .BTSUP (1) - station came up BT%CTL (1B0) + .BTSDW (2) - station went down BT%CTL (1B0) + .BTCMP (3) - transmit complete BT%CTL (1B0) + .BTSSF (4) - a start-up failed .BTPRT will contain the drop that this message pertains to.
23	.BTCHN	Set the interrupt channel so that software interrupts will be generated when input data is available. Applicable Hardware: Processor group A Argument Block 0 .BTPRT Drop,,line number 1 .BTCOD Software interrupt channel

Code	Symbol	Meaning
24	.BTSLS	Set type of line service to be done on a synchronous communications line. Applicable Hardware: Processor group A Argument Block 0 .BTPRT Drop,,line number 1 .BTCOD Define protocol Protocol values can be: 0 .BTNSP NSP protocol 1 .BTDCP DDCMP protocol

The error status flag returned in word .BTERR on failure of a BOOT call (for group B processors) 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

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

BOUT JSYS 51

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: the 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

CACCT JSYS 4

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.

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

CFIBF JSYS 100

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

DESX3: JFN is not assigned

DESX5: File is not open

DEVX2: Device already assigned to another job

TTYX01: Line is not active

CFOBF JSYS 101

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

CFORK JSYS 152

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

ACCEPTS IN AC1: B0(CR%MAP) make the inferior process' map the same as the current process' 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' capabilities the same as the current process'. If this bit is not on, the inferior process has no special capabilities.

B3(CR%ACS) set the inferior process' ACs from the block whose address is in AC2. If this bit is not on, the inferior process' 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-B35 (CR%PCV) PC value for the inferior process if CR%ST 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:

FRKHX6: All relative process handles in use

FRKHX8: Illegal to manipulate an execute-only process

CFRKX3: Insufficient system resources

CHFDB JSYS 64

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.)

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-B17 (CF%DSP)	index into FDB indicating word to be changed
	B18-B35 (CF%JFN)	JFN (for a disk file)
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. And to keep this 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.

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

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

CHKAC JSYS 521

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.
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

CIS JSYS 141

Clears the software interrupt system for the current process. Clears all interrupts in progress and all waiting interrupts.

RETURNS +1: always

CLOSF JSYS 22

Closes a specific file or all files.

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-B35 (CO%JFN)	JFN of the file being closed

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.

CLZFF JSYS 34

Closes process' 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-B35 (CZ%PRH)	process handle

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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3:	Invalid use of multiple process handle
IOX11:	Quota exceeded
IOX34:	Disk full
IOX35:	Unable to allocate disk - structure damaged

COMND JSYS 544

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.

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:

1. Byte pointer to the beginning of the prompting-text buffer (.CMRTY). This pointer is also called the CTRL/R buffer byte pointer, since a CTRL/R causes COMND to redisplay the prompt contained in this buffer, along with anything the user typed on the command line before he typed the CTRL/R.

   The buffer that contains the prompt need not be contiguous with the buffer containing the remainder of the command line.

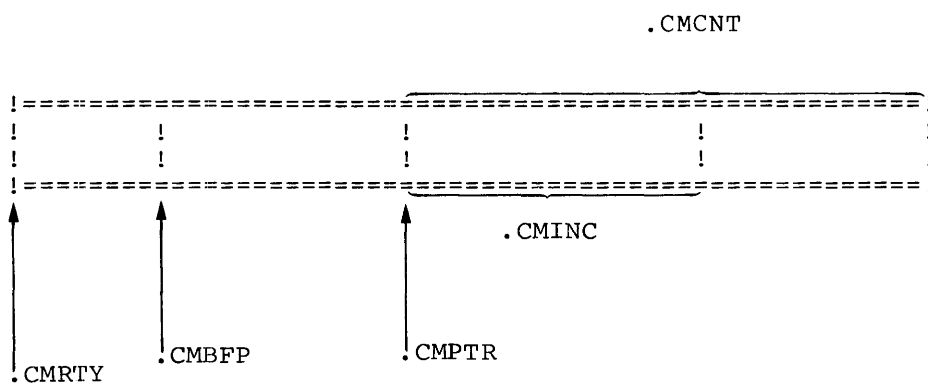
2. Byte pointer to the beginning of the buffer that contains the user's input (.CMBFP). This is the limit back to which the user can edit.

3. Byte pointer to the beginning of the next field of the command line to be parsed (.CMPTR).

4. Count of the space remaining in the text input buffer (.CMCNT).

5. Count of the number of characters in the buffer that have not yet been parsed (.CMINC).

The illustration below is a logical arrangement of the byte pointers and counts. Remember that the prompting text buffer need not be adjacent to the text buffer.



These byte pointers and other information are contained in a **command state block** whose address is given as an argument to the COMND monitor call. The .CMINI function initializes these pointers.

COMND Parses a command line field by field. COMND substitutes default values for missing fields in the command line when the user types a carriage return, ESC, CTRL/F, or question mark. These characters are called action characters because they cause the system to act on the command as typed so far. Other characters that terminate a field are space, tab, slash, comma, and any other nonalphanumeric character.

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

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 uniquely identifies 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           !
        !=======================================================!
```

Command State Block

Word	Symbol	Meaning
0	.CMFLG	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. 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. 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. 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.
1	.CMIOJ	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).
2	.CMRTY	Byte pointer to the beginning of the prompting text.
3	.CMBFP	Byte pointer to the beginning of the user's input. The user cannot edit back past this pointer.
4	.CMPTR	Byte pointer to the beginning of the next field to be parsed.
5	.CMCNT	Count of the space remaining in the buffer after the .CMPTR pointer.
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.

Word	Symbol	Meaning
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. 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   !
 .CMFNP!   code     !    flags    !    descriptor block        !
       !-------------------------------------------------------!
 .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-B8(CM%FNC) Function code B9-B17(CM%FFL) Function-specific flags B18-B35(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.
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-B6 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.
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.)

Code	Symbol		Meaning
0	.CMKEY (Cont.)	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. For COMND to input a guide word, the guide word field must be delimited by parentheses. Word .CMDAT contains a byte pointer to an ASCIZ 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.	

Code	Symbol	Meaning
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 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.

Code	Symbol	Meaning
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 .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.

Code	Symbol	Meaning
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.
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.

Code	Symbol	Meaning
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.
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.)

Code	Symbol	Meaning
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 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.

Code	Symbol	Meaning
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 (e.g., + 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 (e.g., 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.
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
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.
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 (e.g., 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-B8(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.

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 nn where nn is the radix.

Function	Message
.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.
.CMIFI (input file) .CMOFI (output file) .CMFIL (any file)	The help message output depends on the settings of certain bits in the GTJFN call. If bit GJ%OLD is off and bit GJ%FOU is on, the help message is output filespec 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 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 FLDDB. 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:

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	.CMDAT	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.
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 one of the following error codes 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
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
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 BLDB3):

```
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/NOUT and FLIN/FLOUT error can occur on the number functions.

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

CRDIR JSYS 240

Creates, changes, or deletes a directory entry.

RESTRICTIONS: some functions require WHEEL or OPERATOR capabilities 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

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

B17(CD%DEL) delete this directory entry

B18-B35(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:

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 existing directories to revert to older parameters. The process must have WHEEL or OPERATOR capability enabled to set this bit.

Word	Symbol	Meaning	
0	.CDLEN (Cont.)	B2(CD%NED)	Set default on-line expiration date from word .CDDNE. Currently not implemented.
		B3(CD%FED)	Set default on-line expiration date from word .CDDFE. Currently not implemented.
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 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.
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).	

Word	Symbol	Meaning
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 of last login.
13	.CDUGP	address of user group list for this directory.
14	.CDDGP	address of directory group list.
15	.CDSDQ	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.
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.
20	.CDDNE	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. This word is currently reserved and is not implemented.
21	.CDDFE	default off-line expunge date and time. Otherwise similar to .CDDNE (above). This word is currently reserved and is not implemented.

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 examined. If the bit is off, it indicates that the argument should be defaulted.

The following table 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-B17 of AC2 is off, the corresponding parameter is not affected.

When CRDIR is being executed to delete a directory, the settings of B0-B17 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

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

CRJOB JSYS 2

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 capabilities 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

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.

Bit	Symbol	Meaning
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.

Bit	Symbol	Meaning
4	CJ%ETF	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:

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.

Bit	Symbol	Meaning
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.
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 the new job cannot be logged out until the caller releases ownership of it. 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.

Bit	Symbol	Meaning
14	CJ%SLO	Send an IPCF message to the PID supplied in the argument block when the new job is logged out. If 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,,.IPCLO
1	N,,# of job logged out. N is the count of the remaining words in this message (currently 10 octal).
2	flags,,reserved

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 MEXEC(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.

Word	Contents
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

Bit	Symbol	Meaning
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.

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:

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 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).

Word	Symbol	Meaning
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

CRJBX2: Illegal for created job to enter MINI-EXEC

CRJBX4: Terminal is not available

CRJBX5: Unknown name for LOGIN

CRJBX6: Insufficient system resources

CRLNM JSYS 502

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. (Refer to Section 2.2.2 and to the INLNM and LNMST monitor call descriptions for more information on logical names.)

ACCEPTS IN AC1: function code

AC2: byte pointer to the logical name (A terminating colon is optional in the logical name.)

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:

0	.CLNJ1	delete one logical name from the job
1	.CLNS1	delete one logical name from the system
2	.CLNJA	delete all logical names from the job
3	.CLNSA	delete all logical names from the system
4	.CLNJB	create a logical name for the job
5	.CLNSY	create a logical name for the system

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

CVHST JSYS 276

Converts a host number to a primary name.

RESTRICTIONS: for use with ARPANET only

ACCEPTS IN AC1: destination designator for the ASCIZ string

AC2: host number

RETURNS +1: failure
(Use the GETER call to obtain the error code.)

+2: success, host name string returned to area designated by AC1

CVHST ERROR MNEMONICS:

CVHST1: No string for that host number

CVSKT JSYS 275

Converts a local socket number to absolute form.

RESTRICTIONS: for use with ARPANET only

ACCEPTS IN AC1: JFN

RETURNS +1: failure, error code in AC1

+2: success, absolute socket number in AC2

CVSKT ERROR MNEMONICS:

CVHST1: No string for that host number

CVSKX1: Invalid JFN

CVSKX2: Local socket invalid in this context

DEBRK JSYS 136

Dismisses the software interrupt routine in progress and resumes the process at the location specified by the PC stored in the priority level table. (Refer to 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:

DBRKX1: No interrupts in progress

DELDF JSYS 67

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 capabilities enabled.

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

DELF JSYS 26

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 capabilities 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-B35 (DF%JFN)	JFN of the file being deleted.

RETURNS

+1: failure, error code in AC1

+2: success, JFN is released unless B0(DF%NRJ) is on or the file is open.

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 DELF call takes the +1 return if the JFN is assigned to a nondirectory device.

DELF 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
DELFX1:	Delete access required
DELFX2:	File cannot be expunged because it is currently opened
DELFX3:	System scratch area depleted; file not deleted
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
DELFX9:	File is not a directory file
DELF10:	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"
WHELX1:	WHEEL or OPERATOR capability required

DELNF JSYS 317

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 DELNF to fail if bit AR%NDL is set in word .FBBBT of the FDB or if a complete set of tape backup information is not in the FDB.

B18-B35 JFN of the file being deleted

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

DEQ JSYS 514

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 dequeueing 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.

Refer to the TOPS-20 Monitor Calls User's Guide for an overview and description of the Enqueue/Dequeue facility.

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
	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 dequeueing 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
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

DEVST JSYS 121

Translates the given device designator to its corresponding ASCIZ 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

DFIN JSYS 234

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.

+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

DFOUT JSYS 235

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. (Refer to 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

DIAG JSYS 530

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 capabilities 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. Argument block: Word Contents 0 function code 1 device address 2 time limit in milliseconds
2	.DGACH	Assign the channel and all devices. Argument block: Word Contents 0 function code 1 device address
3	.DGRCH	Release the channel and all assigned devices. Argument block: Word Contents 0 function code 1 device address

Function	Symbol	Meaning
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). Argument block: 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. Argument block: 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. Argument block: Word Contents 0 function code 1 device address 2-5 4-word channel logout area
100	.DGGEM	Get memory (for TGHA). Argument block: 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

Function	Symbol	Meaning
100	.DGGEM (Cont.)	

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.)

9. The monitor has set up proper dispatch if TGHA specified an AR/ARX trap address.

101 .DGREM Release memory (for TGHA)

Argument block:

Word	Contents
0	function code

Function	Symbol	Meaning
102	.DGPDL	Inform the monitor that a device previously unknown to it is now available for use (is now online). This functon 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)

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:

```
 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

DIBE JSYS 212

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

DIC JSYS 133

Deactivates the specified software interrupt channels. (Refer to 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

DIR JSYS 130

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.

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

DIRST JSYS 41

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

STRX01: Structure is not mounted

IOX11: Quota exceeded

IOX34: Disk full

IOX35: Unable to allocate disk - structure damaged

DISMS JSYS 167

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).

DOBE JSYS 104

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

DSKAS JSYS 244

Assigns or deassigns specific disk addresses.

RESTRICTIONS: requires WHEEL or OPERATOR capabilities 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).

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.

B18-B35 (DA%ADR) disk address

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

DSKOP JSYS 242

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 capabilities enabled. Some functions can be performed with MAINTENANCE capabilities enabled.

ACCEPTS IN AC1: B0-B1(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-B6(DOP%CN) channel number
- B7-B12(DOP%UN) unit number
- B13-B35(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-B10(DOP%SN) structure designator flag (0 is structure PS:). A value of -1 means the structure is indicated by the structure designator (refer to Section 2.4) in AC4.
- B11-B35(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
B10(DOP%EO)	error if unit offline
B11(DOP%IL)	inhibit error logging
B12(DOP%IR)	inhibit error recovery
B14(DOP%WR)	write data to the disk. If this bit is off, read data from the disk.
B18-B35 (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 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-B11(DOP%C2) channel number
B12-B23(DOP%K2) controller number
B13-B35(DOP%U2) unit number

RETURNS +1: always, AC1 is nonzero if an error occurred, or zero if no error occurred.

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:

WHELX1: WHEEL or OPERATOR capability required

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

DTACH JSYS 115

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.

DTI JSYS 140

Deassigns a terminal interrupt code.

ACCEPTS IN AC1: terminal interrupt code; refer to 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

DUMPI JSYS 65

Reads data words into memory in unbuffered data mode. The file must be open for data mode 17. (Refer to 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-B35 (DM%PTR) address of command list in memory

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.

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

DUMPO JSYS 66

Writes data words from memory in unbuffered data mode. The file must be open for data mode 17. (Refer to 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-B35 (DM%PTR)	address of command list in memory	
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 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

IOX34: Disk full

IOX35: Unable to allocate disk - structure damaged

DVCHR JSYS 117

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
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

EIR JSYS 126

Enables the software interrupt system for a process. (Refer to 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

ENQ JSYS 513

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 TOPS-20 Monitor Calls User's Guide for an overview and description of the Enqueue/Dequeue facility.

RESTRICTIONS: some functions require WHEEL or OPERATOR capabilities 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: 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 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 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.

Code	Symbol	Meaning
3	.ENQMA (Cont.)	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 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.

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	software interrupt channel number in the left half, and the request ID 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 below).
3	.ENQUC	pointer to a string or a 5B2+33-bit user code.
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.
5	.ENQMS	address of a resource mask block.
	. . .	
n-4		flags and level number in the left half, and JFN, -1, -2, or -3 in the right half.
n-3		pointer to a string or a 5B2+33-bit user code.
n-2		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.
n-1		address of a resource mask block.

The following paragraphs describe the words in the argument block.

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 individ is notual 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 currently used by the system but 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.
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-B17	Level number associated with this resource.

(EN%LVL)

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.

.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' ENQ call has not specified a mask block.

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' mask block are not on in another process' 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

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

ENQX22: Invalid mask block length

ENQX23: Mismatched mask block lengths

DESX8: File is not on disk

ENQC JSYS 515

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 TOPS-20 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: some functions require WHEEL or OPERATOR capabilities 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: 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         !
!=======================================================!
```

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-B17 (EN%LVL)	The level number of the resource.
B18-B35 (EN%JOB)	Job number of the owner of the lock. 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: 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, 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.
.ENQCC	One word containing the new quota in the left half and a job number in the right half.	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.

Function	Argument Block	Meaning
.ENQCD	A block of n words. The first word is the length of the block (n). Remaining words contain the returned data. (See below.)	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       !
           !=======================================================!
```

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.

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
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
DESX8:	File is not on disk

EPCAP JSYS 151

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

ERSTR JSYS 11

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

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

ESOUT JSYS 313

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 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.)

FFFFP JSYS 31

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

FFORK JSYS 154

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 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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

FFUFP JSYS 211

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

FLHST JSYS 277

"Flushes" an ARPANET host. Causes the NCP tables containing that host's status information to be purged of all information regarding previous or partially terminated connections between the sending and receiving hosts of the connection. All connections to the flushed host are closed.

RESTRICTIONS:	for ARPANET systems only. Requires OPERATOR, WHEEL, or NET WIZARD capabilities enabled.
ACCEPTS IN AC1:	number of host to be flushed
RETURNS +1:	always

FLIN JSYS 232

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

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

FLOUT JSYS 233

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

GACCT JSYS 546

Returns the current account for the specified job.

RESTRICTIONS: some functions require Confidential Information Access, WHEEL, or OPERATOR capabilities 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

GACTF JSYS 37

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.

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

GCVEC JSYS 300

Returns the entry vector and the UUO 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 UUO 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

GDSKC JSYS 214

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.

GDSKC ERROR MNEMONICS:

DEVX1: Invalid device designator

GDSTS JSYS 145

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 ARPANET, the return sequence for network-connection files is:

AC2: connection state (octal values 01 thru 16) in bits 0 thru 3

AC3: foreign host number (octal)

AC4: foreign socket 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.

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

GDVEC JSYS 542

Returns the entry vector for the Record Management System (RMS).

RESTRICTIONS: Requires RMS software (currently available only with BASIC and COBOL)

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.

Generates an illegal instruction interrupt on error conditions below.

GDVEC ERROR MNEMONICS:

ILINS5: RMS facility is not available

GET JSYS 200

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 (PDVA's) 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.

ACCEPTS IN AC1: process handle,, flag bits and a JFN

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.

Word	Symbol	Meaning
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.

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%HGH	.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. (Operator or wheel privileges are required for use of this bit.)
4	GT%CSH	Place in cache the name of the program being loaded into memory. (Operator or wheel privileges are 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' page are overwritten. If the file contains data for only a portion of the process' 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 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.

GETAB JSYS 10

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 Appendix B for the system table definitions.)

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

GETER JSYS 12

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

GETJI JSYS 507

Obtains information about the specified job.

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

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)

Word	Symbol	Meaning
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).
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). -1 is returned if the system time and date were not set when the job started.
26	.JIBCH	Batch stream number and batch flags

Field	Symbol	Contents
3B1	OB%WTO	Write-to-operator capabilities

Value	Symbol	Meaning
0	.OBALL	WTO (write to operator) and WTOR (write to operator with reply)
1	.OBNWR	No WTOR allowed
2	.OBNOM	No message allowed

Field	Symbol	Contents
1B10	OB%BSS	Indicates that field OB%BSN (below) contains a batch-stream number
177B17	OB%BSN	Batch-stream number

Word	Symbol	Meaning
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.

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

GETNM JSYS 177

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.

RETURNS +1: always, with SIXBIT name of program in AC1

GETOK% JSYS 574

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
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

Code	Symbol	Meaning
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.
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

Code	Symbol	Meaning
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

Code	Symbol	Meaning
13	.GOMDD	Allow entry to MDDT

Argument block (user-specified):

Word	Symbol	Contents
0	.GEERB	Error block address

Code	Symbol	Meaning
14	.GOCLS	Set scheduler class for a job

Argument block (user-specified):

Word	Symbol	Contents
0	.GEERB	Error block address
1	.GEJOB	Job number
2	.GECLS	Class desired

Code	Symbol	Meaning
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

Code	Symbol	Meaning
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)

Code	Symbol	Meaning
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
21	.GODNA	Allow access to DECNET Argument block (user-specified): Word Symbol Contents 0 .GEERB Error block address
22	.GOANA	Allow ARPANET 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 .GOTJB Target job number 1 .GEADD Source TTY number
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	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 offline

MONX01: Insufficient system resources

GOKER1: Illegal function

GOKER2: Request denied by Access Control Facility

GEVEC JSYS 205

Returns the section-relative entry vector of the specified process. (Refer to 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' entry vector word in AC2

The SEVEC monitor call can be used to set the process' entry vector. (Refer to 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

GFRKH JSYS 164

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).

Generates an illegal instruction interrupt on error conditions below.

GFRKH ERROR MNEMONICS:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

FRKHX6: All relative process handles in use

GFRKX1: Invalid process handle

GFRKS JSYS 166

Returns the process structure of the current job from a given process downward.

RESTRICTIONS: some functions require WHEEL or OPERATOR capability

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.

Table format

3 words per entry	parallel pointer	inferior pointer
	superior pointer	process handle or 0 if GF%GFH was off, or when no more process handles are left for the process
This word is -1 if GF%GFS is off.	status word	

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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

FRKHX6: All relative process handles in use

GFKSX1: Area too small to hold process structure

GFUST JSYS 550

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

GIVOK% JSYS 576

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 enabled WHEEL or OPERATOR capability.

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

GJINF JSYS 13

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.

GNJFN JSYS 17

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 (i.e., 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.

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

OPNX1: File is already open

STRX09: Prior structure mount required

GPJFN JSYS 206

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.

The SPJFN monitor call can be used to set the primary JFNs. If this call has not been given, the GPJFN call returns -1 in AC2.

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

GTAD JSYS 227

Returns the current date in the internal system format. (Refer to 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.

GTDAL JSYS 305

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

GTDIR JSYS 241

Returns information about the given directory.

RESTRICTIONS: some functions require WHEEL or OPEPATOR capabilities 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. Note, however, that not all the information that must be provided to the CRDIR call is returned by the GTDIR all. Argument block word 17 (.DCADC), for example, can contain the directory's default account for the CRDIR call, but GTDIR returns zero in that word.

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, GTDIR assumes the length of the argument block is 15 (octal) words long.

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 (.CDSDQ)
online expiration period (.CDDNE)
offline expiration period (.CDDFE)

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.

To obtain all information other than the password of the given directory, the caller must have at least owner access to the directory. (Refer to 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

GTFDB JSYS 63

Returns some or all of the file descriptor block for the specified file. (Refer to 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 (i.e., .FBLEN), but the FDB is an older, small FDB, then the extra words will contain zeroes.

See Section 2.2.8 for the various JSYS's 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

GTHST JSYS 273

Obtains information about ARPANET hosts.

RESTRICTIONS: for ARPANET systems only

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, function-specific data returned in AC's

Code	Symbol	Function
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)
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

Code	Symbol	Function
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 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

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

System Type Flags (HS%STY)

Bits	Symbol	Meaning
1B26	.HS10X	TENEX
2B26	.HSITS	ITS
3B26	.HSDEC	TOPS-10
4B26	.HSTIP	TIP
5B26	.HSMTP	MTIP
6B26	.HSELF	ELF
7B26	.HSANT	ANTS
10B26	.HSMLT	MULTICS
11B26	.HST20	TOPS-20
12B26	.HSUNX	UNIX

GTHST% ERROR MNEMONICS:

ARGX02: Invalid function

GTHSX1: Unknown host number

GTHSX2: No number for that host name

GTHSX3: No string for that host number

GTJIX1: Invalid index

GTJFN JSYS 20

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. (Refer to 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 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

One or more fields of the specification can be defined by a logical name. (Refer to 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.

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.

Bit	Symbol	Meaning
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. (Refer to 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 (e.g., 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.

Bit	Symbol	Meaning
12	GJ%OFG (Cont.)	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.
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.

Bit	Symbol	Meaning	
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 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 (refer to 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.
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.
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
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

IOX11: Quota exceeded

IOX34: Disk full

IOX35: Unable to allocate disk - structure damaged

DESX9: Invalid operation for this device

STRX09: Prior structure mount required

GTJFN JSYS 20

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 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.

Word	Symbol	Meaning
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.
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.

Word	Symbol	Meaning
11	.GJF2 (Cont.)	B6(G1%SLN) Prohibit the expansion of logical names. If, for example, user DBELL defines logical name ME: to be PS:<DBELL> and does a GTJFN for file ME:FOO.BAR, the file specification stored in the JFN block will be: PS:<DBELL>FOO.BAR In this case, the logical name ME: has been expanded to PS:<DBELL>. However, if bit G1%SLN is set, and a GTJFN performed on file FOO.BAR, the file specification stored in the JFN block is: ME:FOO.BAR In this case, the logical name has not been expanded.
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 (i.e., pointer to the CTRL/R buffer). This pointer cannot be equal to the pointer given in AC2. (Refer to 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

Word	Symbol	Meaning
16	.GJATR (Cont.)	The possible keywords and attribute values are as follows:

Keyword	Attribute Value
A:	Installation-defined account string
BDATA:	DECnet binary optional data
BLOCK-LENGTH:	Magnetic-tape block length
BPASSWORD:	DECnet binary password
CHARGE:	DECnet account string
DATA:	DECnet optional data
EXPIRATION-DATE:	Magnetic-tape expiration date
FORMAT:	Magnetic-tape record format. The argument may be one of the following:

Argument	Meaning
F	Fixed-length records
D	Variable-length records
S	Spanned
U	Binary files with 36-bits per word

Keyword	Attribute Value
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
PASSWORD:	DECnet password string
POSITION:	File sequence number to position magnetic-tape to.
RECORD-LENGTH:	Magnetic-tape record length
T	NONE - display-only keyword. The attribute is set by setting bit GJ%TMP in word .GJGEN of the GTJFN block.
USERID:	DECnet user ID string

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. (Refer to 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.
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.

Bit	Symbol	Meaning
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: 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.
16	GJ%FNS	This bit is ignored for the long form of the GTJFN call.
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.
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.
10	GJ%ACT	The account field of the file specification was given.

Bit	Symbol	Meaning
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.
17	GJ%GIV	Invisible files were not considerd when assigning JFNs. This bit is set by the monitor if G1%IIN was not set by the user in the GTJFN call.

Refer to the short form of the GTJFN call for the possible error mnemonics.

GTRPI JSYS 172

Returns the paging trap information for the specified process.

ACCEPTS IN AC1: process handle

RETURNS +1: always, with
AC1 containing number of pager traps (i.e., the number of times a trap has occurred to the pager) for designated process since the process was started
AC2 containing number of page faults (i.e., 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

GTNCP% JSYS 272

Obtains information about the NCP.

RESTRICTIONS:	for ARPANET systems only
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, function-specific data returned in AC's

Code	Symbol	Function
0	.GTNSZ	Returns negative number of NCP connections User-supplied arguments: None Returned data: AC2: -number NCP connections,,0 AC3: -number NVTs ,, line number of first NVT
1	.GTNIX	Returns status of connection number User-supplied arguments: AC2: connection number AC3: 30 bit address of storage block AC4: -length of block ,, index of first data item to return Returned data: (See format of block below)
2	.GTNNI	Return status of NVT line number (input connection) User-supplied arguments: AC2: NVT line number (input) AC3: 30 bit address of storage block AC4: -length of block ,, index of first data item to return Returned data: (See format of block below)

Code	Symbol	
3	.GTNNO	Return status of NVT connection (output connection) User supplied arguments: AC2: NVT line number (output) AC3: 30 bit address of storage block AC4: -length of block ,, index of first data item to return Returned data: (See format of block below)
4	.GTNJF	Return status of network-connection JFN User-supplied arguments: AC2: JFN AC3: 30 bit address of storage block AC4: -length of block ,, index of first data item to return Returned data: (See format of block below)

Format of returned data block:

Word	Symbol	Contents
0	.NCIDX	NCP connection index
1	.NCFHS	Foreign host
2	.NCLSK	Local socket
3	.NCFSK	Foreign socket
4	.NCFSM	State of connection
5	.NCLNK	Link
6	.NCNVT	NVT, or -1 if none
7	.NCSIZ	Byte size of connection
10	.NCMSG	MSG allocation
11	.NCBAL	Bit allocation
12	.NCDAL	Desired allocation
13	.NCBTC	Bits transferred
14	.NCBPB	Bytes per buffer
15	.NCCLK	Time-out countdown
16	.NCSTS	Connection status

GTNCP% ERROR MNEMONICS:

ARGX02: Invalid function

GTJIX1: Invalid index

GTNCX1: Invalid network JFN

GTNCX2: Invalid or inactive NVT

GTRPW JSYS 171

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-B35	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.

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

GTSTS JSYS 24

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 (i.e., 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(GS%FRK)	This is a restricted JFN (GJ%ACC was set in GJTFN 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, line numbers, if 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-B35 (GS%MOD)	data mode of the file. Refer to Chapter 2.

.GSNRM	normal data mode
.GSIMG	image mode
.GSSMB	small buffer mode
.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.

GTTYP JSYS 303

Returns the terminal type number for the specified terminal line. (Refer to Section 2.4.9.4 for the terminal type numbers.)

ACCEPTS IN AC1: file designator (only terminal designators are legal)

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.

GTTYP ERROR MNEMONICS:

DESX1: Invalid source/destination designator

TTYX01: Line is not active

HALTF JSYS 170

Halts the current process and any inferior processes of the current process. Sets the process' PC to the next instruction 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.

Sets bits 1-17(RF%STS) in the status word for this process to 2(.RFVPT). Refer to 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.

HFORK JSYS 162

Halts one or more inferior processes. (Refer to 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). Refer to 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

HPTIM JSYS 501

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)

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. (Refer to the TIME call for obtaining the time in milliseconds.)
1	.HPRNT	CPU runtime for this process. (Refer to the RUNTM call for obtaining the time in milliseconds.)

HPTIM ERROR MNEMONICS:

HPTX1: Undefined clock number

HSYS JSYS 307

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 capabilities enabled

ACCEPTS IN AC1: shutdown time with the date and time in the internal format. (Refer to 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.

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

IDCNV JSYS 223

Converts separate numbers for the local year, month, day, and time into the internal date and time format. (Refer to 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-B17. 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-B17 (IC%TMZ)	time zone to use if B2(IC%UTZ) is on. (Refer to Section 2.9.2 for the time zones.)
B18-B35 (IC%TIM)	local time in seconds since midnight.

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
B1(IC%ADS)	on if daylight savings was applied
B12-B17 (IC%TMZ)	time zone used

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

IDTIM JSYS 221

Inputs the date and time and converts them to the internal date and time format. (Refer to 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). Refer to 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-B3.
B2(IT%SNM)	Interpret the second number in the date as the month (e.g., 6/2/76 is interpreted as Feb. 6, 1976). If this bit is off, the first number is interpreted as the month (e.g., 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-B3 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-B8 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-B10 are both off, a colon is optional between hours and minutes.
B11(IT%AMS)	When B7-B10 are off, always interpret a time specification containing one colon as hhmm:ss.

B12(IT%AHM) When B7-B10 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-B12 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 (e.g., 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.

If AC2 is 0, the IDTIM call accepts any reasonable date and time formats. 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.

IDTNC JSYS 231

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. (Refer to 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-B3. If IT%NDA is off, the date must be input.

B6(IT%NTI) Do not input the time and ignore B7-B16. 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%DAS)	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 day'ight savings time zone was input, or if IT%NTI was set in AC2.
B0(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-B17 (IC%TMZ)	the time zone if one was input, or the local time zone if none was input. (Refer to Section 2.9.2 for the time zones.)
B18-B35 (IC%TIM)	time as seconds since midnight.

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.

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.

IIC JSYS 132

Initiates software interrupts on the specified channels in a process. (Refer to 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

IIC JSYS 132

Initiates software interrupts on the specified channels in a process. (Refer to 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

INLNM JSYS 503

Returns a logical name that is defined either for this job or for the system. (Refer to 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

INLNM ERROR MNEMONICS:

INLNX1: Index is beyond end of logical name table

INLNX2: Invalid function

JFNS JSYS 30

Returns the file specification currently associated with the JFN.

ACCEPTS IN AC1: destination designator where the ASCIZ string is to be written

AC2: indexable file handle (refer to 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 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 (i.e., 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-B20 are divided into 3-bit bytes, each byte representing a field in the file specification. The value of the byte indicates the output for that field. 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-B2(JS%DEV)	output for device field
B3-B5(JS%DIR)	output for directory field
B6-B8(JS%NAM)	output for filename field (2 is illegal)
B9-B11(JS%TYP)	output for file type field (2 is illegal)
B12-B14(JS%GEN)	output for generation number field

B0-B14(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-B17(JS%PRO)	output for protection field
B18-B20(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
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 (i.e., fields whose value is given as 1 or 2), except for first field
B35(JS%PAF)	punctuate all fields from device through ;T

If B32-B35 are 0, punctuation between fields is not used.

If AC3 is 0, the string is output in the format

dev:<directory>name.typ.gen;T

The temporary attribute (;T) is not returned if the JFN is a parse-only JFN (refer to 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

KFORK JSYS 153

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, with 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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

KFRKX1: Illegal to kill top level process

KFRKX2: Illegal to kill self

LGOUT JSYS 3

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 (i.e., a CTRL/C was typed, but no login occurred).

RESTRICTIONS: some functions require WHEEL or OPERATOR capabilities 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 (i.e., 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, and the current date and time. 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

LNMST JSYS 504

Translates a logical name to its original definition string. (Refer to 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.

LNMST ERROR MNEMONICS:

GJFX22: Insufficient system resources (Job Storage Block full)

LNSTX1: No such logical name

LNSTX2: Invalid function

LOGIN JSYS 1

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 processed.

RETURNS +1: failure, error code in AC1

+2: success, date and time of last login (in internal system format; refer to Section 2.9.2) in AC1, and updated string pointers, if pertinent, in AC2 and AC3.

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.

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

LPINI JSYS 547

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 capabilities 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.

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 (e.g., 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

MDDT% JSYS 777

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 capabilities enabled

The MDDT% JSYS accepts no arguments.

MDDT% ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

METER% JSYS 766

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.

RESTRICTIONS: available only on KL10 hardware

ACCEPTS IN AC1: function code

RETURNS +1: always, with 59-bit value in AC2 and AC3

Function Codes:

Code	Symbol	Meaning
1	.MEREA	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 JSYS's 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:

```
!              AC2             !            AC3            !
!==========================================================!
!       High Order Part       !0!  Low Order Part ! Reserved !
!==========================================================!
!0                           35!0!1              23!24      35!
```

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

MRECV JSYS 511

Retrieves an IPCF (Inter-Process Communication Facility) message from the process' input queue. Refer to the TOPS-20 Monitor Calls User's Guide for an overview and description of the Inter-Process Communication Facility.

RESTRICTIONS: Requires 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. (Refer to 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.
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	.IPCFP	Pointer to block where message is to be placed (length of message in the left half and address where message is to be placed in the right half).
4	.IPCFD	User number of sender.
5	.IPCFC	Enabled capabilities of sender.

Word	Symbol	Meaning
6	.IPCSD	Directory number of sender's connected directory.
7	.IPCAS	Account string of sender. The caller supplies a pointer to the block where the account is to be placed.
10	.IPCLL	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 .IPCLL, the monitor will use it to return the ASCIZ string for the logical location of the sender.

Refer to the MSEND call description for the flags that can be set in word .IPCFL 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

IPCF11: WHEEL or IPCF capability required

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

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

IPCF34: Cannot receive into an existing page

MSEND JSYS 510

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. Refer to 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%CFS or 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).
2	.IPCFR	PID of receiver, or 0 if receiver is <SYSTEM>INFO.
3	.IPCFP	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%CFO) 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 (i.e., 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.

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 use

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' 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-B29 (IP%CFE) Error code field for errors encountered by <SYSTEM>INFO during a send or receive request. Code Symbol Meaning

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-B32 (IP%CFC) 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
3	.IPCCP	sent by receiver's <SYSTEM>INFO

B33-B35 (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' 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 (i.e., 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.

Function	Argument	Meaning
.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.
.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 .IPCI1 of the message. This function requires WHEEL or OPERATOR privileges.
.IPCIS		Disassociates all PIDs with names. Used by the monitor on a RESET or LGOUT monitor call. This function is not available to user programs.

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

MSFRK JSYS 312

Starts a process in monitor mode. This call allows job 0 to create multiple processes for handling various asynchronous monitor tasks.

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

MSTR JSYS 555

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 primary structure (PS: on most systems) or any structure declared non-regulated with the .MSSSS function of MSTR.

RESTRICTIONS: some functions require WHEEL, OPERATOR, or MAINTENANCE capability enabled.

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. (Refer to individual function descriptions below.)

Generates an illegal instruction interrupt on all error conditions.

The available functions are summarized below.

Function Code	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.

Function Code	Symbol	Privileged	Meaning
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.
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

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.

The .MSRNU function accepts the channel, controller, and unit numbers in the first three words of the argument block. The first 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 B4(MS%ERR) Unit has an error that was detected during reading

Word	Symbol	Meaning	
3	.MSRST (Cont.)	B5(MS%BBB)	Unit has a bad BAT block. If this bit is on, the data returned in 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
		B9-B17 (MS%TYP)	Type of disk unit 1 .MSRP4 RP04 5 .MSRP5 RP05 6 .MSRP6 RP06 7 .MSRP7 RP07 11 .MSRM3 RM03 24 .MSR20 RP20
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)	
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	

The length of the argument block in words is given by symbol .MSRLN (30).

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:

```
0   2           9 10         17    20          28 29           35
=================================================================
!XX!   CHAR 1    !   CHAR 0    !XX!  CHAR 3    !  CHAR 2       !
-----------------------------------------------------------------
!XX!   CHAR 5    !   CHAR 4    !XX!  CHAR 7    !  CHAR 6       !
-----------------------------------------------------------------
!XX!   CHAR 9    !   CHAR 8    !XX!  CHAR 11   !  CHAR 10      !
=================================================================
```

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

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.

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 and the errors possible on the failure of this function are 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 (usually called PS:) must be brought on line with this function. (The public structure is brought on line during the system startup procedure.)

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:

Word	Symbol	Meaning
0	.MSTNM	Pointer to the ASCIZ string containing the name of the structure.
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:
		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.
		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.
		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 if off, the structure is mounted for general use.
		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.

Word	Symbol	Meaning
3	.MSTUI	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 .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

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.)

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. The caller 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:

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: B0(MS%PS) This structure is a public 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 the public 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.
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 (i.e., 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.

The format of the argument block, the length of which is .MSSLN, is:

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: B1(MS%DIS) Structure is being dismounted. B2(MS%DOM) Structure is domestic. 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

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.

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-B11, function value (MS%FCN) in B12-B17, and number of units in structure (.MSINU) in B18-B35. The flag bits are: 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. The function values that can be given are: 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. 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.

Word	Symbol	Meaning
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 the system. You must supply this word when creating a system structure that does not have the name PS:.

Words 6 through 16 (.MSIST through .MSIOI) 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

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

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 (i.e., 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 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 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

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 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/<.IPCCC>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 (SMOUNT 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.
		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.
.		.
.		.
.		.

Word	Symbol	Meaning
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 enabled WHEEL or OPERATOR program to specify word of homeblock of mounted structure to be modified, which bits should be modified, and what the new values should be.

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

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

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. The process using this JSYS must have 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.

There is no argument block.

The following errors are possible:

MSTRX2: WHEEL or OPERATOR capability required

MTALN JSYS 774

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

MTOPR JSYS 77

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 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 require ARPANET or 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.

ARPANET Functions

ARPANET MTOPR functions are described below. For a complete description of their application, refer to the ARPANET manual.

Code	Symbol	Meaning
20	.MOACP	If a connection is in the RFCR state, use of this function will cause an RFC to be sent to accept the connection.
21	.MOSND	If a connection is operating in buffered send mode, use of this function causes all currently buffered bytes to be sent.
22	.MOSIN	Causes the INS/INR command to be sent.
24	.MOAIN	Assigns interrupt channels through which the program is interrupted on either a change of state (of the connection F.S.M) or receipt of an INS or INR message. The INS/INR PSI channel is stored in field MO%NIN (B0-5) of AC3 and the state change PSI channel is stored in field MO%FSM (B12-17) of AC3. A value of 77 (octal) in either of these fields prevents assignment of a PSI channel.

DECnet Functions

DECnet-20 MTOPR functions are described below. For a complete description of their application, refer to 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:

- connect event pending
- interrupt message available
- 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-B8	MO%CDN	Connect event pending
B9-B17	MO%INA	Interrupt message available
B18-B26	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
.MOCIA	Clear the interrupt
.MONCI	No change

Code	Symbol	Meaning
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	No special error
.DCX1	1	Resource allocation failure
.DCX2	2	Destination node does not exist
.DCX3	3	Node shutting down
.DCX4	4	Destination process does not exist

25 .MORLS (Cont.)

Symbol	Value	Meaning
.DCX5	5	Invalid name field Destination process queue overflow
.DCX7	7	Unspecified error
.DCX8	8	Third party aborted link
.DCX9	9	User abort (asynchronous disconnect)
.DCX11	11	Undefined error code
.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
.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	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.

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.

Code	Symbol	Meaning
30	.MORUS	Return the source task user identification supplied in the connect initiate message. This 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 string; a zero value indicates that no optional data was supplied.

Code	Symbol	Meaning
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 (eight bytes maximum) and that AC4 contain a count of the bytes in the interrupt message (sixteen bytes maximum).
37	.MOROD	Return the unique identification of the source task. This identification is in the format of object-descriptor, and the contents depend on the DECnet implementation on the remote host. In addition, if the source task is running on a system that provides for group and user codes, this information is also returned. If the source task name is on a DECnet-20 host, the data returned is TASK-taskname. This function requires that AC3 contain a string pointer to the location where the object-descriptor of the source task 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 object-descriptor stored as specified. In addition, if the source host system uses group and user codes (sometimes referred to as project and programmer number or p,pn), AC4 contains the group code in the left half and the user code in the right half. If the source host system does not provide for group or user codes, AC4 contains zeros.

Code	Symbol	Meaning
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. This function requires that AC2 contain a reject code in the left half and .MOCLZ in the right half. The reject code is a 2-byte, NSP-defined decimal number indicating the reason that a target task is rejecting a connection. Refer to the description of code 25, .MORLS, for a list of disconnect/reject codes. AC3 contain a string pointer to any data to be returned. (If the byte size exceeds eight bits, bytes are truncated to eight bits.) AC4 contain the count of bytes in the data string (maximum=16). A zero indicates no data.
41	.MOCC	Accept a connection either implicitly or 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 target task attempts to output to the logical link (issues a SOUT or SOUTR monitor call to the network). The target task submits a read request to the logical link (issues a SIN or SINR monitor call to the network). The target task is in input wait state (has enabled itself for a "data available" software interrupt). 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.

Code	Symbol	Meaning
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. The monitor call returns the maximum segment size, in bytes, in AC3.
43	.MOANT	Attach network terminal. This function passes a DECnet logical link from the DECnet backround job (MCBNRT) to TOPS-20 so that TOPS-20 can control terminal I/O to and from the DECnet logical link. The MCBNRT program must establish the logical link and exchange the necessary DECnet protocols before this function of the MTOPR call is executed. The JFN accepted by this function in AC1 is the JFN of the DECnet logical link. This call returns the line number of the DECnet logical link in AC2. The TOPS-20 job is associated with the DECnet logical link until one of the following occurs: 1. The logical link is broken by the foreign host or by DECnet. 2. The job logs out, more data comes through the logical link, and the first character of that data is not a CTRL/C. If the first character is a CTRL/C, a new job is created using the same logical link.
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

Front-End Functions

Code	Symbol	Meaning
3	.MOEOF	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 non-zero, only the buffers are flushed. This function is used for synchronization between a program running on TOPS-20 and a program running on the front end.
4	.MODTE	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. Unless otherwise noted, the JFN must be opened before the MTOPR function can be performed.

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 (refer to 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	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. Use function .MORVL to rewind the current volume.
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.

Code	Symbol	Meaning
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).

Code	Symbol	Meaning
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.

Code	Symbol	Meaning
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.

For labeled tapes, this function is allowed only if the file has been opened in dump mode. If the file has not been opened in dump mode, an MTOX1 error is returned.

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.

Code	Symbol	Meaning
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 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 (i.e., erase gap). This function requires that the magnetic tape be opened for write access. This function is illegal for labeled tapes.

Code	Symbol	Meaning
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. 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 tape, 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.

Code	Symbol	Meaning
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.
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

The JFN need not be open for this function.

Code	Symbol	Meaning
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.

Code	Symbol	Meaning
30	.MOIEL	Inhibit error logging for the tape. If AC3 is non-zero, 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. 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.

Code	Symbol	Meaning
37	.MOSTA	Return current magtape status. Returns an argument block having the following form and contents:

Word	Symbol	Contents
0	.MOCNT	Count of words in the block including this word
1	.MODDN	density flags

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

Word	Symbol	Contents
2	.MODDM	data mode flags

Bit	Symbol	Meaning
B1	SJ%CMC	core dump
B2	SJ%CM6	SIXBIT
B3	SJ%CMA	ANSI ASCII
B4	SJ%CM8	industry compatible
B5	SJ%CMH	high density mode

Word	Symbol	Contents
3	.MOTRK	recording track flags

Bit	Symbol	Meaning
B1	SJ%7TR	7-track drive
B2	SJ%9TR	9-track drive

Word	Symbol	Contents
4	.MOCST	tape status flags

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

Word	Symbol	Contents
5	.MODVT	device type

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.

This function rquires OPERATOR or WHEEL capability. The JFN need not be open for this function.

Code	Symbol	Meaning
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 EOV 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	Symbol	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)

44 .MOVLS (Cont.)

Word	Symbol	Function
4	.VSMRV (Cont.)	For .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 non-zero, the translate flag is set.

This function is for labeled tapes only.

Code	Symbol	Meaning
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 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 EOV has been received, the user program may write a UTL label into the EOV 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 EOV PSI from within the code writing the UTL labels for the file. It is not possible to receive an EOV PSI while writing the trailer labels in the EOV set.

Code	Symbol	Meaning
		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

Value	Symbol	Label Type
1	.LTUNL	Unlabeled
2	.LTANS	ANSI
3	.LTEBC	EBCDIC
4	.LTT20	TOPS-20

Word	Contents
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
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.

Code	Symbol	Meaning
50	.MORLI (Cont.)	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.

Code	Symbol	Meaning
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.

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	Enable for a software interrupt on nonfatal device conditions. Examples of these conditions are:

1. Device changed from offline to online.

2. Device changed from online to offline.

3. Device's page counter has overflowed.

Other device errors or software conditions are not handled by this function; instead they cause a software interrupt on channel 11 (.ICDAE).

Code	Symbol	Meaning
27	.MOPSI (Cont.)	Argument Block E: 3 E+1: interrupt channel number E+2: flags. The following flag is defined: B0(MO%MSG) Suppress standard CTY device messages.
31	.MONOP	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.
32	.MOLVF	Load the line printer's VFU (Vertical Formatting Unit) from the file indicated in the argument block. Argument Block E: 2 E+1: JFN of the file containing the VFU 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 E: 3 E+1: pointer to destination area for ASCIZ name string E+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 E: 2 E+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 E: 3 E+1: pointer to destination area for ASCIZ name string E+2: number of bytes in destination area

Code	Symbol	Meaning
36	.MOSTS	Set the status of the line printer.

Argument Block

E: 3
E+1: software status word, with the following status bits settable by the caller:

Code	Symbol	Meaning
36	.MOSTS	

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.

E+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.

Code	Symbol	Meaning
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

E: 3
E+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.

Code	Symbol	Meaning	
37	.MORST (Cont.)	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 (e.g., 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.
		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.	
		E+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: 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). Argument Block E: 3 E+1: interrupt channel number E+2: flags. The following flag is defined: B0(MO%MSG) Suppress standard CTY device messages.
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 E: 2 E+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. 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 E: 3 E+1: interrupt channel number E+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 E: 2 E+1: status word. Bits 2-17 contain the software status word from the front end, and bits 20-35 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 (e.g., interrupt character) occurred.

Code	Symbol	Meaning	
37	.MORST (Cont.)	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. (Refer to 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-B17(MO%SIC) software interrupt channel number for output to the PTY. The channel number used for input from the PTY is one greater than the channel number used for output to the PTY. B18-B35 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.
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, the 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.
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.

Code	Symbol	Meaning
36	.MOSIG	Specify if input on this terminal line is to be ignored when the line is inactive (i.e., is not assigned or opened). This function requires AC3 to contain 0 if characters on this line are 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.
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.

Code	Symbol	Meaning
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 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 elft half, and the unpause character in the right half.

Generates an illegal instruction interrupt on error conditions below.

MTOPR ERROR MNEMONICS:

ANTX01: No more network terminals available

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
TTYX01:	Line is not active

MTU% JSYS 600

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 enabled WHEEL or OPERATOR capabilities

ACCEPTS IN AC1: function code

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	.MTNNV	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

Code	Symbol	Function
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.

Code	Symbol	Function
3	.MTASI	return assignment information Argument Block: Word / Symbol / Contents 0 .MTCNT count of words in block 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 capabilities 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

MUTIL JSYS 512

Performs various IPCF (Inter-Process Communication Facility) functions, such as enabling and disabling PIDs, assigning PIDs, and setting quotas. Refer to 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:

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	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. Argument PID
2	.MUDIS	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. Argument PID

Code	Symbol	Meaning
3	.MUGTI	Return the PID associated with <SYSTEM>INFO. The PID is returned in word 2 of the argument block. Argument PID or job number
4	.MUCPI	Create a private copy of <SYSTEM>INFO for the specified job. The caller must have IPCF capability enabled. Arguments PID to be assigned to <SYSTEM>INFO PID or number of job creating private copy
5	.MUDES	Delete the specified PID. The caller must own the PID being deleted. 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-B26, and the new receive quota is given in B27-B35. 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

Code	Symbol	Meaning
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
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-B26 and the receive quota in B27-B35. 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

Code	Symbol	Meaning
20	.MUQRY	Return the Packet Descriptor Block for the next packet in the queue associated with the specified 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

Code	Symbol	Meaning
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 PID 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 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 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

NIN JSYS 225

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.

ACCEPTS IN AC1: source designator

AC3: radix (2-10) 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 10

IFIXX2: First nonspace character is not a digit

IFIXX3: Overflow (number is greater than 2**35)

DESX1: Invalid source/destination designator

DESX2: Terminal is not available to this job

DESX3: JFN is not assigned

DESX5: File is not open

NODE JSYS 567

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.

RESTRICTIONS: Some functions require WHEEL, OPERATOR, or MAINTENANCE capability, or TOPS-20, Version 5.1.

ACCEPTS IN AC1: function code

AC2: address of argument block

RETURNS +1: always. If an error occurs, an illegal instruction trap is generated.

The available functions and their argument blocks are described below.

Code Symbol Function

0 .NDSLN Set local node name

Requires WHEEL or OPERATOR capabilities.

Argument Block:

Word	Symbol	Contents
0	.NDNOD	Byte pointer to ASCIZ node name.

1 .NDGLN Get local node name

Argument Block:

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 capabilities.

Argument Block:

Word	Symbol	Contents
0	.NDNOD	Number to set (Phase II: 2 < n < 127; Phase III: from 1 to .NDMAX)

3	.NDGNM	Get local node number Argument Block: Word / Symbol / Contents 0 .NDNOD Returned node number
4	.NDSLP	Set loopback port (2020 only) Requires WHEEL, OPERATOR or MAINTENANCE capabilities. Argument Block: Word / Symbol / Contents 0 .NDPRT NSP port number. The .BTCLI function of the BOOT monitor call converts a line number to an NSP port number.
5	.NDCLP	Clear loopback port (2020 only) Requires WHEEL, OPERATOR, or MAINTENANCE capabilities. Argument Block: Word / Symbol / Contents 0 .NDPRT NSP port number.
6	.NDFLP	Find loopback port (2020 only) Argument Block: Word / Symbol / Contents 0 .NDPRT NSP port number 1B0(ND%LPR) Loopback running 1B1(ND%LPA) Loopback port assigned
7	.NDSNT	Set network topology. Sets the system's table of reachable nodes. Requires WHEEL or OPERATOR capabilities. Argument Block for monitors prior to TOPS-20, Version 5.1: Word / Symbol / Contents 0 .NDNND Number of following words in right half. Left half is reserved. 1 .NDCNT Number of words in a node block 2 .NDBK1 Addresses of N node blocks (one for each node for which updated information is to be conveyed to the monitor).

7 .NDSNT (Cont.)

Node Block:

Word	Symbol	Contents
0	.NDNAM	Byte pointer to ASCIZ node name
1	.NDSTA	Node state: .NDSON On Add to table of reachable nodes if not already there. .NDSOF Off Remove from table if previously there.
2	.NDNXT	Byte pointer to the DN20 name.

Argument Block for TOPS-20, Version 5.1:

Word	Symbol	Contents
0	.NDNNO	Number of nodes reported in topology message.
1	.NDMSK	Address of topology message.

The topology message is made up of 8-bit bytes. The bytes are left-justified within the topology message word. Each byte contains 4 2-bit fields.

A two-bit field giving the topology status for a node has the following format:

00	Node not reachable
01	(reserved)
10	Reachable Phase II node
11	Reachable Phase III node

The bit fields are packed four to a byte (see below), low-order to high-order. The first byte represents nodes 4, 3, 2, 1; the second byte represents nodes 10, 7, 6, 5; and so on.

```
   4   3   2   1   10   7   6   5   14  13  12  11  20  17  16  15
-----------------------------------------------------------------
!   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !
!   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !
-----------------------------------------------------------------
```

10 .NDGNT Get network topology.

Reads the system's table of reachable nodes.

10 .NDGNT (Cont.)

Argument Block:

Word	Symbol	Contents
0	.NDNND	Number of following words 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	Off

Word	Symbol	Contents
2	.NDNXT	Obsolete (always 0)
3-4	--	ASCIZ node name (if node name .LE. 4 characters, Word 4 NOT returned)

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.

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

14 .NDGLI Get line information

Returns information on lines known to NSP.

Argument Block:

Word	Symbol	Contents
0	.NDNLN	Number of following words in right half (set by user on call) and number of lines (N) for which information was returned in the left half (set by monitor on return).
1	.NDBK1	Addresses of N blocks of information for each line for which the monitor will return data to the user. The format of these blocks is described below.
	.NDBK1+N	Start of an area into which the monitor will sequentially place line blocks (described below). If there is not enough space to hold all of the information, the NODE JSYS will store as much as possible and then fail with error code ARGX04.

14	.NDGLI (Cont.)	Line Block:

Word	Symbol	Contents
0	.NDLNM	line number
1	.NDLST	State of Line
		.NDLON On
		.NDLOF Off
		.NDLCN Controller loopback
		.NDLCB Cable loopback
2	.NDLND	Byte pointer to ASCIZ name of node at the end of the line.
3	.NDLSZ	Size of node block.

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:

ND%EXM	The specified node exactly matches a node name in the monitor's node database.

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.

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
NODX03:	Another line already looped
NSPX25:	Illegal DECnet node number
NSPX26:	Table of topology watchers is full

NOUT JSYS 224

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 (e.g., 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 this bit is not set and bit 4 (NO%OOV) is set, all necessary digits are output on column overflow.
B11-B17 (NO%COL)	number of columns (including sign column) to output. If this field is 0, as many columns as necessary are output.
B18-B35 (NO%RDX)	radix (2-36) of number being output

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

NTMAN% JSYS 604

Provides an interface between the DECnet-20 Network Management layer and lower layers of the Digital Network Architecture.

RESTRICTIONS: WHEEL or OPERATOR privileges are required.

ACCEPTS IN AC1: address of argument block

RETURNS: +1 always. If an error occurs, generates an illegal instruction trap, with error code returned in AC1.

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

Word	Symbol	Contents
2	.NTEID	Byte pointer to Entity ID. (See the Network Management Specification for format.)
3	.NTFNC	Function to be performed

Code	Symbol	Meaning
-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

Word	Symbol	Contents
4	.NTSEL	Selection criterion for function

Selectors for Show Selected Items (.NTSHO)

Code	Symbol	Meaning
0	.NTSUM	Summary
1	.NTSTA	Status
2	.NTCHA	Characteristics
3	.NTCOU	Counters
4	.NTEVT	Event

4	.NTSEL (Cont.)	Selectors for Return List of Items (.NTRET)

Code	Symbol	Meaning
-1	.NTKNO	Known Items
-2	.NTACT	Active Items
-3	.NTLOP	Loop

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

ODCNV   JSYS 222

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. (Refer to Section 2.9.2 for more information.) The ODCNV call gives the caller the 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

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-B17(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-B17 (IC%TMZ)	time zone to use if B2(IC%UTZ) is on.

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-B17 (IC%TMZ)	time zone used
B18-B35 (IC%TIM)	local time in seconds since midnight

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.

Generates an illegal instruction interrupt on error conditions below.

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

ODTIM JSYS 220

Outputs the date and time by converting the internal format of the date and/or time to text. (Refer to 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-B8.

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 (e.g., 6 Feb 76). If B6(OT%DAM) is also on, a comma is output after the day of the month (e.g., Feb 6, 76).

B8(OT%SLA) Output the date with slashes (e.g., 2/6/76).

If B7-B8 are both off, the date is output with dashes between the items (e.g., 6-Feb-76).

B9(OT%NTM) Do not output the time and ignore B10-B13.

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 (e.g., -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.

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-B2,B4-B7, and B17 were on (i.e., 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

Generates an illegal instruction interrupt on error conditions below.

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.

ODTNC JSYS 230

Outputs the date and/or the time as separate numbers for local year, month, day, or time. (Refer to 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-B17(IC%TMZ) time zone in which to output

B18-B35(IC%TIM) local time in seconds since midnight

AC5: format option flags (refer to 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.

Generates an illegal instruction interrupt on error conditions below.

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.

OPENF JSYS 21

Opens the given file. Refer to 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-B5(OF%BSZ) Byte size (maximum of 36 decimal). If a zero byte size is supplied, the byte size defaults to 36 bits.

B6-B9(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

(See Section 2.5 for more information on software data modes.)

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(OF%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 this 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(OF%RD) Allow read access.

B20(OF%WR) Allow write access.

B21(OF%EX) Allow execute access.

B22(OF%APP) Allow append access.

B23(OF%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 OF%THW or OF%WR bits.
B25(OF%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 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 controlling 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 may be opened with this file until the current JFN is released.

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 new files.

Ordinarily, TOPS-20 automatically updates modified memory pages to disk approximately once each minute. OF%DUD prohibits this automatic update. However, there are three sources of "manual" updating that are not controlled by OF%DUD:

1. A CLOSF is performed
2. A UFPGS is performed
3. Swapping space becomes full

OF%DUD and OF%THW interact in the following ways:

OF%THW	OF%DUD	Effect
0	-	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. This means that if the process has access to read the file and it sets only the write access bit, the process will have the file opened for read, write, and execute access. However, if an existing file is opened and only write access is specified (only OF%WR is set), the 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:

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 alway 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
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
TTYX01:	Line is not active

PBIN JSYS 73

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. (Refer to 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

PBOUT JSYS 74

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. (Refer to bit OF%HER of the OPENF call description.)

PBOUT 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

PDVOP% JSYS 603

Manipulates program data vectors (PDV's), which begin at program data vector addresses (PDVA's). 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.

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 ASCIZ representation. (In most cases, a byte pointer should be created to access this string.)
2	.PVSTR	Program starting address.
3	.PVREE	Program reenter address.
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 G, for a description of this block.
6	.PVSYM	Address of the program symbol table.
7	.PVCTM	Time at which the program was compiled.

Word	Symbol	Meaning
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.

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. 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:

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

PEEK JSYS 311

Transfers a block of words from the monitor to the user space. The desired monitor pages must 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

PLOCK JSYS 561

Acquires physical memory and places a designated section of the process' 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.

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. (Refer to the PMAP call.) This will unlock the process' 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.

Generates an illegal instruction interrupt on error conditions below.

PLOCK ERROR MNEMONICS:

ARGX22: Invalid flag

ARGX24 invalid count

PMAP JSYS 56

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' 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:

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-B4.
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 a process page number. If the section containing the page does not exist, a private section is created.
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-B35 (PM%RPT) Number of pages to be mapped if 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 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 nonexistant 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' 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' 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.

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.

(Refer to Section 2.2.8 for the format of the FDB fields.)

Case III Mapping One Process' 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 nonexistant 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:	B0(PM%CNT)	Repeat count. Only the process page numbers are incremented.
		B18-B35	Number of pages to remove from process

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' 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

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.

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

PMCTL JSYS 560

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 or OPERATOR capability enabled

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 .MCPPN, and its status is returned in word .MCPST.
		Argument Block
		0 .MCPPN 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. Nonexistent memory is marked as off line at system startup.
		3 .MCPSE Page is off line because the monitor detected an error.

Function	Symbol	Meaning
3	.MCSPS	Set the status of the given page. The number of the page is given in word .MCPPN, and the status value is given in word .MCPST.

Argument Block

0	.MCPPN	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.
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
6. any page belonging to the resident free space pool

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

PPNST JSYS 557

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

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

PRARG JSYS 545

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 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

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 ASCIZ 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

PSOUT JSYS 76

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. (Refer to 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

RCDIR JSYS 553

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)

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 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.

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

RCM JSYS 134

Returns the word mask of the activated interrupt channels for the specified process. (Refer to 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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

RCUSR JSYS 554

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.

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, refer to 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, refer to 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.

RCUSR ERROR MNEMONICS:

RCUSX1: Insufficient system resources

RCDIX4: Monitor internal error

STRX07: Invalid user number

STRX08: Invalid user name

RCVIM JSYS 751

Retrieves a message from the ARPANET special message queue. The queue must have been previously assigned with the ASNSQ JSYS.

RESTRICTIONS: for ARPANET systems only.

ACCEPTS IN AC1:

Bit0:	If set, the user will receive a 96-bit leader. If reset, the user will receive a 32-bit leader.
Bit1:	If set, the user will receive data in the high-order 32 bits of each word of the message. If reset, the user will receive data in all 36 bits of each word of the message.
Bits 18-35:	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 specified 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

RCVOK% JSYS 575

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)
10	.RCRJB	Requested job number
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

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

RDTTY JSYS 523

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).

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 (e.g., 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 (i.e., do not print XXX, and on display terminals, do not delete the characters from the screen).

B18-B35 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.

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

RELD JSYS 71

Releases one or all devices assigned to the job. When a device is released by the job, the resource allocator receives an IPCF packet. (Refer to 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

RELSQ JSYS 753

Deassigns the ARPANET special message queue. (The LGOUT JSYS deassigns all special message queues.) All pending messages relative to the specified queue(s) are discarded.

RESTRICTIONS: for ARPANET 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.

RESET JSYS 147

Resets and initializes the current process. It is a good programming practice to include this call at the beginning of each assembly language program.

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 (i.e., has never been closed), it is closed and then expunged.

2. Kills all inferior processes.

3. Clears the current process' 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 (refer to Section 2.4.9.1):

TT%WAK(B18-B23)	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' PIDs.

6. Dequeues all of the current process' ENQ requests.

7. Clears the compatibility package's entry vector.

8. Releases all process handles that can be released. (Refer to the RFRKH call description.)

RFACS JSYS 161

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:

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

RFBSZ JSYS 45

Returns the byte size for a specific opening of a file. (Refer to 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

RFCOC JSYS 112

Returns the control character output control (CCOC) words for the specified terminal. (Refer to 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).

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

RFMOD JSYS 107

Returns the JFN mode word associated with the specified file. (Refer to 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

RFORK JSYS 155

Resumes one or more processes that had been directly frozen. This monitor call does not resume a process that has been indirectly frozen. (Refer to 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. (Refer to 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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

RFPOS JSYS 111

Returns the current position of the specified terminal's pointer. (Refer to Section 2.4.9.1 for information on page lengths and widths of terminals.)

ACCEPTS IN AC1: device designator

RETURNS +1: always, with AC2 contains position within a page (i.e., line number) in the left half, and position within a line (i.e., 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.

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

RFPTR JSYS 43

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

RFRKH JSYS 165

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 (i.e., 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

The process handles released when AC1 is -1 are the ones released on a RESET or a KFORK monitor call.

RFRKH ERROR MNEMONICS:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

RFSTS JSYS 156

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-B17 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-B17(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.
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).

Value	Symbol	Meaning
7	.RFABK	The process is dismissed because it encountered an instruction on which an address break was set (by means of the ADBRK call).

B18-B35(RF%SIC) The number of the software interrupt channel that caused the forced process termination.

The RFSTS call returns with -1 (fullword) in AC3 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

Flags:

B0	RF%LNG	Long form call (must be on)
B1-B17		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

RFTAD JSYS 533

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.) Requires WHEEL or OPERATOR capability enabled.
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 (i.e., the argument block contains more than 4 words).

The following table illustrates which JSYS's 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.

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

RIN JSYS 54

Inputs a byte nonsequentially (i.e., 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.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to 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

RIR JSYS 144

Returns the channel and priority level table addresses for the specified process. (Refer to 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

RIRCM JSYS 143

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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

RLJFN JSYS 23

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

RJFNX1: File is not closed

RJFNX2: JFN is being used to accumulate filename

RJFNX3: JFN is not accessible by this process

OPNX1: File is already open

RMAP JSYS 61

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

RNAMF JSYS 35

Renames an existing file. The JFNs of both the existing file and the new file specification must be closed.

ACCEPTS IN AC1: JFN of existing file to be renamed (i.e., source file)

AC2: JFN of new file specification (i.e., 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 (refer to 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.

RNAMF 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

RNAMX1: Files are not on same device

RNAMX2: Destination file expunged

RNAMX3:	Write or owner access to destination file required
RNAMX4:	Quota exceeded in destination of rename
RNAMX5:	Destination file is not closed
RNAMX6:	Destination file has bad page table
RNAMX7:	Source file expunged
RNAMX8:	Write or owner access to source file required
RNAMX9:	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

ROUT JSYS 55

Outputs a byte nonsequentially (i.e., 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. (Refer to 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

IOX11: Quota exceeded

IOX34: Disk full

IOX35: Unable to allocate disk - structure damaged

RPACS JSYS 57

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 half 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.

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

RPCAP JSYS 150

Returns the capabilities for the specified process. (Refer to 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

RSCAN JSYS 500

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.

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 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

RSMAP% JSYS 610

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 mapping to another fork's section. If n is a JFN, the mapping is a shared mapping to a file section. These are called section handles.

The access information 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
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

RTFRK JSYS 322

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.

RTFRK ERROR MNEMONICS:

FRKHX6: All relative process handles in use

RTIW JSYS 173

Reads the terminal interrupt word (refer to 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-B35 (RT%PRH) process handle, or -5 for entire job

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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

RUNTM JSYS 15

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

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

RWM JSYS 135

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

RWSET JSYS 176

Releases the working set by removing all of the current process' 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

SACTF JSYS 62

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

VACCX0: Invalid account

VACCX1: Account string exceeds 39 characters

VACCX2: Account has expired

SAVE JSYS 202

Saves, in nonsharable format, pages of a process in the specified file. The process must run in one section of memory. (Refer to 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

FRKHX8: 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.

SCTTY JSYS 324

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 (refer to 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:

Code	Symbol	Meaning
0	.SCRET	Return the designator of the given process' controlling terminal. The designator is returned in AC2.
1	.SCSET	Change the given process' 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' 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 (refer to 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

SCVEC JSYS 301

Sets the entry vector and the UUO locations for the compatibility package.

ACCEPTS IN AC1: process handle

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.

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

SDSTS JSYS 146

Sets the status of a device. (Refer to 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

SDVEC JSYS 543

Sets the entry vector for the Record Management System (RMS).

RESTRICTIONS: requires RMS software (currently available only with BASIC and COBOL)

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.

Generates an illegal instruction interrupt on error conditions below.

SDVEC ERROR MNEMONICS:

ILINS5: RMS facility is not available

FRKHX8: Illegal to manipulate an execute-only process

SETER JSYS 336

Sets the most recent error condition encountered by a process. This error condition is stored in the process' 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.

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

SETJB JSYS 541

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.
	.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

Function	Values	Meaning
.SJRS(3) (Cont.)		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
.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

SETNM JSYS 210

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.

SETSN JSYS 506

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. (Refer to 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)".

SEVEC JSYS 204

Sets the entry vector of the specified process. The process must run in only one section of memory. (Refer to Section 2.3.2.)

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' 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

SFACS JSYS 160

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.

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

SFBSZ JSYS 46

Resets the byte size for a specific opening of a file. (Refer to 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

SFCOC JSYS 113

Sets the control character output control (CCOC) for the specified terminal. (Refer to Section 2.4.9.2 and the RFCOC call description.)

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.

The RFCOC monitor call can be used to obtain the CCOC words for a specified terminal.

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

SFMOD JSYS 110

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. (Refer to 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

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

SFORK JSYS 157

Starts the specified process in section zero. 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: the 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.

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

SFPOS JSYS 526

Sets the position of the specified terminal's pointer. (Refer to 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

SFPTR JSYS 27

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

AC2: byte number to which the pointer is to be set, or -1 to set the pointer to the current end of the file

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

SFRKV JSYS 201

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 that 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

SFTAD JSYS 534

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.

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:

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

SFUST JSYS 551

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

SIBE JSYS 102

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

SIN JSYS 52

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.

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. (Refer to Section 2.4.7.1 for more information about magnetic tape I/O.)

This call can cause several software interrupts or process terminations on certain file conditions. (Refer to 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

SINR JSYS 531

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

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

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
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

SIR JSYS 125

Sets the addresses of the channel and priority level tables for the specified process. (Refer to Section 2.6.3.) The process must run in one section of memory. 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

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

SIRCM JSYS 142

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 (e.g., 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

SIZEF JSYS 36

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 (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

SJPRI JSYS 245

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 DEC. Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: job number

AC2: priority word

RETURNS +1: always

The priority word has the following format:

```
0                            17 18       24      29 30       35
===============================================================
!              PERC            ! !       !  HIGH  !   LOW    !
===============================================================
```

Where:

PERC is the percentage of CPU resources to be guaranteed for the job. This value may be in the range 0<= n <=99.

B18 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.

LOW is the lowest priority queue the job may run in. This queue is always specified as the desired queue + 1. For example, queue 2 is specified as 3.

HIGH is the highest priority queue the job may run in

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.

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.

Generates an illegal instruction interrupt on error conditions below.

SJPRI ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

SJPRX1: Job is not logged in

SKED% JSYS 577

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

The available functions are:

Code	Symbol	Function
1	.SKRBC	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.

Argument block:

Word	Symbol	Contents
0	.SACNT	Count of words in argument block (Including this word)
1	.SAKNB	Bias control knob setting

Code	Symbol	Function
2	.SKSBC	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<= n <=20. Requires WHEEL or OPERATOR capabilities enabled.

Argument block:

Word	Symbol	Contents
0	.SACNT	Count of words in argument block (Including this word)
1	.SAKNB	Bias control knob setting

Code	Symbol	Function
3	.SKRCS	Read class parameters. Returns the following values:

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<= n <=1.
3. Amount of processor actually used by the class. The amount used is returned as a floating-point value n, such that 0<= n <=1.

Code	Symbol	Function
3	.SKRCS (Cont.)	

4. 1 minute load average. The load average = (J/P) where J is the number of CPU-runnable jobs in 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

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.

Code	Symbol	Function
5	.SKICS (Cont.)	

Word	Symbol	Contents
0	.SACNT	Count of words in argument block (Including this word)
1	.SACTL	Control flags

The flags are as follows:

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 capabilities 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:

Word	Symbol	Contents
0	.SACNT	Count of words in argument block (Including this word)
1	.SAJOB	Job number
2	.SAJCL	Class of job
3	.SAWA	Windfall allocation

7 .SKRJP Read class parameters for a job

Returns the following values:

1. Job's share of the processor. This value is returned as a floating-point value n, such that 0<= n <=1.

2. Job's use of the processor. This value is returned as a floating-point value n, such that 0<= n <=1.

Code	Symbol	Function
7	.SKRJP (Cont.)	Argument block:

Word	Symbol	Contents
0	.SACNT	Count of words in argument block (including this word)
1	.SAJSH	Job's share allotment
2	.SAJUS	Job's current use

Code	Symbol	Function
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

Code	Symbol	Function
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

Code	Symbol	Function
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 non-zero (set). A non-zero indicates that batch jobs should be run in the "dregs" queue.

Argument block:

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 non-zero = run in dregs queue

Code	Symbol	Function
13	.SKDDC	Reserved
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:

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

SKPIR JSYS 127

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

SMAP% JSYS 767

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

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. 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.
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 non-zero section before 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 nonexistant 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 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.

ARGX23:	Invalid section number
ARGX24:	Invalid count
SMAPX1:	Attempt to delete a section still shared
SMAPX2:	Indirect section map loop detected

SMON JSYS 6

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 ARPANET 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.
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	ARPANET terminal LOGINs 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
44	.SFNTN	Turn ARPANET on.
45	.SFNDU	Reinitialize ARPANET if it is down.
46	.SFNHI	Initialize ARPANET host table.
47	.SFTMZ	Set the local time zone to the value given in AC2.
50	.SFLHN	Set the local ARPANET 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

Code	Symbol	Meaning
53	.SFSOK (Cont.)	Flags: Bit Symbol Meaning B0 SF%ECK 0 = Disable access checking 1 = Enable access checking 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) can be enabled/disabled by using function code 400000. 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

The TMON monitor call can be used to obtain the settings of the various monitor flags.

Generates an illegal instruction interrupt on error conditions below.

SMON ERROR MNEMONICS:

SMONX1: WHEEL or OPERATOR capability required

SMONX2: Invalid SMON function

SNDIM JSYS 750

Places a message in a previously assigned ARPANET special message queue.

RESTRICTIONS: for ARPANET systems only.

ACCEPTS IN AC1:

AC1:	Bit0:	If set, the message contains a 96-bit leader. If reset, the message contains a 32-bit leader.
	Bit1:	If set, the data resides in the high-order 32 bits of each word of the message. If reset, the data resides in all 36 bits of each word of the message.
	Bits 18-35:	Special Queue Header
AC2:	address of an extended message	

RETURNS +1: failure, error code in AC1

+2: success, message queued

See the ARPANET manual for the format of the message.

The RCVIM JSYS can be used to retrieve a message from the special message queue.

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

SNOOP JSYS 516

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.

RESTRICTIONS:		requires WHEEL or OPERATOR 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:

Function 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: instuction to be executed before the patched instruction is executed. The instruction can be: 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.

Function Code	Symbol	Meaning
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. AC2: 36-bit value of symbol that is to be looked up in the monitor's symbol table. 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

SOBE JSYS 103

Tests to see if the designated file output buffer is empty.

ACCEPTS IN AC1: destination designator

RETURNS +1: output buffer is not empty. Number of bytes remaining in output buffer is returned in AC2.

+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.

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

SOBF JSYS 175

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).

SOUT JSYS 53

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

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.

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. Refer to Section 2.4.7.1 for more information about magnetic tape I/O.

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

Generates an illegal instruction interrupt on error conditions below.

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

SOUTR JSYS 532

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 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.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to 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

SPACS JSYS 60

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

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 (i.e., 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

SPJFN JSYS 207

Sets the primary JFNs (.PRIIN and .PRIOU) for the specified process.

ACCEPTS IN AC1: process handle

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' 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

SPLFK JSYS 314

Splices a process structure. The process that becomes the new superior must be either the one executing the SPLFK monitor call, or an inferior of it. The new inferior process must be an inferior of the executing process. The new superior process must not be the same process as the new inferior process, and must not be inferior to the new inferior process. The new inferior process and all of its inferiors will be frozen after execution of the SPLFK call.

ACCEPTS IN AC1: process handle of the new superior process

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).

SPLFK ERROR MNEMONICS:

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

FRKHX1: Invalid process handle

SPOOL JSYS 517

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 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.

Code	Symbol	Meaning
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 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

<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

SPRIW JSYS 243

Sets the priority word for the specified process.

RESTRICTIONS: requires WHEEL or OPERATOR capability enabled

ACCEPTS IN AC1: process handle

AC2: priority word

RETURNS +1: always

Refer to 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

SSAVE JSYS 203

Creates a sharable, save-format file for the given JFN by copying (not sharing) pages from the given process. (Refer to 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. 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 AC1) 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 non-zero 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:

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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

SSAVX1: Illegal to save files on this device

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.

STAD JSYS 226

Sets the system's date. (Refer to Section 2.9.2.)

RESTRICTIONS: Some functions require 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

STCMP JSYS 540

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.

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.

STDEV JSYS 120

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

+2: success, device designator (refer to 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

STI JSYS 114

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 (e.g., 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. (Refer to 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.

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

STIW JSYS 174

Sets the terminal interrupt word (refer to 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 (i.e., 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.

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 disable the code for CTRL/C interrupts or to give -5 as an argument.

ACCEPTS IN AC1: B0(ST%DIM) set the deferred terminal interrupt mask given in AC3

B18-B35 (ST%PRH) process handle, or -5 for entire job

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

STO JSYS 246

Simulates terminal output.

ACCEPTS IN AC1: file designator (only terminal designators are legal)

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.

Generates an illegal instruction interrupt on error conditions below.

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

STPAR JSYS 217

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. (Refer to 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.

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

STPPN JSYS 556

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 ASCIZ string containing the directory name, a JFN, or a 36-bit directory number

RETURNS +1: always, with the corresponding project-programmer number in AC2

Generates an illegal instruction interrupt on error conditions below.

STPPN ERROR MNEMONICS:

STRX02: Insufficient system resources

STRX03: No such directory name

STRX04: Ambiguous directory specification

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

STSTS JSYS 25

Clears the status of a file. (Refer to 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

STTYP JSYS 302

Sets the terminal type number for the specified terminal line. (Refer to Section 2.4.9.4.)

ACCEPTS IN AC1: terminal designator are legal)

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.

Generates an illegal instruction interrupt on error conditions below.

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

SWJFN JSYS 47

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

Generates an illegal instruction interrupt on error conditions below.

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

SWTRP% JSYS 573

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 LUUO's

```
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 non-zero 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:

```
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
```

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 non-zero section makes use of the UPT (user process table). SWTRP% allows a process to store the desired address in the UPT so that subsequent LUUO's 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 Hardware Reference Manual for more information on LUUO's.

SYERR JSYS 527

Places information in the System Error file (ERROR.SYS). (Refer to 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.

SYERR ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required

SYEX1: Unreasonable SYSERR block size

SYEX2: No buffer space available for SYSERR

SYSGT JSYS 16

Returns the table number, table length, and word 0 of the specified system table. (Refer to 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.

TBADD JSYS 536

Adds an entry to a standard-formatted command table used for user program command recognition. (Refer to the TBLUK call description for the format of the command table.)

ACCEPTS IN AC1: address of word 0 (header word) of table

AC2: entry to be added to table. (Refer to 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

TBDEL JSYS 535

Deletes an entry from a standard-formatted command table used for user program command recognition. (Refer to the TBLUK call description for the format of the command table.)

ACCEPTS IN AC1: address of word 0 (header word) of 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

TBLUK JSYS 537

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

```
0                        17 18                           35
!=========================================================!
!             ADR           !       for use by program    !
!=========================================================!
```

Argument Block

```
     0            6 7                                    35
     !=========================================================!
ADR  !      0     !1!             data bits                    !
     !---------------------------------------------------------!
     !                    start of string                      !
     !=========================================================!
```

The following bits are currently defined:

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 (e.g., 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

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 (i.e., 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

TEXTI JSYS 524

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

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).

Word	Symbol	Meaning
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 (i.e., 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 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.

```
                              destination buffer
                                             can be edited
!==========================================================!
!  CTRL/R buffer;          ! Can be edited, !      !       !
!  cannot be edited,       ! but user is    !      !       !
!  and will be output      ! informed       !      !       !
!  on a CTRL/R             !                !      !       !
!==========================================================!

CTRL/R          Beginning of          Backup       Destination
buffer          destination           limit        string
pointer         buffer pointer        pointer      pointer
(.RDRTY)        (.RDBFP)              (.RDBKL)     (.RDDBP)
```

The flag bits that can be set in word 1 (.RDFLG) of the argument block are as follows:

Bit	Symbol	Meaning
0	RD%BRK	Break on CTRL/Z or ESC.
1	RD%TOP	TOPS-10 character set. Break on CTRL/G, CTRL/K, CTRL/L, CTRL/Z, ESC, carriage return, line feed.
2	RD%PUN	Break on punctuation: CTRL/A-CTRL/F ASCII codes 34-37 CTRL/H-CTRL/I ASCII codes 40-57 CTRL/N-CTRL/Q ASCII codes 72-100 CTRL/S-CTRL/T ASCII codes 133-140 CTRL/X-CTRL/Y ASCII codes 173-176
3	RD%BEL	Break on end of line (carriage return and line feed, or line feed only).
4	RD%CRF	Suppress a carriage return and return a line feed only.
5	RD%RND	Return to user program if the user tries to delete beyond the beginning of the destination buffer. If this bit is not set, the TEXTI call causes the terminal's bell to ring and waits for more input.

Bit	Symbol	Meaning
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 (i.e., do not print XXX and on display terminals, do not delete the characters from the screen).

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

TFORK JSYS 321

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.
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.

Code	Symbol	Meaning
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	.TFUUO	Set monitor call intercepts for TOPS-10 monitor calls (UUOs) 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 .TFUUO. 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), .TFUUO (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.

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. RFORK 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 JSYS's:

1. TFORK (trap)

2. RTFRK (read)

3. UTFRK (untrap)

Generates an illegal instruction interrupt on error conditions below.

TFORK ERROR MNEMONICS:

FRKHX8: Illegal to manipulate an execute-only process

TFRKX1: Invalid function code

TFRKX2: Unassigned process handle or not immediate inferior

TFRKX3: Process not frozen

THIBR JSYS 770

Blocks the current process for the specified elapsed time or until awakened by a TWAKE monitor call, whichever occurs first. The THIBR call is a temporary call and may not be defined in future releases.

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

TIME JSYS 14

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.

TIMER JSYS 522

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.

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. Refer to 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 (refer to 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 (refer to section 2.9.2) of the date and time of the interrupt request to be removed. AC3 is not used for this function.

Code	Symbol	Meaning
4	.TIMBF	Remove any pending interrupt requests that are to occur for the process before the given time. AC2 contains the internal format (refer to 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

TLINK JSYS 216

Controls terminal linking. (Refer to 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.
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 refusing links the object designator is.
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 (TL%OBJ)	Object designator

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.

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

TMON JSYS 7

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 parameters with the CRDIR monitor call.
13	.SFNVT	ARPANET terminal LOGINs 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.
44	.SFNTN	ARPANET is on.
45	.SFNDU	ARPANET will be reinitialized if it is down.
46	.SFNHI	ARPANET host table will be initialized.
47	.SFTMZ	Local time zone
50	.SFLHN	ARPANET 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

Code	Symbol	Meaning
		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 1B0 — MT%UUT — Set = unload unrecognizable tapes; Clear = treat unrecognizable tapes as unlabeled
61	.SFWSP	Enable working set preloading

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

TTMSG JSYS 775

Sends a message to a specified terminal or to all terminals. The TTMSG call is a temporary call and may not be defined in future releases.

RESTRICTIONS: requires WHEEL or OPERATOR capability enabled to send to all terminals

ACCEPTS IN AC1: 400000 + TTY number, or -1 to send to all terminals

AC2: byte pointer to string in caller's address space to be sent

RETURNS +1: always

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

TWAKE JSYS 771

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. The TWAKE call is a temporary call and may not be defined in future releases.

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

UFPGS JSYS 525

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. The FDB is 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.

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

USAGE JSYS 564

Controls accounting on the system by writing entries into the system's 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.

Refer to 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 function is used internally by the LOGIN monitor call. AC2 contains the address of the record descriptor block.

Code	Symbol	Meaning
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 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.

Code	Symbol	Meaning
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. (Refer to Appendix D for the definition of the UITEM. macro.) 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

USRIO JSYS 310

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 (e.g., 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

UTEST JSYS 563

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 or OPERATOR 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:

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

UTFRK JSYS 323

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 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

VACCT JSYS 566

Verifies accounts by validating the supplied account for the given user.

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

WAIT JSYS 306

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. (Refer to Section 2.6.7.)

WFORK JSYS 163

Causes the current process to wait for an inferior process to terminate (voluntarily or involuntarily). A process is considered terminated if its state is either .RFHLT or .RFFPT (refer to RFSTS JSYS for a description of process status).

ACCEPTS IN AC1: inferior process handle, or -4 in the right half to wait for any one of the inferior processes to terminate

RETURNS +1: always, when one of the specified processes terminates

This call returns immediately if the specified process or one of the inferior processes 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

WILD% JSYS 565

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:

Code	Symbol	Meaning
0	.WLSTR	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:

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.

On return, AC1 contains zero if a match occurred, or the following flags if no match occurred:

B0(WL%NOM)	If set, this bit indicates that the non-wild string did not match the wild string.
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.

Code	Symbol	Meaning
1	.WLJFN	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:

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

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.

WILD% ERROR MNEMONICS:

DESX3: JFN is not assigned

RDTX1: Invalid string pointer

ARGX02: Invalid function

ARGX22: Invalid flags

XGSEV% JSYS 614

Gets an extended special entry vector that has been set to allow use of TOPS-10 Compatibility and RMS entry vectors in non-zero sections.

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.

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

XGTPW% JSYS 612

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

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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

XGVEC% JSYS 606

Returns the entry vector of the specified process. The process can be one that runs in one or more sections of memory. (Refer to 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:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

XRIR% JSYS 601

Reads the addresses of the channel and priority level tables for the specified process. (Refer to 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, refer to Section 2.6.3.

XRMAP% JSYS 611

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:

```
!=======================================================!
!   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.

Note that 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

XSFRK% JSYS 605

Starts the specified process in a non-zero 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:

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

XSIR% JSYS 602

Sets the addresses of the channel and priority level tables for the specified process. (Refer to 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

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, refer to 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

XSSEV% JSYS 613

Allows setting of extended special entry vector for use with TOPS-10 Compatibility and RMS entry vectors in non-zero sections.

ACCEPTS IN AC1: vector type code,,fork handle

AC2: length of entry vector

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 UUO 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 UUO word. If this bit is off, the entry vector points to old format PC and UUO words.

XSSEV% ERROR MNEMONICS:

XSEVX1: Illegal entry vector type

XSEVX2: Invalid entry vector length

XSVEC% JSYS 607

Sets or clears the entry vector of the specified process. The process can be one that runs in one or more sections of memory. (Refer to 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

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
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

Table A-1 (Cont.)
ASCII and SIXBIT Collating Sequence and Conversion to EBCDIC

Character	SIXBIT	ASCII 7-bit	EBCDIC 9-bit	Character	ASCII 7-bit	EBCDIC 9-bit
@	40	100	174	#	140	171
A	41	101	301	a	141	201
B	42	102	302	b	142	202
C	43	103	303	c	143	203
D	44	104	304	d	144	204
E	45	105	305	e	145	205
F	46	106	306	f	146	206
G	47	107	307	g	147	207
H	50	110	310	h	150	210
I	51	111	311	i	151	211
J	52	112	321	j	152	221
K	53	113	322	k	153	222
L	54	114	323	l	154	223
M	55	115	324	m	155	224
N	56	116	325	n	156	225
O	57	117	326	o	157	226
P	60	120	327	p	160	227
Q	61	121	330	q	161	230
R	62	122	331	r	162	231
S	63	123	342	s	163	242
T	64	124	343	t	164	243
U	65	125	344	u	165	244
V	66	126	345	v	166	245
W	67	127	346	w	167	246
X	70	130	347	x	170	247
Y	71	131	350	y	171	250
Z	72	132	351	z	172	251
[	73	133	255[1]	{	173	300[1]
\	74	134	340	|	174	117
]	75	135	275	}	175	320
^	76	136	137	~	176	241
–	77	137	155	Delete	177	007

[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.

Table A-2 shows the EBCDIC collating sequence and the conversion from EBCDIC to ASCII.

Table A-2
EBCDIC Collating Sequence and Conversion to ASCII

EBDIC code	EBCDIC character	ASCII code	ASCII character	EBCDIC code	EBCDIC character	ASCII code	ASCII character
000	NUL	000	NUL	050		134	\
001	SOH	001	SOH	051		134	\
002	STX	002	STX	052	SM	030	CAN
003	ETX	003	ETX	053	CUZ	134	\
004	PF	024	DC4	054		134	\
005	HT	011	HT	055	ENQ	005	ENQ
006	LC	016	SO	056	ACK	006	ACK
007	Delete	177	Delete	057	BEL	007	BEL
010		134	\	060		134	\
011		134	\	061		134	\
012	SMM	134	\	-62		134	\
013	VT	013	VT	063		134	\
014	FF	014	FF	064	PN	022	DC2
015	CR	134	\	065	RS	023	DC3
016	SO	134	\	066	UC	017	SI
017	SI	134	\	067	EOT	004	EOT
020	DLE	134	\	070		134	\
021	DC1	134	\	071		134	\
022	DC2	134	\	072		134	\
023	TM	034	FS	073		134	\
024	RES	021	DC1	074	CU3	134	\
025	NL	015	CR	075	DC4	025	NAK
026	BS	010	BS	076	NAK	134	\
027	IL	026	SYN	077	SUB	134	\
030	CAN	134	\	100	Space	040	Space
031	EM	031	EM	101		134	\
032	CC	032	SUB	102		134	\
033	CU1	134	\	103		134	\
034	IFS	134	\	104		134	\
035	IGS	134	\	105		134	\
036	IRS	134	\	106		134	\
037	IUS	134	\	107		134	\
040	DS	036	RS	110		134	\
041	SOS	035	GS	111		134	\
042	FS	037	US	112	CENT	134	\
043		134	\	113	.	056	.
044	BYP	020	DLE	114	<	074	<
045	LF	012	LF	115	(	050	(
046	ETB	027	ETB	116	+	053	+
047	ESC	033	ESC	117	|	174	|

Table A-2 (Cont.)
EBCDIC Collating Sequence and Conversion to ASCII

EBCDIC code	EBCDIC character	ASCII code	ASCII character	EBCDIC code	EBCDIC character	ASCII code	ASCII character
120	&	046	&	170		134	\
121		134	\	171		140	
122		134	\	172	:	072	:
123		134	\	173	#	043	#
124		134	\	174	@	100	@
125		134	\	175	'	47	'
126		134	\	176	=	075	=
127		134	\	177	"	042	"
130		134	\	200		134	\
131		134	\	201	a	141	a
132	!	041	!	202	b	142	b
133	$	044	$	203	c	143	c
134	*	052	*	204	d	144	d
135	)	051	)	205	e	145	e
136	^	073	^	206	f	146	f
137		137	\	207	g	147	g
140	-	055	-	210	h	150	h
141	\	057	/	211	i	151	i
142		134	\	212		134	\
143		134	\	213		134	\
144		134	\	214		134	\
145		134	\	215		134	\
146		134	\	216		134	\
147		134	\	217		134	\
150		134	\	220		134	\
151		134	\	221	j	152	j
152		134	\	222	k	153	k
153	,	054	,	223	l	154	l
154	%	045	%	224	m	155	m
155		137		225	n	156	n
156	>	076	>	226	o	157	o
157	?	077	?	227	p	160	p
160		134	\	230	q	161	q
161		134	\	231	r	162	r
162		134	\	232		134	\
163		134	\	233		134	\
164		134	\	234		134	\
165		134	\	235		134	\
166		134	\	236		134	\
167		134	\	237		134	\

Table A-2 (Cont.)
EBCDIC Collating Sequence and Conversion to ASCII

EBCDIC code	EBCDIC character	ASCII code	ASCII character	EBCDIC code	EBCDIC character	ASCII code	ASCII character
240		134	\	310	H	110	H
241		176	~	311	I	110	I
242	s	163	s	312		134	\
243	t	164	t	313		134	\
244	u	165	u	314		134	\
245	v	166	v	315		134	\
246	w	167	w	316		134	\
247	x	170	x	317		134	\
250	y	171	y	320		175	}
251	z	172	z	321	J	112	J
252		134	\	322	K	113	K
253		134	\	323	L	114	L
254		134	\	324	M	115	M
255	[	133	[	325	N	116	N
256		134	\	326	O	117	O
257		134	\	327	P	120	P
260		175		330	Q	121	Q
261		134	\	331	R	122	R
262		134	\	332		134	\
263		134	\	333		134	\
264		134	\	334		134	\
265		134	\	335		134	\
266		134	\	336		134	\
267		134	\	337		134	\
270		134	\	340		134	\
271		134	\	341		134	\
272		134	\	342	S	123	S
273		134	\	343	T	124	T
274		134	\	344	U	125	U
275	]	135	]	345	V	126	V
276		134	\	346	W	127	W
277		134	\	347	X	130	X
300		173	{	350	Y	131	Y
301	A	101	A	351	Z	132	Z
302	B	102	B	352		134	\
303	C	103	C	353		134	\
304	D	104	D	354		134	\
305	E	105	E	355		134	\
306	F	106	F	356		134	\
307	G	107	G	357		134	\
360	0	060	1	370	8	070	8
361	1	061	1	371	9	071	9
362	2	062	2	372		134	\
363	3	063	3	373		134	\
364	4	064	4	374		134	\
365	5	065	5	375		134	\
366	6	066	6	376		134	\
367	7	067	7	377		134	\

APPENDIX B

MONSYM

```
; UPD ID= 74, <5.UTILITIES>MONSYM.MAC.78,  22-Jan-82 16:08:17 by MURPHY


;THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED
;  OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.
;
;COPYRIGHT (C) 1976,1977,1978,1979,1980,1981 BY DIGITAL EQUIPMENT CORPORATION,
;MAYNARD, MASS.

SEARCH MACSYM                   ;SOME SYMBOLS ARE DEFINED VIA MACROS

;MONITOR CALL DEFINITIONS AND ERROR MNEMONICS

;NOTE:
;       THE FOLLOWING SYMBOLS ARE RESERVED:
;
;               SYMBOL          RESERVED BY
;               ======          ===========
;
;               .OF???          RMS-20
;               .SZ???          RMS-20
;               .PS???          RMS-20

;MACRO TO DEFINE JSYS NAMES

DEFINE DEFJS (NAME,NUM,SECT,XTRA,OLDNEW)<
        OPDEF NAME'% [104B8+NUM]
        IFDEF .PSECT,<
        INTERN NAME'%>
        IFIDN <OLDNEW>,<OLD>,<
        OPDEF NAME [104B8+NUM]
        IFDEF .PSECT,<
        INTERN NAME>>>

        SALL

IFNDEF REL,<REL==0>             ;ASSEMBLING REL IF NON-0
  IFE REL,<
        UNIVERSAL MONSYM>
   IFN REL,<
        TITLE MONSYM
        IFNDEF .PSECT,<
        .DIRECT .XTABM>
        >
```

```
;JSYS DEFINITIONS WITH 'NIM' AS A FOURTH ARGUMENT ARE CLASSIFIED
; AS 'NOT IN MONITOR'

DEFINE JSLIST <

DEFJS LOGIN,1,MSEC1,,OLD
DEFJS CRJOB,2,MSEC1,,OLD
DEFJS LGOUT,3,MSEC1,,OLD
DEFJS CACCT,4,MSEC1,,OLD
DEFJS EFACT,5,MSEC1,NIM,OLD
DEFJS SMON,6,MSEC1,,OLD
DEFJS TMON,7,MSEC1,,OLD
DEFJS GETAB,10,MSEC1,,OLD
DEFJS ERSTR,11,MSEC1,,OLD
DEFJS GETER,12,MSEC1,,OLD
DEFJS GJINF,13,MSEC1,,OLD
DEFJS TIME,14,MSEC1,,OLD
DEFJS RUNTM,15,MSEC1,,OLD
DEFJS SYSGT,16,MSEC1,,OLD
DEFJS GNJFN,17,MSEC1,,OLD
DEFJS GTJFN,20,MSEC1,,OLD
DEFJS OPENF,21,MSEC1,,OLD
DEFJS CLOSF,22,MSEC1,,OLD
DEFJS RLJFN,23,MSEC1,,OLD
DEFJS GTSTS,24,MSEC1,,OLD
DEFJS STSTS,25,MSEC1,,OLD
DEFJS DELF,26,MSEC1,,OLD
DEFJS SFPTR,27,MSEC1,,OLD
DEFJS JFNS,30,MSEC1,,OLD
DEFJS FFFFP,31,MSEC1,,OLD
DEFJS RDDIR,32,MSEC1,,OLD        ;OBSOLETE,,OLD
DEFJS CPRTF,33,,NIM,OLD
DEFJS CLZFF,34,MSEC1,,OLD
DEFJS RNAMF,35,MSEC1,,OLD
DEFJS SIZEF,36,MSEC1,,OLD
DEFJS GACTF,37,MSEC1,,OLD
DEFJS STDIR,40,MSEC1,,OLD        ;OBSOLETE,,OLD
DEFJS DIRST,41,MSEC1,,OLD
DEFJS BKJFN,42,MSEC1,,OLD
DEFJS RFPTR,43,MSEC1,,OLD
DEFJS CNDIR,44,,NIM,OLD
DEFJS RFBSZ,45,MSEC1,,OLD
DEFJS SFBSZ,46,MSEC1,,OLD
DEFJS SWJFN,47,MSEC1,,OLD
DEFJS BIN,50,MSEC1,,OLD
DEFJS BOUT,51,MSEC1,,OLD
DEFJS SIN,52,MSEC1,,OLD
DEFJS SOUT,53,MSEC1,,OLD
DEFJS RIN,54,MSEC1,,OLD
DEFJS ROUT,55,MSEC1,,OLD
DEFJS PMAP,56,MSEC1,,OLD
DEFJS RPACS,57,MSEC1,,OLD
DEFJS SPACS,60,MSEC1,,OLD
DEFJS RMAP,61,MSEC1,,OLD
DEFJS SACTF,62,MSEC1,,OLD
DEFJS GTFDB,63,MSEC1,,OLD
DEFJS CHFDB,64,MSEC1,,OLD
DEFJS DUMPI,65,MSEC1,,OLD
DEFJS DUMPO,66,MSEC1,,OLD
DEFJS DELDF,67,MSEC1,,OLD
DEFJS ASND,70,MSEC1,,OLD
DEFJS RELD,71,MSEC1,,OLD
DEFJS CSYNO,72,,NIM,OLD
```

```
DEFJS PBIN,73,MSEC1,,OLD
DEFJS PBOUT,74,MSEC1,,OLD
DEFJS PSIN,75,,NIM,OLD
DEFJS PSOUT,76,MSEC1,,OLD
DEFJS MTOPR,77,MSEC1,,OLD
DEFJS CFIBF,100,MSEC1,,OLD
DEFJS CFOBF,101,MSEC1,,OLD
DEFJS SIBE,102,MSEC1,,OLD
DEFJS SOBE,103,MSEC1,,OLD
DEFJS DOBE,104,MSEC1,,OLD
DEFJS GTABS,105,MSEC1,,OLD          ;OBSOLETE
DEFJS STABS,106,MSEC1,,OLD          ;OBSOLETE
DEFJS RFMOD,107,MSEC1,,OLD
DEFJS SFMOD,110,MSEC1,,OLD
DEFJS RFPOS,111,MSEC1,,OLD
DEFJS RFCOC,112,MSEC1,,OLD
DEFJS SFCOC,113,MSEC1,,OLD
DEFJS STI,114,MSEC1,,OLD
DEFJS DTACH,115,MSEC1,,OLD
DEFJS ATACH,116,MSEC1,,OLD
DEFJS DVCHR,117,MSEC1,,OLD
DEFJS STDEV,120,MSEC1,,OLD
DEFJS DEVST,121,MSEC1,,OLD
DEFJS MOUNT,122,MSEC1,,OLD          ;OBSOLETE
DEFJS DSMNT,123,,,OLD               ;OBSOLETE
DEFJS INIDR,124,MSEC1,,OLD          ;OBSOLETE
DEFJS SIR,125,MSEC1,,OLD
DEFJS EIR,126,MSEC1,,OLD
DEFJS SKPIR,127,MSEC1,,OLD
DEFJS DIR,130,MSEC1,,OLD
DEFJS AIC,131,MSEC1,,OLD
DEFJS IIC,132,MSEC1,,OLD
DEFJS DIC,133,MSEC1,,OLD
DEFJS RCM,134,MSEC1,,OLD
DEFJS RWM,135,MSEC1,,OLD
DEFJS DEBRK,136,MSEC1,,OLD
DEFJS ATI,137,MSEC1,,OLD
DEFJS DTI,140,MSEC1,,OLD
DEFJS CIS,141,MSEC1,,OLD
DEFJS SIRCM,142,MSEC1,,OLD
DEFJS RIRCM,143,MSEC1,,OLD
DEFJS RIR,144,MSEC1,,OLD
DEFJS GDSTS,145,MSEC1,,OLD
DEFJS SDSTS,146,MSEC1,,OLD
DEFJS RESET,147,MSEC1,,OLD
DEFJS RPCAP,150,MSEC1,,OLD
DEFJS EPCAP,151,MSEC1,,OLD
DEFJS CFORK,152,MSEC1,,OLD
DEFJS KFORK,153,MSEC1,,OLD
DEFJS FFORK,154,MSEC1,,OLD
DEFJS RFORK,155,MSEC1,,OLD
DEFJS RFSTS,156,MSEC1,,OLD
DEFJS SFORK,157,MSEC1,,OLD
DEFJS SFACS,160,MSEC1,,OLD
DEFJS RFACS,161,MSEC1,,OLD
DEFJS HFORK,162,MSEC1,,OLD
DEFJS WFORK,163,MSEC1,,OLD
DEFJS GFRKH,164,MSEC1,,OLD
DEFJS RFRKH,165,MSEC1,,OLD
DEFJS GFRKS,166,MSEC1,,OLD
DEFJS DISMS,167,MSEC1,,OLD
DEFJS HALTF,170,MSEC1,,OLD
DEFJS GTRPW,171,MSEC1,,OLD
```

```
DEFJS GTRPI,172,MSEC1,,OLD
DEFJS RTIW,173,MSEC1,,OLD
DEFJS STIW,174,MSEC1,,OLD
DEFJS SOBF,175,MSEC1,,OLD
DEFJS RWSET,176,MSEC1,,OLD
DEFJS GETNM,177,MSEC1,,OLD
DEFJS GET,200,MSEC1,,OLD
DEFJS SFRKV,201,MSEC1,,OLD
DEFJS SAVE,202,MSEC1,,OLD
DEFJS SSAVE,203,MSEC1,,OLD
DEFJS SEVEC,204,MSEC1,,OLD
DEFJS GEVEC,205,MSEC1,,OLD
DEFJS GPJFN,206,MSEC1,,OLD
DEFJS SPJFN,207,MSEC1,,OLD
DEFJS SETNM,210,MSEC1,,OLD
DEFJS FFUFP,211,MSEC1,,OLD
DEFJS DIBE,212,MSEC1,,OLD
DEFJS FDFRE,213,,NIM,OLD
DEFJS GDSKC,214,MSEC1,,OLD
DEFJS LITES,215,MSEC1,,OLD          ;OBSOLETE
DEFJS TLINK,216,MSEC1,,OLD
DEFJS STPAR,217,MSEC1,,OLD
DEFJS ODTIM,220,MSEC1,,OLD
DEFJS IDTIM,221,MSEC1,,OLD
DEFJS ODCNV,222,MSEC1,,OLD
DEFJS IDCNV,223,MSEC1,,OLD
DEFJS NOUT,224,MSEC1,,OLD
DEFJS NIN,225,MSEC1,,OLD
DEFJS STAD,226,MSEC1,,OLD
DEFJS GTAD,227,MSEC1,,OLD
DEFJS ODTNC,230,MSEC1,,OLD
DEFJS IDTNC,231,MSEC1,,OLD
DEFJS FLIN,232,MSEC1,,OLD
DEFJS FLOUT,233,MSEC1,,OLD
DEFJS DFIN,234,MSEC1,,OLD
DEFJS DFOUT,235,MSEC1,,OLD

DEFJS CRDIR,240,MSEC1,,OLD
DEFJS GTDIR,241,MSEC1,,OLD
DEFJS DSKOP,242,MSEC1,,OLD
DEFJS SPRIW,243,MSEC1,,OLD
DEFJS DSKAS,244,MSEC1,,OLD
DEFJS SJPRI,245,MSEC1,,OLD
DEFJS STO,246,MSEC1,,OLD
DEFJS ARCF,247,MSEC1,,OLD                   ;ARCHIVE SYSTEM JSYS
DEFJS ASNDP,260,,NIM,OLD
DEFJS RELDP,261,,NIM,OLD
DEFJS ASNDC,262,,NIM,OLD
DEFJS RELDC,263,,NIM,OLD
DEFJS STRDP,264,,NIM,OLD
DEFJS STPDP,265,,NIM,OLD
DEFJS STSDP,266,,NIM,OLD
DEFJS RDSDP,267,,NIM,OLD
DEFJS WATDP,270,,NIM,OLD

DEFJS GTNCP,272,MSEC1,,OLD                ;TOPS20AN
DEFJS GTHST,273,MSEC1,,OLD                ;TOPS20AN
DEFJS ATNVT,274,MSEC1,,OLD                ;TOPS20AN
DEFJS CVSKT,275,MSEC1,,OLD                ;TOPS20AN
DEFJS CVHST,276,MSEC1,,OLD                ;TOPS20AN
DEFJS FLHST,277,MSEC1,,OLD                ;TOPS20AN

DEFJS GCVEC,300,MSEC1,,OLD
```

```
DEFJS SCVEC,301,MSEC1,,OLD
DEFJS STTYP,302,MSEC1,,OLD
DEFJS GTTYP,303,MSEC1,,OLD
DEFJS BPT,304,MSEC1,,OLD            ;OBSOLETE
DEFJS GTDAL,305,MSEC1,,OLD
DEFJS WAIT,306,MSEC1,,OLD
DEFJS HSYS,307,MSEC1,,OLD
DEFJS USRIO,310,MSEC1,,OLD
DEFJS PEEK,311,MSEC1,,OLD
DEFJS MSFRK,312,MSEC1,,OLD
DEFJS ESOUT,313,MSEC1,,OLD
DEFJS SPLFK,314,MSEC1,,OLD
DEFJS ADVIS,315,,NIM,OLD
DEFJS JOBTM,316,,NIM,OLD
DEFJS DELNF,317,MSEC1,,OLD
DEFJS SWTCH,320,MSEC1,,OLD          ;OBSOLETE
DEFJS TFORK,321,MSEC1,,OLD
DEFJS RTFRK,322,MSEC1,,OLD
DEFJS UTFRK,323,MSEC1,,OLD
DEFJS SCTTY,324,MSEC1,,OLD

DEFJS SETER,336,MSEC1,,OLD
```

```
;NEW (NOT IN BBN TENEX) JSYS'S ADDED STARTING AT 500

DEFJS RSCAN,500,MSEC1,,OLD
DEFJS HPTIM,501,MSEC1,,OLD
DEFJS CRLNM,502,MSEC1,,OLD
DEFJS INLNM,503,MSEC1,,OLD
DEFJS LNMST,504,MSEC1,,OLD
DEFJS RDTXT,505,MSEC1,,OLD          ;OBSOLETED BY RDTTY AND TEXTI
DEFJS SETSN,506,MSEC1,,OLD
DEFJS GETJI,507,MSEC1,,OLD
DEFJS MSEND,510,MSEC1,,OLD
DEFJS MRECV,511,MSEC1,,OLD
DEFJS MUTIL,512,MSEC1,,OLD
DEFJS ENQ,513,MSEC1,,OLD
DEFJS DEQ,514,MSEC1,,OLD
DEFJS ENQC,515,MSEC1,,OLD
DEFJS SNOOP,516,MSEC1,,OLD
DEFJS SPOOL,517,MSEC1,,OLD
DEFJS ALLOC,520,MSEC1,,OLD
DEFJS CHKAC,521,MSEC1,,OLD
DEFJS TIMER,522,MSEC1,,OLD
DEFJS RDTTY,523,MSEC1,,OLD
DEFJS TEXTI,524,MSEC1,,OLD
DEFJS UFPGS,525,MSEC1,,OLD
DEFJS SFPOS,526,MSEC1,,OLD
DEFJS SYERR,527,MSEC1,,OLD
DEFJS DIAG,530,MSEC1,,OLD
DEFJS SINR,531,MSEC1,,OLD
DEFJS SOUTR,532,MSEC1,,OLD
DEFJS RFTAD,533,MSEC1,,OLD
DEFJS SFTAD,534,MSEC1,,OLD
DEFJS TBDEL,535,MSEC1,,OLD
DEFJS TBADD,536,MSEC1,,OLD
DEFJS TBLUK,537,MSEC1,,OLD
DEFJS STCMP,540,MSEC1,,OLD
DEFJS SETJB,541,MSEC1,,OLD
DEFJS GDVEC,542,MSEC1,,OLD
DEFJS SDVEC,543,MSEC1,,OLD
DEFJS COMND,544,MSEC1,,OLD
DEFJS PRARG,545,MSEC1,,OLD
DEFJS GACCT,546,MSEC1,,OLD
DEFJS LPINI,547,MSEC1,,OLD
DEFJS GFUST,550,MSEC1,,OLD
DEFJS SFUST,551,MSEC1,,OLD
DEFJS ACCES,552,MSEC1,,OLD
DEFJS RCDIR,553,MSEC1,,OLD
DEFJS RCUSR,554,MSEC1,,OLD
DEFJS MSTR,555,MSEC1,,OLD
DEFJS STPPN,556,MSEC1,,OLD
DEFJS PPNST,557,MSEC1,,OLD
DEFJS PMCTL,560,MSEC1,,OLD
DEFJS PLOCK,561,MSEC1,,OLD
DEFJS BOOT,562,MSEC1,,OLD
DEFJS UTEST,563,MSEC1,,OLD
DEFJS USAGE,564,MSEC1,,OLD
DEFJS WILD,565,MSEC1
DEFJS VACCT,566,MSEC1,,OLD
DEFJS NODE,567,MSEC1,,OLD
DEFJS ADBRK,570,MSEC1,,OLD
DEFJS SINM,571,MSEC1
DEFJS SOUTM,572,MSEC1
DEFJS SWTRP,573,MSEC1
DEFJS GETOK,574,MSEC1
```

```
DEFJS RCVOK,575,MSEC1
DEFJS GIVOK,576,MSEC1
DEFJS SKED,577,MSEC1            ;SCHEDULER CONTROL JSYS
DEFJS MTU,600,MSEC1             ;MTU JSYS
DEFJS XRIR,601,MSEC1            ;EXTENDED RIR
DEFJS XSIR,602,MSEC1            ;EXTENDED SIR
DEFJS PDVOP,603,MSEC1           ;MANIPULATE PROGRAM DATA VECTORS
DEFJS NTMAN,604,MSEC1           ;DECNET NETWORK MANAGEMENT INTERFACE
DEFJS XSFRK,605,MSEC1           ;START FORK AT GLOBAL PC
DEFJS XGVEC,606,MSEC1           ;GET FULL ENTRY VECTOR
DEFJS XSVEC,607,MSEC1           ;SET FULL ENTRY VECTOR
DEFJS RSMAP,610,MSEC1           ;READ SECTION MAP
DEFJS XRMAP,611,MSEC1           ;EXTENDED RMAP
DEFJS XGTPW,612,MSEC1           ;EXTENDED GET TRAP WORD
DEFJS XSSEV,613,MSEC1           ;EXTENDED SET SPECIAL ENTRY VECTOR
DEFJS XGSEV,614,MSEC1           ;EXTENDED GET SPECIAL ENTRY VECTOR

;TEMPORARY JSYS DEFINITIONS

DEFJS SNDIM,750,MSEC1,,OLD            ;TOPS20AN
DEFJS RCVIM,751,MSEC1,,OLD            ;TOPS20AN
DEFJS ASNSQ,752,MSEC1,,OLD            ;TOPS20AN
DEFJS RELSQ,753,MSEC1,,OLD            ;TOPS20AN

DEFJS METER,766,MSEC1           ;METER JSYS. FOR KL ONLY
DEFJS SMAP,767,MSEC1            ;CREATE AND MAP SECTIONS
DEFJS THIBR,770,MSEC1,,OLD
DEFJS TWAKE,771,MSEC1,,OLD
DEFJS MRPAC,772,MSEC1,,OLD
DEFJS SETPV,773,,NIM,OLD
DEFJS MTALN,774,MSEC1,,OLD
DEFJS TTMSG,775,MSEC1,,OLD
DEFJS MDDT,777,MSEC1

>  ;;; END OF DEFINE JSLIST

;NOW EXPAND THE JSYS DEFINITIONS
```

```
JSLIST
;ERROR CONDITION INSTRUCTIONS.  THESE ARE NOP'S UNLESS  IMMEDIATELY
;FOLLOWING A JSYS WHICH FAILS.

OPDEF ERJMP [JUMP 16,0]         ;JUMP ON ERROR
OPDEF ERCAL [JUMP 17,0]         ;CALL ON ERROR (SIMULATE PUSHJ 17,ADR)
   IFNDEF FOR,<
   IFDEF .PSECT,<
INTERN ERJMP,ERCAL
   >>

DEFINE GOPDEF (OP,DEF)<
        OPDEF OP [DEF]
   IFNDEF FOR,<
   IFDEF .PSECT,<
        INTERN OP>>>

; THE FOLLOWING OPCODES ARE USED TO PERFORM THE EXTENDED
; ADDRESSING FUNCTIONS.

GOPDEF XJRSTF,<JRST 5,0>        ;RESTORE FLAGS AND PC
GOPDEF XJEN,<JRST 6,0>          ;RESTORE FLAGS,PC AND DISMISS
GOPDEF XPCW,<JRST 7,0>          ;EXCHANGE FLAGS AND PC
GOPDEF XSFM,<JRST 14,0>         ;SAVE PC FLAGS IN MEMORY
GOPDEF XMOVEI,<SETMI 0,0>       ;EXTENDED MOVEI
GOPDEF XHLLI,<HLLI 0,0>         ;INSTRUCTION TO PUT IMMEDIATE ADDRESS IN LH

;OTHER VARIANTS OF JRST

GOPDEF PORTAL,<JRST 1,0>
GOPDEF JRSTF,<JRST 2,0>
GOPDEF JEN,<JRST 12,0>

IFIW==:1B0                      ;INSTRUCTION FORMAT INDIRECT WORD
EFIW==:0                        ;EXTENDED FORMAT INDIRECT WORD

;THE NO-OPERATION INSTRUCTION (MAY CHANGE FROM PROCESSOR TO PROCESSOR)

GOPDEF NOP,<TRN 0,0>
.NODDT NOP

;SPECIAL LOSEG SYMBOLS

.JBHSO==:75                     ; 0 ,, HIGHSEG ORIGIN PAGE NUMBER
.JBEDV==:112                    ;POINTER TO EXEC DATA VECTOR
        .EDCNT==:0              ;'EDV',,COUNT (INCLUDES THIS WORD)
        .EDHSB==:1              ;POINTER TO HIDDEN SYMBOL MAP SWITCHING BLOCK
        .EDSYM==:2              ;.JBSYM IN SYMBOL SPACE
        .EDUSY==:3              ;.JBUSY IN SYMBOL SPACE
        .EDHSF==:4              ;POINTER TO SYMBOLS HIDDEN FLAG WORD
```

```
;*****************************************
;JSYS SPECIFIC ARGUMENTS
;THE FOLLOWING ARE ORDERED ALPHABETICALLY BY JSYS NAME
;*****************************************


;ACCES - ACCESS A DIRECTORY (E.G., BY CONNECTING)

AC%CON==:1B0                    ;CONNECT TO THE SPECIFIED DIRECTORY
AC%OWN==:1B1                    ;GAIN OWNERSHIP
AC%REM==:1B2                    ;REMOVE OWNERSHIP

;OFFSETS IN ARGUMENT BLOCK

.ACDIR==:0                      ;DIRECTORY DESIGNATOR
.ACPSW==:1                      ;POINTER TO PASSWORD STRING
.ACJOB==:2                      ;JOB NUMBER (-1 FOR SELF)


;ADBRK - Address break JSYS function codes and bits

;FUNCTION CODES

.ABSET==:0                      ;SET USER ADDRESS BREAK
.ABRED==:1                      ;READ USER ADDRESS BREAK
.ABCLR==:2                      ;CLEAR USER ADDRESS BREAK
.ABGAD==:3                      ;GET ADDRESS OF TRAPPED INSTRUCTION

;FUNCTION BITS FOR FUNCTION .ABSET

AB%RED==:1B0                    ;READ
AB%WRT==:1B1                    ;WRITE
AB%XCT==:1B2                    ;EXECUTE

;ALLOC JSYS FUNCTION CODES

.ALCAL==:0                      ;ALLOCATE A DEVICE


; ARCF

.ARRAR==:0                      ; Request file archive (user)
        .ARCLR==:0              ; Clear the request
        .ARSET==:1              ; Set the request
;  AR%NDL can be specified in AC2, defined elsewhere
.ARRIV==:1                      ; Request file migration (system)
.AREXM==:2                      ; File exempt from migration (system)
.ARRFR==:3                      ; Request file's contents be restored to disk
   AR%NMS==:1B0                 ; Request no msg on restoration
   AR%WAT==:1B1                 ; Wait for file to be restored to disk
.ARDIS==:4                      ; Clear archive status for file
  AR%CR1==:1B0                  ; Clear 1st run info
  AR%CR2==:1B1                  ; Clear 2nd run info
.ARSST==:5                      ; Set archive status for file
  .AROFL==:0                    ; Flags
    AR%O1==:1B0                 ; Set run 1 info
    AR%O2==:1B1                 ; Set run 2 info
    AR%OFL==:1B2                ; Flush contents of file
    AR%ARC==:1B3                ; Set FB%ARC (archive the file)
    AR%CRQ==:1B4                ; Clear archive/migration request
  .ARTP1==:1                    ; Tape 1 ID
  .ARSF1==:2                    ; XWD TSN 1, TFN 1
    AR%TSN==:777777B17          ; Tape saveset number
```

```
    AR%TFN==:777777B35          ; Tape file number
  .ARTP2==:3                    ; Tape 2 ID
  .ARSF2==:4                    ; XWD TSN 2, TFN 2
;;;     AR%TSN==:777777B17      ; Tape saveset number
;;;     AR%TFN==:777777B35      ; Tape file number
  .ARODT==:5                    ; Date and time
  .ARPSZ==:6                    ; Number of pages in the file (.ARGST only)
.ARRST==:6                      ; Restore contents to archived file
.ARGST==:7                      ; Get tape info for file (blk as for ARSST)
.ARRFL==:10                     ; Retrieve failed
.ARNAR==:11                     ; Set/clear resist archive

; Function & reason codes for IPCF msgs

.RETM==:0                       ; Send retrieve message
  .RETR==:0                     ; Normal retrieve
  .RETRW==:1                    ; User waiting for retrieve
.NOTM==:1                       ; Send notification message
  .FLXP==:0                     ; Archive file expunged
  .ACLR==:1                     ; Archive status cleared

;ATNVT                  ;TOPS20AN

AN%NTP==:1B2            ;TOPS20AN ;NEW TELNET PROTOCOL


;ATACH

AT%CCJ==:1B0                    ;^C JOB WHEN ATTACHED
AT%NAT==:1B1                    ;NO ATTACH
AT%TRM==:1B2                    ;ATTACH JOB TO TERMINAL IN REGISTER 4
AT%JOB==:777777B35              ;JOB NUMBER
```

```
;BOOT

.BTROM==:0                      ;ACTIVATE ROM BOOT
        .BTDTE==:0              ;DTE-20 NUMBER
.BTLDS==:1                      ;LOAD SECONDARY BOOTSTRAP PROGRAM
        .BTERR==:1              ;ERROR FLAGS
        .BTSEC==:2              ;ADDRESS OF SECONDARY BOOTSTRAP PROGRAM
.BTLOD==:2                      ;LOAD MEMORY (OBSOLETE)
.BTSMP==:2                      ;SEND MOP MESSAGE
        .BTFLG==:3              ;FLAGS
                BT%BEL==:1B0    ;SEND TO -11 DOORBELL AFTER SETUP
        .BTCNT==:4              ;NUMBER OF BYTES TO BE TRANSFERRED
        .BTLPT==:5              ;BYTE POINTER TO DATA TO BE LOADED
.BTDMP==:3                      ;DUMP MEMORY
        .BTDPT==:5              ;BYTE POINTER TO DESTINATION OF DUMPED DATA
.BTIPR==:4                      ;INITIALIZE COMMUNICATIONS PROTOCOL
        .BTPRV==:1              ;PROTOCOL VERSION NUMBER
.BTTPR==:5                      ;TERMINATE COMMUNICATIONS PROTOCOL
.BTSTS==:6                      ;RETURN PROTOCOL STATUS
        .BTCOD==:1              ;STATUS CODE
.BTBEL==:7                      ;WAIT FOR DOORBELL
.BTRMP==:10                     ;READ MOP MESSAGE
        .BTMPT==:5              ;POINTER TO DESTINATION FOR MOP MESSAGE
.BTKML==:11                     ;LOAD KMC11
        .BTKMC==:0              ;KMC11 ADDRESS
        .BTKER==:1              ;ERROR FLAGS
                BT%CVE==:1B0    ;CRAM VERIFY ERROR (RH IS BAD DATA)
                BT%DVE==:1B1    ;DRAM VERIFY ERROR (RH IS BAD DATA)
                BT%RVE==:1B2    ;REG VERIFY ERROR (RH IS BAD DATA)
        .BTKCC==:2              ;COUNT OF CRAM DATA
        .BTKCP==:3              ;POINTER TO CRAM DATA (16 BIT DATA)
        .BTKDC==:4              ;COUNT OF DRAM DATA
        .BTKDP==:5              ;POINTER TO DRAM DATA (8 BIT DATA)
        .BTKRC==:6              ;COUNT OF REGISTER DATA
        .BTKRP==:7              ;POINTER TO REGISTER DATA (16 BIT DATA)
        .BTKSA==:10             ;RH IS STARTING ADDRESS
                BT%KSA==:1B0    ;IS SET RH WANT TO START KMC11
.BTKMD==:12                     ;DUMP KMC11
.BTRLC==:13                     ;RETURN LINE COUNTERS
        .BTPRT==:0              ;PORT NUMBER
                BT%ZRO==:1B0    ;CLEAR COUNTERS AFTER READING
        .BTZTM==:1              ;TIME SINCE COUNTERS HAVE BEEN ZEROED
        .BTSCC==:2              ;STATUS COUNT COUNT
        .BTSCP==:3              ;STATUS COUNT POINTER
        .BTRCC==:4              ;RECEIVE COUNT COUNT
        .BTRCP==:5              ;RECEIVE COUNT POINTER
        .BTTCC==:6              ;TRANSMIT COUNT COUNT
        .BTTCP==:7              ;TRANSMIT COUNT POINTER
.BTCLI==:14                     ;CONVERT LINEID TO PORT NUMBER
        .BTLID==:1              ;POINTER TO ASCIZ LINE-ID
.BTCPN==:15                     ;CONVERT PORT NUMBER TO LINE-ID
.BTD60==:16                     ;DN60 PROTOCOL OPERATION
        DEFSTR  (BT6DTE,0,35,36)        ;DTE number
        .BT6DTE==:0
        DEFSTR  (BT6ERR,1,35,36)        ;returned error flags
        .BT6ERR==:1
                                        ;protocol flags
                D6%BSY==:1B0            ;port is busy - sign bitness is used
                                        ; in testing
                D6%QHD==:1B1            ;header has been queued
                D6%HDD==:1B2            ;to -11 done for header seen
                D6%NDT==:1B3            ;this is a no-data-transfer operation
                D6%RED==:1B4            ;this is a read data type operation
                D6%QDT==:1B5            ;data has been queued(for write fcn)
```

```
                D6%DTD==:1B6            ;to -11 done for write data seen
                D6%RBL==:1B7            ;to -10 doorbell for response header se
                D6%RDN==:1B8            ;to -10 done for response header seen
                D6%DBL==:1B9            ;to -10 doorbell for read data seen
                D6%DDN==:1B10           ;to -10 done for read data seen
                D6%FDN==:1B11           ;to -10 done for read data was faked
                                        ;error flags
                D6%BDP==:1B30           ;bad data byte ptr
                D6%ARD==:1B31           ;11 attempted to send read data when
                                        ; when none was expected
                D6%TRS==:1B32           ;timed out waiting for response header
                D6%TDT==:1B33           ;timed out waiting for read data

                D6%TPO==:1B34           ;timed out waiting for port to be free
                D6%NT6==:1B35           ;not a DN60 front end

        DEFSTR  (BT6HBC,2,17,18)        ;DN60 header byte count
        .BT6HBC==:2
        DEFSTR  (BT6HDR,2,35,18)        ;DN60 header address(begins on word)
        .BT6HDR==:2
        DEFSTR  (BT6DBC,3,35,36)        ;data byte count
        .BT6DBC==:3
                                        ;       positive => write data mode
                                        ;       zero     => no data transfer
                                        ;       negative => read data mode
        DEFSTR  (BT6PTR,4,35,36)        ;data byte ptr
        .BT6PTR==:4
                                        ;the following are returned for timing
                                        ; analysis
        DEFSTR  (BT6TMR,5,35,36)        ;time of request
        .BT6TMR==:5
        DEFSTR  (BT6TAS,6,35,36)        ;TIME DTE ASSIGNED
        .BT6TAS==:6
        DEFSTR  (BT6THQ,7,35,36)        ;time header queued to 11
        .BT6THQ==:7
        DEFSTR  (BT6TRD,10,35,36)       ;time of -10 done for response header
        .BT6TRD==:10
        DEFSTR  (BT6TDD,11,35,36)       ;time of -10 done for data
        .BT6TDD==:11
        DEFSTR  (BT6TFR,12,35,36)       ;time finished request
        .BT6TFR==:12

.BTSTA==:16                     ;SET STATION POLLING STATUS
.BTSSP==:17                     ;SET LINE STARTUP PRIORITY
        .BTPRI==:1              ;PRIORITY VALUE
.BTSTP==:20                     ;SET STATION POLLING PRIORITY
.BTSDD==:21                     ;SEND DDCMP MESSAGE
        .BTMSG==:1              ;ADDR OF MESSAGE
        .BTLEN==:2              ;BYTE COUNT OF MESSAGE
.BTRDD==:22                     ;RECEIVE A MESSAGE FROM DDCMP
        .BTSUP==:1              ;STATION CAME UP
        .BTSDW==:2              ;STATION WENT DOWN
        .BTCMP==:3              ;XMIT COMPLETE
        .BTSSF==:4              ;STARTUP FAILED
                BT%CTL==:1B0    ;CONTROL MESSAGE
.BTCHN==:23                     ;ASSIGN A SOFTWARE INTERRUPT CHANNEL
        .BTESI==:1              ;CHANNEL NUMBER


;CFORK

CR%MAP==:1B0                    ;SET MAP FOR NEW FORK TO POINT TO
                                ; THIS PROCESS
CR%CAP==:1B1                    ;MAKE CAPABILITIES IDENTICAL
```

```
CR%ACS==:1B3                        ;SET ACS FROM BLOCK
CR%ST==:1B4                         ;START PROCESS AT PC
CR%PCV==:777777B35                  ;VALUE OF PC


;CHFDB

CF%NUD==:1B0                        ;NO UPDATE DIRECTORY
CF%DSP==:777B17                     ;FDB DISPLACEMENT
CF%JFN==:777777B35                  ;JFN
```

```
;CHKAC JSYS DEFINITIONS

;CHKAC FLAG DEFINITIONS

CK%JFN==:1B0                    ;JFN IS GIVEN AS AN ARGUMENT

;CHKAC ARGUMENT BLOCK OFFSET VALUES

.CKAAC==:0                      ;ACCESS CODE
.CKALD==:1                      ;LOGGED IN USER NUMBER OF USER
.CKACD==:2                      ;CONNECTED DIR NUMBER OF USER
.CKAEC==:3                      ;ENABLED CAPABILITIES OF USER BEING CHK'D
.CKAUD==:4                      ;DIR NUMBER OF DIRECTORY CONTAINING FILE
.CKAPR==:5                      ;PROTECTION OF FILE

;CHKAC ACCESS CODES

.CKARD==:0                      ;READ AN EXISTING FILE
.CKAWT==:1                      ;WRITE AN EXISTING FILE
.CKAWR==:1                      ; (ANOTHER NAME FOR ABOVE)
.CKAEX==:2                      ;EXECUTE AN EXISTING FILE
.CKAAP==:3                      ;APPEND TO AN EXISTING FILE
.CKADL==:4                      ;GET DIR LISTING OF AN EXISTING FILE
.CKADR==:6                      ;READ THE DIRECTORY
.CKAOF==:7                      ;OPEN FILES IN DIR (NOT IMPLEMENTED)
.CKACN==:10                     ;CONNECT TO A DIR
.CKACF==:11                     ;CREATE FILES IN DIR


;CLOSF

CO%NRJ==:1B0                    ;NO RELEASE JFN
CO%WCL==:1B1          ;TOPS20AN ;WAIT UNTIL MATCHING CLS IS RECEIVED
CO%JFN==:777777B35              ;JFN


;CLZFF

CZ%NIF==:1B0                    ;NO INFERIOR FORK FILES
CZ%NSF==:1B1                    ;NO SELF FORK FILES
CZ%NRJ==:1B2                    ;NO RELEASE JFN
CZ%NCL==:1B3                    ;NO CLOSE FILE
CZ%UNR==:1B4                    ;UNRESTRICT
CZ%ARJ==:1B5                    ;ALWAYS RELEASE JFN
CZ%ABT==:1B6                    ;ABORT
CZ%NUD==:1B7                    ;NO UPDATE DIRECTORY
CZ%PRH==:777777B35              ;PROCESS HANDLE
```

```
;CNDIR

CN%CKP==:1B0                    ;CHECK PASSWORD ONLY
CN%NOC==:1B1                    ;NO CONNECT
CN%JOB==:1B2                    ;DOING CONNECT FOR ANOTHER JOB
CN%DIR==:777777B35              ;DIRECTORY NUMBER


;COMND

;COMND - COMMAND STATE BLOCK

.CMFLG==:0                      ;USER FLAGS,,REPARSE DISPATCH ADDRESS
.CMIOJ==:1                      ;INJFN,,OUTJFN
.CMRTY==:2                      ;^R BUFFER POINTER
.CMBFP==:3                      ;PTR TO TOP OF BUFFER
.CMPTR==:4                      ;PTR TO NEXT INPUT TO BE PARSED
.CMCNT==:5                      ;COUNT OF SPACE LEFT IN BUFFER AFTER PTR
.CMINC==:6                      ;COUNT OF CHARACTERS FOLLOWING PTR
.CMABP==:7                      ;ATOM BUFFER POINTER
.CMABC==:10                     ;ATOM BUFFER SIZE
.CMGJB==:11                     ;ADR OF GTJFN ARG BLOCK
 CM%GJB==:777777                ;ADR OF GTJFN ARG BLOCK

;COMND - FUNCTION DESCRIPTOR BLOCK

.CMFNP==:0                      ;FUNCTION AND POINTER
  CM%FNC==:777B8                ;FUNCTION CODE
  CM%FFL==:777B17               ;FUNCTION-SPECIFIC FLAGS
  CM%LST==:777777               ;LIST POINTER TO OTHER BLOCKS
.CMDAT==:1                      ;DATA FOR FUNCTION
.CMHLP==:2                      ;HELP TEXT POINTER
.CMDEF==:3                      ;DEFAULT STRING POINTER
.CMBRK==:4                      ;FIELD BREAK MASK POINTER
```

```
;COMND - FLAGS IN .CMFLG

CM%ESC==:1B0                    ;ESC SEEN
CM%NOP==:1B1                    ;NO PARSE
CM%EOC==:1B2                    ;END OF COMMAND SEEN
CM%RPT==:1B3                    ;REPEAT PARSE NEEDED
CM%SWT==:1B4                    ;SWITCH TERMINATED WITH ":"
CM%PFE==:1B5                    ;PREVIOUS FIELD ENDED WITH ESC
CM%RAI==:1B6                    ;RAISE INPUT
CM%XIF==:1B7                    ;EXCLUDE INDIRECT FILES
CM%WKF==:1B8                    ;WAKEUP AFTER EACH FIELD

;FUNCTION BLOCK FLAGS (IN WORD .CMFNP)

CM%NSF==:1B12                   ;SUFFIX MAY BE OMITTED IF DESIRED
CM%BRK==:1B13                   ;BREAK MASK PRESENT
CM%PO==:1B14                    ;PARSE-ONLY
CM%HPP==:1B15                   ;HELP POINTER PRESENT
CM%DPP==:1B16                   ;DEFAULT POINTER PRESENT
CM%SDH==:1B17                   ;SUPPRESS DEFAULT HELP MESSAGE

;FLAGS FOR CMDIR FUNCTION

CM%DWC==:1B0                    ;DIRECTORY WILD CARDING ALLOWED

;FLAGS FOR CMTAD FUNCTION

CM%IDA==:1B0                    ;INPUT DATE
CM%ITM==:1B1                    ;INPUT TIME
CM%NCI==:1B2                    ;NO CONVERT TO INTERNAL

;FLAGS IN KEYWORD TABLE (FIRST WORD OF STRING IF B0-6 = 0)

CM%INV==:1B35                   ;INVISIBLE
CM%NOR==:1B34                   ;NO-RECOGNIZE (PLACEHOLDER)
CM%ABR==:1B33                   ;ABBREVIATION FOR ANOTHER ENTRY
CM%FW==:1B7                     ;FLAG WORD (MUST ALWAYS BE ON)
```

```
;COMND - FUNCTION CODES

.CMKEY==:0                        ;KEYWORD
.CMNUM==:1                        ;NUMBER
.CMNOI==:2                        ;NOISE WORD
.CMSWI==:3                        ;SWITCH
.CMIFI==:4                        ;INPUT FILE
.CMOFI==:5                        ;OUTPUT FILE
.CMFIL==:6                        ;GENERAL FILESPEC
.CMFLD==:7                        ;ARBITRARY FIELD
.CMCFM==:10                       ;CONFIRM
.CMDIR==:11                       ;DIRECTORY NAME
.CMUSR==:12                       ;USER NAME
.CMCMA==:13                       ;COMMA
.CMINI==:14                       ;INIT LINE
.CMFLT==:15                       ;FLOATING POINT NUMBER
.CMDEV==:16                       ;DEVICE NAME
.CMTXT==:17                       ;TEXT TO ACTION CHAR
.CMTAD==:20                       ;TIME AND DATE
.CMQST==:21                       ;QUOTED STRING
.CMUQS==:22                       ;UNQUOTED STRING
.CMTOK==:23                       ;TOKEN
.CMNUX==:24                       ;NUMBER DELIMITED BY NON-DIGIT
.CMACT==:25                       ;ACCOUNT
.CMNOD==:26                       ;NODE NAME

;DEFINE BREAK MASKS

        BRINI.                    ;INITIALIZE BREAK MASK FOR STANDARD FIELD
        BRKCH. (0,37)             ;ALL CONTROL CHARACTERS
        BRKCH. (40,54)            ;SPACE THROUGH COMMA
        BRKCH. (56,57)            ;DOT AND SLASH
        BRKCH. (72,77)            ;COLON THROUGH QUESTION MARK
        BRKCH. (100)              ;ATSIGN
        BRKCH. (133,140)          ;OPEN BRACKET THROUGH ACCENT GRAVE
        BRKCH. (173,177)          ;CLOSE BRACKET THROUGH TILDE

FLDB0.==W0.                       ;STANDARD FIELD BREAK MASK
FLDB1.==W1.
FLDB2.==W2.
FLDB3.==W3.

;KEYWORD BREAK SET.  SAME AS STANDARD FIELD FOR NOW

KEYB0.==FLDB0.
KEYB1.==FLDB1.
KEYB2.==FLDB2.
KEYB3.==FLDB3.
```

```
;USERNAME BREAK SET.  BREAKS ON EVERYTHING EXCEPT DOT AND ALPHABETICS.

        UNBRK. "."              ;MODIFY FIELD BREAK SET INTO USER BREAK SET.
                                ; DON'T BREAK ON DOT
        UNBRK. "%"              ;DON'T BREAK ON PERCENT
        UNBRK. "*"              ;STAR
        UNBRK. "$"              ;ALLOW DOLLARSIGN! (I NEVER KNEW THAT BEFORE)
        UNBRK. "_"              ;ALLOW UNDERSCORE IN ATOM

USRB0.==W0.
USRB1.==W1.
USRB2.==W2.
USRB3.==W3.

;ACCOUNT MASK CURRENTLY THE SAME AS USER MASK

ACTB0.==USRB0.
ACTB1.==USRB1.
ACTB2.==USRB2.
ACTB3.==USRB3.

;FILESPEC FIELD - FILESPEC PUNCTUATION CHARACTERS
;ARE LEGAL ( :, <, >, ., ;)

        UNBRK. ":"              ;MODIFY USERNAME BREAK SET INTO FILE BREAK SET.
                                ;  DON'T BREAK ON THESE
        UNBRK. "<"
        UNBRK. ">"
        UNBRK. "["
        UNBRK. "]"
        UNBRK. ";"

FILB0.==W0.
FILB1.==W1.
FILB2.==W2.
FILB3.==W3.
```

```
;READ DEVICE NAME

BRINI. FLDB0.,FLDB1.,FLDB2.,FLDB3.       ;VERY SIMILAR TO STANDARD FIELD
UNBRK. "$"                          ;ALLOW DOLLARSIGN IN DEVICE NAME (LIKE FILESPI
UNBRK. "_"                          ;UNDERSCORE TOO

DEVB0.==W0.
DEVB1.==W1.
DEVB2.==W2.
DEVB3.==W3.

;READ TO END OF LINE

        BRINI.                      ;IN TIALIZE END OF LINE BREAK SET
        BRKCH. .CHLFD               ;BREAK ON LINEFEED
        BRKCH. .CHCRT               ;AND CARRIAGE RETURN
        BRKCH. .CHFFD               ;FORMFEED IS VALID END-OF-LINE

EOLB0.==W0.
EOLB1.==W1.
EOLB2.==W2.
EOLB3.==W3.
```

```
;CRDIR

CD%LEN==:1B0                    ;FLAGS ,, LENGTH OF CRDIR BLOCK
CD%PSW==:1B1                    ;SET PASSWORD STRING
CD%LIQ==:1B2                    ;SET LOGGED IN QUOTA
CD%PRV==:1B3                    ;SET PRIVILEGES
CD%MOD==:1B4                    ;SET MODE BITS
CD%LOQ==:1B5                    ;SET LOGGED OUT QUOTA
CD%NUM==:1B6                    ;SET DIRECTORY NUMBER FROM PARAM BLK
CD%FPT==:1B7                    ;SET DEFAULT FILE PROTECTION
CD%DPT==:1B8                    ;SET DIRECTORY PROTECTION
CD%RET==:1B9                    ;SET DEFAULT RETENTION COUNT
CD%LLD==:1B10                   ;SET LAST LOGIN DATE
CD%UGP==:1B11                   ;SET USER GROUPS
CD%DGP==:1B12                   ;SET DIRECTORY GROUPS
CD%SDQ==:1B13                   ;SET SUBDIRECTORY QUOTA
CD%CUG==:1B14                   ;SET CREATABLE USER GROUPS
CD%DAC==:1B15                   ;SET DEFAULT ACCOUNT
CD%DEL==:1B17                   ;DELETE DIRECTORY
CD%APB==:777777B35              ;ADDRESS OF PARAMETER BLOCK
.CDLEN==:0                      ;LENGTH OF ARGUMENT BLOCK
        CD%NSQ==:1B0            ;DO NOT UPDATE QUOTAS OF SUPERIOR DIR
        CD%NCE==:1B1            ;DO NOT CHANGE PARAMETERS OF EXISTING DIRS
        CD%NED==:1B2            ; Set def online exp from .CDDNE
        CD%FED==:1B3            ; Set def offline exp from .CDDFE
.CDPSW==:1                      ;POINTER TO PASSWORD STRING
.CDLIQ==:2                      ;LOGGED IN QUOTA
.CDPRV==:3                      ;PRIVILEGE WORD
.CDMOD==:4                      ;MODE WORD
        CD%DIR==:1B0            ;DIRECTORY NAME FOR CNDIR ONLY (FILES ONLY)
        CD%ANA==:1B1            ;ALPHANUMERIC ACCOUNTS
        CD%RLM==:1B2            ;REPEAT LOGIN MESSAGES
        CD%DAR==:1B7            ; Archived online expired files
.CDLOQ==:5                      ;LOGGED OUT QUOTA
.CDNUM==:6                      ;DIRECTORY NUMBER
.CDFPT==:7                      ;DEFAULT FILE PROTECTION
.CDDPT==:10                     ;DIRECTORY PROTECTION
.CDRET==:11                     ;DEFAULT RETENTION COUNT
.CDLLD==:12                     ;LAST LOGIN DATE
.CDUGP==:13                     ;USER GROUPS
.CDDGP==:14                     ;DIRECTORY GROUPS
.CDSDQ==:15                     ;MAXIMUM NUMBER OF SUBDIRECTORIES
.CDCUG==:16                     ;POINTER TO CREATABLE USER GROUP LIST
.CDDAC==:17                     ;POINTER TO DEFAULT ACCOUNT
.CDDNE==:20                     ; Default online expiration
.CDDFE==:21                     ; Default offline expiration
```

```
;CRJOB

CJ%LOG==:1B0                        ;ATTEMPT TO LOG IN THE NEW JOB
CJ%NAM==:1B1                        ;USE NAME AND PSWD IN ARG BLK
CJ%ACT==:3B3                        ;WHERE TO GET ACCOUNT
        .CJUCA==:0                  ;USE CURRENT ACCT OF CREATOR
        .CJUAA==:1                  ;USE ACCOUNT IN ARG BLOCK
        .CJUDA==:2                  ;USE DEFAULT ACCOUNT OF NEW USER
CJ%ETF==:1B4                        ;PUT EXEC IN TOP FORK
CJ%FIL==:1B5                        ;GET FILE IN ARG BLOCK
CJ%ACS==:1B6                        ;LOAD THE ACS FROM ARG BLOCK
CJ%OWN==:1B7                        ;RETAIN OWNERSHIP OF NEW JOB
CJ%WTA==:1B8                        ;NEW JOB WAITS TIL ATTACHED
CJ%NPW==:1B9                        ;NO PASSWORD CHECK AT LOGIN TIME
CJ%NUD==:1B10                       ;NO UPDATE OF LAST-LOGIN DATE
CJ%SPJ==:1B11                       ;DO SPJFN IN NEW JOB FROM ARG BLK
CJ%CAP==:1B12                       ;PASS ENABLED CAPABILITIES AS ALLOWED
CJ%CAM==:1B13                       ;CAPABILITY MASK AT LOGIN
CJ%SLO==:1B14                       ;SIGNAL (IPCF) AT LOGOUT TIME
CJ%DSN==:1B17                       ;DISOWN EXISTING JOB # IN 3

.CJNAM==:0                          ;NAME STRING POINTER
.CJPSW==:1                          ;PASSWORD STRING POINTER
.CJACT==:2                          ;ACCOUNT DESIGNATOR/STRING
.CJFIL==:3                          ;FILE NAME STRING POINTER
.CJSFV==:4                          ;SFRKV OFFSET
.CJTTY==:5                          ;TTY DESIGNATOR, OR NULL DESIGNATOR
.CJTIM==:6                          ;TIME LIMIT
.CJACS==:7                          ;ADDRESS OF 16. WORDS OF AC'S
.CJEXF==:10                         ;EXEC FLAGS, FOR EXEC AC1
.CJPRI==:11                         ;PRIMARY JFN'S FOR SPJFN IN NEW JOB
.CJCPU==:12                         ;CPU LIMIT (0 IF NONE)
.CJCAM==:13                         ;CAPABILITY MASK TO APPLY TO LOGIN
.CJSLO==:14                         ;PID TO SIGNAL AT LOGOUT TIME

CR%PRA==:2545                       ;MAGIC # FOR EXEC/CRJOB LINKAGE VIA PRARG


;CRLNM

.CLNJ1==:0                          ;DELETE 1 LOGICAL NAME FROM JOB
.CLNS1==:1                          ;DELETE 1 LOGICAL NAME FROM SYSTEM
.CLNJA==:2                          ;DELETE ALL JOB WIDE LOGICAL NAMES
.CLNSA==:3                          ;DELETE ALL SYSTEM LOGICAL NAMES
.CLNJB==:4                          ;CREATE A JOB WIDE LOGICAL NAME
.CLNSY==:5                          ;CREATE A SYSTEM WIDE LOGICAL NAME
```

```
;DELDF

DD%DTF==:1B0                      ;DELETE TEMPORARY FILES
DD%DNF==:1B1                      ;DELETE NONEXISTENT FILES
DD%RST==:1B2                      ;REBUILD THE SYMBOL TABLE
DD%CHK==:1B3                      ;CHECK THE DIR FOR CONSISTENCY ONLY


;DELF

DF%NRJ==:1B0                      ;DON'T RELEASE JFN
DF%EXP==:1B1                      ;EXPUNGE CONTENTS
DF%FGT==:1B2                      ;FORGET (EXPUNGE W/O DEASSIGNING ADDRESSES)
DF%DIR==:1B3                      ;DELETE, FORGET, AND EXPUNGE A DIRECTORY
                                  ; FILE.  (ONLY IF ^E-CREATE KILL FAILED)
DF%ARC==:1B4                      ; Delete of archive status file allowed
DF%CNO==:1B5                      ; Delete only contents of file
                                  ;  Immediate expunge implied

DF%JFN==777777B35                 ;JFN


;DIAG JSYS DEFINITIONS

DG%ADT==:7B2                      ;ADDRESS TYPE FIELD
DG%DVC==:177B9                    ;DEVICE CODE FIELD
        .DGRH0==:130              ;MBC0
        .DGRH7==:137              ;MBC7
DG%UNI==:77B29                    ;UNIT NUMBER
DG%SUN==:77B35                    ;SUBUNIT NUMBER

;DIAG JSYS FUNCTION CODES

.DGACU==:1                        ;ASSIGN DEVICE
.DGACH==:2                        ;ASSIGN CONTROLLER AND ALL DEVICES
.DGRCH==:3                        ;RELEASE DEVICE(S)
.DGSCP==:4                        ;SETUP CHANNEL PROGRAM
.DGRCP==:5                        ;RELEASE CHANNEL PROGRAM
.DGGCS==:6                        ;GET CHANNEL STATUS

;DIAG NEW CONTROL FUNCTIONS

.DGGEM==:100                      ;LEAVE LARGE HOLE FOR MORE RH20 FUNCTIONS
                                  ;GET MEM (FOR TGHA)
.DGREM==:101                      ;RELEASE MEM (FOR TGHA)
.DGPDL==:102                      ;UNIT ONLINE
```

```
;DSKAS

DA%DEA==:1B0                    ;DEASSIGN DISK ADDRESS
DA%ASF==:1B1                    ;ASSIGN FREE PAGE
DA%CNV==:1B2                    ;CONVERT SOFTWARE TO HARDWARE ADDRESS
DA%HWA==:1B3                    ;HARDWARE ADDRESS GIVEN
DA%INI==:1B4                    ;INITIALIZE THE BIT TABLE
DA%WRT==:1B5                    ;WRITE THE BIT TABLE FILE
DA%ADR==:777777B35              ;DISK ADDRESS
```

```
;DVCHR, DEVUNT AND DVCH1 BIT DEFINITIONS

DV%OUT==:1B0                  ;DEVICE CAN DO OUTPUT
DV%IN==:1B1                   ;DEVICE CAN DO INPUT
DV%DIR==:1B2                  ;DEVICE HAS A DIRECTORY
DV%AS==:1B3                   ;DEVICE IS ASSIGNABLE
DV%MDD==:1B4                  ;DEVICE IS A MULTIPLE DIRECTORY DEVICE
DV%AV==:1B5                   ;DEVICE IS AVAILABLE TO THIS JOB
DV%ASN==:1B6                  ;DEVICE IS ASSIGNED BY ASND
DV%MDV==:1B7                  ;RESERVED (HISTORICAL)
DV%MNT==:1B8                  ;DEVICE IS MOUNTED
DV%TYP==:777B17               ;DEVICE TYPE FIELD
DV%PSD==:1B18                 ;PSEUDO DEVICE
DV%UNT==:77777                ;UNIT MASK
DV%MOD==:177777B35            ;DEVICE DATA MODE
DV%M0==:1B35                  ;DEVICE CAN BE OPENED IN MODE 0
DV%M1==:1B34                  ;DEVICE CAN BE OPENED IN MODE 1
DV%M2==:1B33                  ;DEVICE CAN BE OPENED IN MODE 2
DV%M3==:1B32                  ;DEVICE CAN BE OPENED IN MODE 3
DV%M4==:1B31                  ;DEVICE CAN BE OPENED IN MODE 4
DV%M5==:1B30                  ;DEVICE CAN BE OPENED IN MODE 5
DV%M6==:1B29                  ;DEVICE CAN BE OPENED IN MODE 6
DV%M7==:1B28                  ;DEVICE CAN BE OPENED IN MODE 7
DV%M10==:1B27                 ;DEVICE CAN BE OPENED IN MODE 10
DV%M11==:1B26                 ;DEVICE CAN BE OPENED IN MODE 11
DV%M12==:1B25                 ;DEVICE CAN BE OPENED IN MODE 12
DV%M13==:1B24                 ;DEVICE CAN BE OPENED IN MODE 13
DV%M14==:1B23                 ;DEVICE CAN BE OPENED IN MODE 14
DV%M15==:1B22                 ;DEVICE CAN BE OPENED IN MODE 15
DV%M16==:1B21                 ;DEVICE CAN BE OPENED IN MODE 16
DV%M17==:1B20                 ;DEVICE CAN BE OPENED IN MODE 17
D1%SPL==:1B0                  ;DEVICE IS SPOOLED
D1%ALC==:1B1                  ;DEVICE IS UNDER CONTROL OF ALLOCATOR
D1%VVL==:1B2                  ;VOLUME VALID
D1%NIU==:1B3                  ;DEVICE SLOT IS NOT IN USE (FOR STRUCTURES
                              ; NOT YET MOUNTED)
D1%INI==:1B4                  ;DEVICE IS BEING INITIALIZED (STRUCTURE
                              ; IS AVAILABLE ONLY TO THE FORK WHOSE NUMBER
                              ; IS STORED IN SDBSTS)
```

```
;DEVICE TYPE DEFINITIONS

.DVDSK==:0                          ;DISK
.DVMTA==:2                          ;MAGTAPE
.DVDTA==:3                          ;DECTAPE
.DVPTR==:4                          ;PAPER TAPE READER
.DVPTP==:5                          ;PAPER TAPE PUNCH
.DVDSP==:6                          ;DISPLAY
.DVLPT==:7                          ;LINE PRINTER
.DVCDR==:10                         ;CARD READER
.DVFE==:11                          ;FRONT END DEVICE
.DVTTY==:12                         ;TERMINAL
.DVPTY==:13                         ;PTY
.DVNUL==:15                         ;NULL DEVICE
.DVNET==:16                         ;ARPA NETWORK
.DVPLT==:17                         ;PLOTTER
.DVCDP==:21                         ;CARD PUNCH
.DVDCN==:22                         ;DECNET ACTIVE COMPONENT
.DVSRV==:23                         ;DECENT PASSIVE COMPONENT
.DVATS==:24                         ;APPLICATIONS TERMINAL SERVICE
.DVADS==:25                         ;AYDIN DISPLAY


;DSKOP

DOP%SA==:1B0                        ;SOFTWARE ADDRESS
DOP%AT==:3B1                        ;ADDRESS TYPE FIELD
 .DOPPU==:1                         ;PHYSICAL CHANNEL AND UNIT
  DOP%CN==:37B6                     ;CHANNEL NUMBER (OLD FORMAT)
  DOP%UN==:77B12                    ;UNIT NUMBER (OLD FORMAT)
  DOP%UA==:37777777                 ;UNIT ADDRESS
 .DOPSR==:2                         ;STRUCTURE AND RELATIVE ADDRESS
  DOP%SN==:777B10                   ;STRUCTURE NUMBER
  DOP%RA==:177777777                ;RELATIVE ADDRESS
  DOP%C2==:7777B11                  ;CHANNEL NUMBER (NEW FORMAT)
  DOP%K2==:7777B23                  ;CONTROLLER NUMBER (NEW FORMAT)
  DOP%U2==:7777B35                  ;UNIT NUMBER (NEW FORMAT)
DOP%NF==:1B9                        ;USE NEW FORMAT FOR CHANNEL, UNIT NUMBERS
DOP%EO==:1B10                       ;ERROR IF UNIT OFFLINE
DOP%IL==:1B11                       ;INHIBIT ERROR LOGGING
DOP%IR==:1B12                       ;INHIBIT ERROR RECOVERY
DOP%WR==:1B14                       ;WRITE
DOP%CT==:777777B35                  ;WORD COUNT

;DUMPI/DUMPO

DM%NWT==:1B0                        ;NO WAIT FOR COMPLETION
DM%FIN==:1B1                        ;FINISH PREVIOUS REQUEST
                                    ;***NOT INPLEMENTED YET***
DM%PTR==:777777B35                  ;POINTER TO COMMAND LIST
```

```
;DEFINE DECNET DISCONNECT CODES. THESE ARE STIPULATED BY THE NSP SPEC
;AND MAY HAVE MEANINGS NOT IMPLIED BY THE COMMENTS

.DCX0==:0                         ;NO SPECIAL ERROR
.DCX1==:1                         ;RESOURCE ALLOCATION FAILURE
.DCX2==:2                         ;DESTINATION NODE DOES NOT EXIST
.DCX3==:3                         ;NODE SHUTTING DOWN
.DCX4==:4                         ;DESTINATION PROCESS DOES NOT EXIST
.DCX5==:5                         ;INVALID NAME FIELD
.DCX6==:6                         ;DESTINATION PROCESS QUEUE OVERFLOW
.DCX7==:7                         ;UNSPECIFIED ERROR
.DCX8==:^D8                       ;THIRD PARTY ABORTED LINK
.DCX9==:^D9                       ;USER ABORT (ASYNCHRONOUS DISCONNECT)
.DCX11==:^D11                     ;UNDEFINED ERROR CODE
.DCX21==:^D21                     ;CI WITH ILLEGAL DESTINATION ADDRESS
.DCX22==:^D22                     ;CC WITH ILLEGAL DESTINATION ADDRESS
.DCX23==:^D23                     ;CI OR CC WITH ZERO SOURCE ADDRESS
.DCX24==:^D24                     ;FLOW CONTROL VIOLATION
.DCX32==:^D32                     ;TOO MANY CONNECTIONS TO NODE
.DCX33==:^D33                     ;TOO MANY CONNECTIONS TO DEST. PROCESS
.DCX34==:^D34                     ;ACCESS NOT PERMITTED
.DCX35==:^D35                     ;LOGICAL LINK SERVICES MISMATCH
.DCX36==:^D36                     ;INVALID ACCOUNT
.DCX37==:^D37                     ;SEGSIZE TOO SMALL
.DCX38==:^D38                     ;PROCESS ABORTED
.DCX39==:^D39                     ;NO PATH TO DESTINATION NODE
.DCX40==:^D40                     ;LINK ABORTED DUE TO DATA LOSS
.DCX41==:^D41                     ;DESTINATION PROCESS DOES NOT EXIST
.DCX42==:^D42                     ;CONFIRMATION IF DI
.DCX43==:^D43                     ;IMAGE DATA FIELD TOO LONG


;EFACT - FACT FILE ENTRY DEFINITIONS

.EFHDR==:0                        ;HEADER WORD
        EF%COD==:777B8            ;ENTRY TYPE CODE
        EF%JOB==:777B17           ;JOB NUMBER
        EF%LIN==:7777B29          ;LINE NUMBER
        EF%SIZ==:77B35            ;ENTRY SIZE
.EFUSR==:1                        ;USER NUMBER WORD
.EFTAD==:2                        ;TIME AND DATE OF ENTRY

; FACT FILE ENTRY TYPE CODES

.EFLGI==:501                      ;LOGIN
.EFLGO==:141                      ;LOGOUT
.EFCAC==:502                      ;CHANGE ACCOUNT
.EFATT==:142                      ;CONSOLE ATTACH
.EFDET==:143                      ;CONSOLE DETACH
.EFCHK==:201                      ;CHECKPOINT
.EFSDU==:540                      ;START DISK-UTILIZATION ENTRIES
.EFDSK==:601                      ;DISK SPACE UTILIZATION
.EFTIM==:741                      ;TIME SET
.EFRES==:740                      ;SYSTEM RESTARTED
.EFLPT==:401                      ;LINE PRINTER USAGE
.EFCDR==:402                      ;CARD READER USAGE
```

```
;ENQ/DEQ BIT DEFINITIONS AND FUNCTION CODES

;FUNCTION CODES

.ENQBL==:0                      ;ENQ BLOCK OPTION
.ENQAA==:1                      ;ENQ ALLOCATE ONLY IF AVAILABLE
.ENQSI==:2                      ;ENQ SOFTWARE INTERRUPT WHEN LOCKED
.ENQMA==:3                      ;ENQ MODIFY ACCESS
.DEQDR==:0                      ;DEQ RESOURCE
.DEQDA==:1                      ;DEQ ALL RESOURCES OF THIS FORK
.DEQID==:2                      ;DEQ THIS ID NUMBER
.ENQCS==:0                      ;ENQC STATUS
.ENQCG==:1                      ;ENQC GET ENQ/DEQ QUOTA FOR A JOB
.ENQCC==:2                      ;ENQC CHANGE ENQ/DEQ QUOTA FOR A JOB
.ENQCD==:3                      ;ENQC DUMP LOCKS AND QUEUE ENTRIES


;BIT DEFINITIONS

EN%SHR==:1B0                    ;SHARABLE REQUEST
EN%BLN==:1B1                    ;BYPASS LEVEL NUMBER
EN%NST==:1B2                    ;ALLOW NESTING
EN%LTL==:1B3                    ;LONG TERM LOCK
EN%LVL==:777B17                 ;LEVEL NUMBER
EN%JOB==:777777B35              ;JOB NUMBER
EN%QCE==:1B0                    ;ERROR CODE IN RH OF STATUS WORD
EN%QCL==:1B0                    ;LOCK DUMP (.ENQCD ONLY)
EN%QCO==:1B1                    ;THIS FORK OWNS THE LOCK
EN%QCQ==:1B2                    ;THIS FORK IS IN THE QUEUE FOR THIS LOCK
EN%QCT==:1B2                    ;LOCK CONTAINS A TEXT STRING
EN%QCX==:1B3                    ;THE LOCK IS LOCKED EXCLUSIVELY
EN%QCB==:1B4                    ;USER IS BLOCKED FOR LOCK

;ENQ/DEQ ARGUMENT BLOCK DATA STRUCTURE

.ENQLN==:0                      ;# OF LOCKS ,, LENGTH OF ARGUMENT BLOCK
        .ENHLN==:77B5           ;LENGTH OF HEADER AREA
        .ENNLK==:7777B17        ;NUMBER OF LOCKS
        .ENALN==:777777B35      ;LENGTH OF ARGUMENT BLOCK
.ENQID==:1                      ;PSI CHANNEL # ,, REQUEST ID
.ENQLV==:2                      ;FLAGS & LEVEL NUMBER ,, JFN, -1, -2, OR -3
.ENQUC==:3                      ;STRING POINTER OR USER CODE
.ENQRS==:4                      ;# OF RESOURCES IN POOL ,, # OF RESOURCES WANT
.ENQMS==:5                      ;ADDRESS OF RESOURCE BLOCK
```

```
;ENQC DUMP DATA STRUCTURE

.ENQDF==:0                   ;FLAGS + LEVEL # ,, OFN, 400000+JOB #, -2, OR -3
                             ;OR:  FLAGS + PSI # ,, JOB # OF Q-ENTRY CREATOR

.ENQDR==:1                   ;TOTAL RESOURCES IN POOL ,, RESOURCES REMAINING
.ENQDT==:2                   ;TIME STAMP OF LAST REQUEST LOCKED
.ENQDC==:3                   ;USER CODE OF LOCK OR START OF TEXT STRING

.ENQDI==:1                   ;GROUP # OR # REQUESTED ,, ENQ ID


;FLOUT/DFOUT
;FORMAT CONTROL WORD

FL%SGN==:3B1                 ;FIRST FIELD SIGN CONTROL
 .FLDIG==:0                  ;DIGIT
 .FLSPC==:1                  ;SPACE
 .FLPLS==:2                  ;PLUS SIGN
 .FLSPA==:3                  ;SPACE
FL%JUS==:3B3                 ;FIRST FIELD JUSTIFICATION CONTROL
 .FLLSP==:0                  ;LEADING SPACES
 .FLLZR==:1                  ;LEADING ZEROS
 .FLLAS==:2                  ;LEADING ASTERISKS
 .FLTSP==:3                  ;TRAILING SPACES
FL%ONE==:1B4                 ;FIRST FIELD NONBLANK
FL%DOL==:1B5                 ;DOLLAR SIGN PREFIX
FL%PNT==:1B6                 ;DECIMAL POINT
FL%EXP==:3B8                 ;THIRD FIELD EXPONENT CONTROL
 .FLEXN==:0                  ;NO EXPONENT
 .FLEXE==:1                  ;E EXPONENT PREFIX
 .FLEXD==:2                  ;D EXPONENT PREFIX
 .FLEXM==:3                  ;*10^ EXPONENT PREFIX
FL%ESG==:3B10                ;EXPONENT SIGN CONTROL
 .FLDGE==:0                  ;DIGIT
 .FLPLE==:1                  ;PLUS SIGN
 .FLSPE==:2                  ;SPACE
 .FLDGT==:3                  ;DIGIT
FL%OVL==:1B11                ;COLUMN OVERFLOW
FL%RND==:37B17               ;DIGIT POSITION FOR ROUNDING
FL%FST==:77B23               ;FIRST FIELD WIDTH
FL%SND==:77B29               ;SECOND FIELD WIDTH
FL%THD==:77B35               ;THIRD FIELD WIDTH
```

```
;GDSTS

;SEE MTOPR FOR CARD READER AND LINE PRINTER STATUS BITS
;SEE GENERAL FIELD AND VALUE SECTION FOR MAGTAPE STATUS BITS
;SEE TOPS20AN SECTION FOR NETWORK STATUS BITS

.GDFSM==:17B3           ;TOPS20AN ;FINITE MACHINE STATE

;TTY BITS

GD%PAR==:1B35                     ;IF ON, TERMINAL ACCEPTS PARITY


;GET

        ;Argument block for GET:

        .GFLAG==:0                ;FLAG WORD
                GT%LOW==:1B0      ;USE LOW ADDRESS IN .GLOW
                GT%HGH==:1B1      ;USE HIGH ADDRESS IN .GHIGH
                GT%BAS==:1B2      ;USE BASE SECTION IN .GBASE
                GT%CCH==:1B3      ;CLEAR PROGRAM CACHE
                GT%CSH==:1B4      ;CACHE THIS PROGRAM
                GT%ADR==:1B19     ; (IN AC1) USE ADDRESS LIMITS IN AC2
                GT%PRL==:1B20     ; (IN AC1) PRELOAD PAGES
                GT%NOV==:1B21     ; (IN AC1) DON'T OVERLAY EXISTING PAGES
                GT%ARG==:1B22     ; (IN AC1) IF ON, AC2 CONTAINS ADDRESS
                                  ; OF ARG BLOCK
                GT%JFN==:7777B35 ; (IN AC1) JFN

        .GLOW==:1                 ;LOW ADDRESS IF GT%LOW ON
        .GHIGH==:2                ;HIGH ADDRESS IG GT%HGH ON
        .GBASE==:3                ;BASE IF GT%BAS ON
```

```
;GETAB - TABLE INDICES

.JOBTT==:0                        ;JOB NUMBER TO TTY NUMBER
.JOBRT==:1                        ;JOB RUNTIME
.TICKP==:2                        ;TICKS PER SECOND
.JOBDI==:3                        ;JOB NUMBER TO DIRECTORY NUMBERS (OBS)
.TTYJO==:4                        ;TTY NUMBER TO JOB NUMBER
.NCPGS==:5                        ;NUMBER PHYSICAL CORE PAGES
.DEVNA==:6                        ;DEVICE NAME
.DEVCH==:7                        ;DEVICE CHARACTERISTICS
.DEVUN==:10                       ;DEVICE UNIT NUMBERS
.DSKER==:11                       ;DISK ERROR WORDS
.DRMER==:12                       ;DRUM ERROR WORDS
.SYSVE==:13                       ;VERSION TEXT
.SYSTA==:14                       ;STATISTICS
.QTIME==:15                       ;SCHED QUEUE TIMES
.JOBNA==:16                       ;JOB NUMBER TO PROGRAM NAME
.SNAME==:17                       ;SUBSYSTEM NAME
.STIME==:20                       ; " TIME
.SPFLT==:21                       ; " PAGE FAULTS
.SSIZE==:22                       ; " SIZE INTEGRAL
.SNBLK==:23                       ; " NUMBER WAKEUPS
.DBUGS==:24                       ;DBUGSW, DCHKSW
.LOGDE==:25                       ;LOG, JOB 0 DESIGNATORS
.PTYPA==:26                       ;PTY PARAMETERS
.SYMTA==:27                       ;GTTAB SYMBOL TABLE
.DWNTI==:30                       ;HSYS VARIABLES
.JOBPN==:31                       ;JOB NUMBER TO PROGRAM NAME
.BLDTD==:32                       ;MONITOR BUILD TIME AND DATE
.LSTDR==:33                       ;LAST DIR NUMBER ASSIGNED (OBS)
.APRID==:34                       ;APR SERIAL NUMBER
.HQLAV==:35                       ;HIGH QUEUE LOAD AVERAGES
.LQLAV==:36                       ;LOW QUEUE LOAD AVERAGES
.NETRD==:37             ;TOPS20AN ;ARPANET STATUS
.IMPHR==:40             ;TOPS20AN ;HOST READY
.HSTST==:41             ;TOPS20AN ;DEAD HOST STATUS
.HSTNA==:42             ;TOPS20AN ;HOST NAMES
.HOSTN==:43             ;TOPS20AN ;HOST NAME INDEX
.NETLS==:44             ;TOPS20AN ;LOCAL SOCKET
.NETFS==:45             ;TOPS20AN ;FOREIGN SOCKET
.NETAW==:46             ;TOPS20AN ;ARPA CONNECTION ADDRESS
.NETBA==:47             ;TOPS20AN ;BIT ALLOCATION
.NETST==:50             ;TOPS20AN ;CONNECTION STATUS
.NETBU==:51             ;TOPS20AN ;ARPANET BUFFERS
.NETBT==:52             ;TOPS20AN ;BYTE COUNT STATISTICS
.IMPL1==:53             ;TOPS20AN ;IMP LINK TABLE ONE
.IMPL2==:54             ;TOPS20AN ;IMP LINK TABLE TWO
.IMPL3==:55             ;TOPS20AN ;IMP LINK TABLE THREE
.IMPL4==:56             ;TOPS20AN ;IMP LINK TABLE FOUR
.LHOST==:57             ;TOPS20AN ;LOCAL HOST NUMBER
.JBONT==:60                       ;OWNING JOB
.NSWPG==:61                       ;DEFAULT SWAPPING PAGES
.SCOUN==:62                       ;COUNT SNAMES TABLE
```

```
;GETJI

.JIJNO==:0                      ;JOB NUMBER
.JITNO==:1                      ;TERMINAL NUMBER
.JIUNO==:2                      ;USER NUMBER
.JIDNO==:3                      ;DIRECTORY NUMBER
.JISNM==:4                      ;SUBSYS NAME
.JIPNM==:5                      ;PROGRAM NAME
.JIRT==:6                       ;RUN TIME
.JICPJ==:7                      ;CONTROLLING PTY JOB NUMBER
.JIRTL==:10                     ;RUN TIME LIMIT (SET BY TIMER JSYS)
.JIBAT==:11                     ;CONTROLLED BY BATCH
.JIDEN==:12                     ;MAGTAPE DEFAULT DENSITY
.JIPAR==:13                     ;MAGTAPE DEFAULT PARITY
.JIDM==:14                      ;MAGTAPE DEFAULT DATA MODE
.JIRS==:15                      ;MAGTAPE DEFAULT RECORD SIZE
.JIDFS==:16                     ;DEFERRED SPOOLING
.JILNO==:17                     ;LOGGED-IN DIRECTORY NUMBER
.JISRM==:20                     ;POINTER TO JOB SESSION REMARK
.JILLN==:21                     ;LAST LOGIN DATE & TIME
.JISRT==:22                     ;JOB RUNTIME AT START OF THIS ACCOUNTING SESSI
.JISCT==:23                     ;JOB CONSOLE TIME AT START OF THIS SESSION
.JIT20==:24                     ;-1 IF AT TOPS20 COMMAND LEVEL
.JISTM==:25                     ;DATE & TIME JOB WAS INITIALIZED
.JIBCH==:26                     ;BATCH STREAM AND FLAGS
        OB%WTO==3B1             ;WRITE TO OPERATOR CAPABILITIES
            .OBALL==0           ;WTO AND WTOR ALLOWED
            .OBNWR==1           ;NO WTOR ALLOWED
            .OBNOM==2           ;NO MESSAGE ALLOWED
        OB%BSS==1B10            ;BATCH STREAM NUMBER SET
        OB%BSN==177B17          ;BATCH-STREAM NUMBER
.JILLO==:27                     ;LOGICAL LOCATION (NODE NAME)

.JIMAX==:.JILLO                 ;CURRENT HIGHEST GETJI OFFSET


;GFRKS

GF%GFH==:1B0                    ;GET RELATIVE FORK HANDLES
GF%GFS==:1B1                    ;GET FORK STATUS


;GFUST

.GFAUT==:0                      ;GET FILE AUTHOR
.GFLWR==:1                      ;GET FILE LAST WRITER

;GTHST                          ;TOPS20AN

.GTHSZ==:0                      ;HOST TABLE SIZES
.GTHIX==:1                      ;INDEX TO STRING CONVERSION
.GTHNS==:2                      ;NUMBER TO STRING CONVERSION
.GTHSN==:3                      ;STRING TO NUMBER CONVERSION
.GTHHN==:4                      ;HOST NUMBER TO STATUS
.GTHHI==:5                      ;HOST INDEX TO STATUS
```

```
;GETOK DEFINITIONS
.GOASD==:1                        ;ASSIGN DEVICE
  .GEERB==:0                      ;ERROR BLOCK ADDRESS
  .GEADD==:1                      ;DEVICE DESIGNATOR
.GOCAP==:2                        ;ENABLE CAPABILITIES
  .GENCP==:1                      ;NEW CAPABILITIES
.GOCJB==:3                        ;ALLOW CRJOB JSYS
.GOLOG==:4                        ;ALLOW LOGINS
  .GELUN==:1                      ;USER NUMBER
.GOCFK==:5                        ;ALLOW CFORK JSYS
  .GEFCT==:1                      ;NUMBER OF FORKS
.GOTBR==:6                        ;ALLOW SET TERMINAL BAUD RATE
  .GELIN==:1                      ;LINE NUMBER
  .GESPD==:2                      ;SPEED
.GOLGO==:7                        ;ALLOW LOGOUT
  .GEUSD==:1                      ;PAGES USED
  .GEQUO==:2                      ;QUOTA
  .GERLG==:3                      ;JOB TO BE LOGGED OUT, -1 FOR CALLER
.GOENQ==:10                       ;ALLOW SET ENQ QUOTA
  .GEEQU==:1                      ;DESIRED QUOTA
  .GEEUN==:2                      ;JOB NUMBER
.GOCRD==:11                       ;ALLOW CREDIR
.GOSMT==:12                       ;ALLOW SMOUNT
  .GESDE==:1                      ;DEVICE DESIGNATOR

.GOMDD==:13                       ;ALLOW MDDT ENTRY
.GOCLS==:14                       ;VERIFY CLASS ASSIGNMENT FOR A JOB
   .GEJOB==:1                     ;JOB #
   .GECLS==:2                     ;CLASS DESIRED
.GOCL0==:15                       ;SET CLASS AT LOGIN
.GOMTA==:16                       ;MT ACCESS REQUEST

   .GEACC==:1                     ;ACCESS CODE FROM HDR1
   .GEUSN==:2                     ;USER NUMBER
   .GEUNT==:3                     ;MT UNIT NUMBER
   .GEACD==:4                     ;DESIRED ACCESS (BITS)
   .GELTP==:5                     ;LABEL TYPE
.GOACC==:17                       ;ACCESS AND CONNECT
        .GOAC0==:1                ;FLAGS FROM ACESS JSYS
        .GOAC1==:2                ;DIRECTORY NUMBER
.GOOAD==:20                       ;ASSIGN DUE TO OPENF
                                  ;.GEADD IS THE ARG OFFSET FOR THE
                                  ; DEVICE DESIGNATOR
.GODNA==:21                       ;ACCESS TO DECNET
.GOANA==:22                       ;ACCESS TO ARPANET

.GOATJ==:23                       ;ATACH JSYS
        .GOTJB==:1                ;TAGET JOB NUMBER
        .GOTTY==:2                ;SOURCE TTY NUMBER

.GOKMZ==:^D20                     ;MAX ARGUMENT BLOCK SIZE FOR GETOK REQUEST
;ERROR BLOCK ADDRESS OFFSETS

.GESIZ==:0                        ;SIZE OF THIS BLOCK
.GEERN==:1                        ;ERROR NUMBER
.GEPTR==:2                        ;POINTER TO ERROR STRING
.GEBSZ==:3                        ;STRING SIZE
```

```
;GTJFN DEFINITIONS

;FLAGS PROVIDED TO GTJFN ON CALL

GJ%FOU==:1B0                      ;FILE IS FOR OUTPUT USE
GJ%NEW==:1B1                      ;NEW FILE ONLY
GJ%OLD==:1B2                      ;OLD FILE ONLY
GJ%MSG==:1B3                      ;PRINT AN APPROPRIATE MESSAGE
GJ%CFM==:1B4                      ;CONFIRMATION IS REQUIRED
GJ%TMP==:1B5                      ;TEMPORARY
GJ%NS==:1B6                       ;DONT SEARCH SEARCH LISTS
GJ%ACC==:1B7                      ;NO ACCESS BY OTHER FORKS
GJ%DEL==:1B8                      ;IGNORE "DELETED" BIT
GJ%JFN==:3B10                     ;JFN USE FIELD
.GJDNU==:0                        ;DO NOT USE JFN PROVIDED
.GJERR==:2                        ;ERROR IF CANNOT USE JFN PROVIDED
.GJALT==:3                        ;USE ALTERNATE IF CANNOT USE GIVEN JFN
GJ%IFG==:1B11                     ;ACCEPT INPUT FILE GROUP DESCRIPTORS
GJ%OFG==:1B12                     ;ACCEPT OUTPUT FILE GROUP DESCRIPTORS
GJ%FLG==:1B13                     ;RETURN FLAGS
GJ%PHY==:1B14                     ;PHYSICAL DEVICE ONLY
GJ%XTN==:1B15                     ;EXTENDED FORMAT (E+11 EXISTS)
GJ%FNS==:1B16                     ;ACCUMULATOR 2 CONTAINS JOB FILE NUMBERS
GJ%SHT==:1B17                     ;SHORT CALL FORMAT

;FLAGS PROVIDED TO GTJFN (IN SECOND FLAG WORD)

G1%RND==:1B0                      ;RETURN ON NULL(IN ALTERNATE FLAG WORD)
G1%RBF==:1B1                      ;^R BUFFER IS DISJOINT (OBSOLETE)
G1%NLN==:1B2                      ;NO LONG NAMES
G1%RCM==:1B3                      ;RETURN CONFIRM MESSAGE
G1%RIE==:1B4                      ;RETURN WHEN MAIN STRING IS EMPTY
G1%IIN==:1B5                      ; Ignore invisible status
G1%SLN==:1B6                      ;SUPPRESS EXPANSION OF LOGICAL NAMES
```

```
;FLAGS RETURNED BY GTJFN

GJ%DEV==:1B0                    ;ASTERISK WAS GIVEN FOR DEVICE
GJ%UNT==:1B1                    ;ASTERISK WAS GIVEN FOR UNIT
GJ%DIR==:1B2                    ;ASTERISK WAS GIVEN FOR DIRECTORY
GJ%NAM==:1B3                    ;ASTERISK WAS GIVEN FOR NAME
GJ%EXT==:1B4                    ;ASTERISK WAS GIVEN FOR EXTENSION
GJ%VER==:1B5                    ;ASTERISK WAS GIVEN FOR GENERATION
GJ%UHV==:1B6                    ;USE HIGHEST GENERATION
GJ%NHV==:1B7                    ;USE NEXT HIGHER GENERATION
GJ%ULV==:1B8                    ;USE LOWEST GENERATION
GJ%PRO==:1B9                    ;PROTECTION GIVEN
GJ%ACT==:1B10                   ;ACCOUNT GIVEN
GJ%TFS==:1B11                   ;TEMPORARY FILE SPECIFIED (;T)
GJ%GND==:1B12                   ;COMPLEMENT OF GJ%DEL ON CALL
GJ%GIV==:1B17                   ; Comp of G1%IIV

;GTJFN TABLE OFFSETS

.GJGEN==:0                      ;FLAGS ,, GENERATION
.GJDEF==:<Z 0>                  ;DEFAULT GENERATION
.GJNHG==:<Z -1>                 ;NEXT HIGHER GENERATION
.GJLEG==:<Z -2>                 ;LOWEST EXISTING GENERATION
.GJALL==:<Z -3>                 ;ALL GENERATIONS (I.E., ;*)
.GJSRC==:1                      ;SOURCE JFN ,, OUTPUT JFN
.GJDEV==:2                      ;DEFAULT DEVICE
.GJDIR==:3                      ;DEFAULT DIRECTORY
.GJNAM==:4                      ;DEFAULT NAME
.GJEXT==:5                      ;DEFAULT EXTENSTION
.GJPRO==:6                      ;DEFAULT PROTECTION
.GJACT==:7                      ;DEFAULT ACCOUNT
.GJJFN==:10                     ;DESIRED JFN
.GJF2==:11                      ;SECOND GROUP FLAGS,,COUNT
.GJCPP==:12                     ;COPY BUFFER POINTER
.GJCPC==:13                     ;COPY BUFFER COUNT
.GJRTY==:14                     ;RETYPE (^R) POINTER
.GJBFP==:15                     ;TOP OF BUFFER POINTER
.GJATR==:16                     ;POINTER TO ARBITRARY ATTRIBUTE BLOCK
.GJNOD==:17                     ;DEFAULT NODE

;GNJFN - FLAGS RETURNED

GN%STR==:1B13                   ;STRUCTURE CHANGED
GN%DIR==:1B14                   ;DIRECTORY CHANGED
GN%NAM==:1B15                   ;NAME CHANGED
GN%EXT==:1B16                   ;EXTENSION CHANGED

;GTNCP                          ;TOPS20AN

.GTNSZ==:0                      ;SIZE OF TABLE
.GTNIX==:1                      ;NCP INDEX
.GTNNI==:2                      ;NVT INPUT
.GTNNO==:3                      ;NVT OUTPUT
.GTNJF==:4                      ;JFN

.NCIDX==:0                      ;NCP INDEX
.NCFHS==:1                      ;FOREIGN HOST
.NCLSK==:2                      ;LOCAL SOCKET
.NCFSK==:3                      ;FOREIGN SOCKET
.NCFSM==:4                      ;FINITE STATE MACHINE STATE
.NCLNK==:5                      ;LINK
.NCNVT==:6                      ;NVT, -1 IF NOT A TELNET CONNECTION
.NCSIZ==:7                      ;BYTE SIZE OF CONNECTION
```

```
.NCMSG==:10                    ;MSG ALLOC
.NCBAL==:11                    ;BIT ALLOC
.NCDAL==:12                    ;DESIRED ALLOC
.NCBTC==:13                    ;BITS XFERRED
.NCBPB==:14                    ;BYTES/BUFFER
.NCCLK==:15                    ;TIME-OUT COUNTDOWN
.NCSTS==:16                    ;CONNECTION STATUS
```

```
;GTRPW

PF%USR==:1B0                  ;PAGE FAIL WORD - USER MODE REFERENCE
PF%WTF==:1B1                  ; " - WRITE REFERENCE (XGTPW)
PF%WRT==:1B5                  ; " - WRITE REFERENCE
TSW%RD==:1B14                 ;TRAP STATUS WORD - READ
TSW%WT==:1B15                 ; " - WRITE
TSW%WR==:1B15                 ; (ANOTHER NAME FOR ABOVE)
TSW%EX==:1B16                 ; " - EXECUTE
TSW%MN==:1B17                 ; " - MONITOR MODE REFERENCE


;GTSTS BITS RETURNED IN 2

GS%OPN==:1B0                  ;FILE IS OPEN
GS%RDF==:1B1                  ;IF OPEN, FILE IS OPEN FOR READ
GS%WRF==:1B2                  ;IF OPEN, FILE IS OPEN FOR WRITE
GS%XCF==:1B3                  ;IF OPEN, FILE IS OPEN FOR EXECUTE
GS%RND==:1B4                  ;OK TO RESET BYTE POINTER
                              ; (FILE IS NOT APPEND)
GS%APT==:1B5                  ;ACCESS PER PAGE TABLE
                              ; (NOT IMPLEMENTED -- OBSOLETE)
GS%CAL==:1B6                  ;OK TO CALL AS A PROCEDURE
                              ; (NOT IMPLEMENTED -- OBSOLETE)
GS%LNG==:1B7                  ;FILE IS LONG
GS%EOF==:1B8                  ;AT END OF FILE ON READ
GS%ERR==:1B9                  ;FILE MAY BE IN ERROR
GS%NAM==:1B10                 ;FILE HAS A NAME (JFN EXISTS)
GS%AST==:1B11                 ;ONE OR MORE FIELDS OF NAME
                              ; IS WILD
GS%ASG==:1B12                 ;JFN IS BEING ASSIGNED
GS%HLT==:1B13                 ;TERMINATE ON I/O ERROR
GS%FRK==:1B17                 ;JFN IS RESTRICTED TO CREATING FORK
GS%PLN==:1B18                 ;DON'T STRIP LINE NUMBERS ON SIN/BIN
GS%MOD==:17B35                ;DATA MODE
        .GSNRM==:0            ;NORMAL MODE
        .GSSMB==:1            ;SMALL BUFFER MODE (DCN:, SRV:)
        .GSIMG==:10           ;IMAGE (BINARY) MODE
        .GSDMP==:17           ;DUMP MODE
```

```
;HPTIM

.HPELP==:0                       ;ELAPSED TIME
.HPRNT==:1                       ;RUN TIME


;IDCNV (ALSO IDTNC AND ODCNV)

IC%DSA==:1B0                     ;DAYLIGHT SAVINGS IF APPROPRIATE
IC%ADS==:1B1                     ;APPLY DAYLIGHT SAVINGS
IC%UTZ==:1B2                     ;USE TIME ZONE GIVEN
IC%JUD==:1B3                     ;USE JULIAN DATE CONVERSION
IC%TMZ==:77B17                   ;TIME ZONE
IC%TIM==777777B35                ;LOCAL TIME


;IDTIM & IDTNC

IT%NDA==:1B0                     ;NO DATE
IT%NNM==:1B1                     ;NO NUMERIC MONTH
IT%SNM==:1B2                     ;SECOND NUMBER IS MONTH
IT%ERR==:1B3                     ;ERROR IF NUMBERS ARE NOT IN SPECIFIED
                                 ; ORDER
IT%NTI==:1B6                     ;NO TIME
IT%NIS==:1B7                     ;NO SECONDS
IT%AIS==:1B8                     ;ALWAYS INCLUDE SECONDS
IT%NAC==:1B9                     ;NO COLON ALLOWED BETWEEN HH AND MM
IT%AAC==:1B10                    ;ALWAYS ALLOW COLON
IT%AMS==:1B11                    ;ALWAYS INTERPRET ONE COLON AS HHMM:SS
IT%AHM==:1B12                    ;ALWAYS INTERPRET ONE COLON AS HH:MM
IT%N24==:1B14                    ;NO 24-HOUR FORMAT
IT%NTM==:1B15                    ;NO TIME MODIFIER (AM, PM)
IT%NTZ==:1B16                    ;NO TIME ZONE


;.IMOPR - MONITOR ROUTINE USED BY MDDT AND SNOOP.  THIS IS NOT
;A JSYS SO THAT CALLS ARE FAST.

.IMALC==:1                       ;ALLOCATE PAGES FOR USE IN MAPPING SYMBOLS
.IMMAP==:2                       ;MAP PAGES OF THE SYMBOL TABLE
.IMUMP==:3                       ;UNMAP PAGES OF THE SYMBOL TABLE


;INLNM

.INLJB==:0                       ;GET JOB WIDE LOGICAL NAME FROM INDEX
.INLSY==:1                       ;GET SYSTEM LOGICAL NAME FROM INDEX
```

```
;IPCF BIT DEFINITIONS AND DATA STRUCTURES

;PACKET FORMAT

.IPCFL==:0                      ;FLAGS WORD
IP%CFB==:1B0                    ;DON'T BLOCK READ
IP%CFS==:1B1                    ;INDIRECT SENDER'S PID
IP%CFR==:1B2                    ;INDIRECT RECEIVER'S PID
IP%CFO==:1B3                    ;OVERDRAW SEND
IP%TTL==:1B4                    ;TRUNCATE ON TOO LARGE MESSAGE
IP%CPD==:1B5                    ;CREATE A PID ON THE SEND
IP%JWP==:1B6                    ;MAKE THE CREATED PID BE JOB WIDE
IP%NOA==:1B7                    ;NO ACCESS OF PID BY OTHER FORKS
IP%CFP==:1B18                   ;SENDER IS PRIV'D AND IS ENVOKING PRIVS
IP%CFV==:1B19                   ;PAGE TRANSFER MODE
IP%CFZ==:1B20                   ;ZERO LENGTH MESSAGE WAS SENT
IP%INT==:1B21                   ; Internal call - unavailable to users
IP%EPN==:1B22                   ;PAGE NUMBER IS 18 BITS
IP%CFE==:77B29                  ;ERROR FIELD

;ERRORS SENT BY INFO

 .IPCPI==:15                    ;INSUFFICIENT PRIVILEGE
 .IPCUF==:16                    ;ILLEGAL FUNCTION
 .IPCSN==:67                    ;SEND INFO YOUR NAME
 .IPCFF==:72                    ;INFO FREE SPACE EXHAUSTED
 .IPCBP==:74                    ;PID HAS NO NAME OR IS ILLEGAL
 .IPCDN==:75                    ;DUPLICATE NAME
 .IPCNN==:76                    ;UNKNOWN NAME
 .IPCEN==:77                    ;ILLEGAL NAME
 .IPCKM==:66                    ;NOTIFICATION THAT PID HAS BEEN DELETED
IP%CFC==:7B32                   ;SYSTEM SENDER CODE
 .IPCCC==:1                     ;SENT BY [SYSTEM]IPCF
 .IPCCF==:2                     ;SENT BY SYSTEM WIDE [SYSTEM]INFO
 .IPCCP==:3                     ;SENT BY RECEIVER'S [SYSTEM]INFO
IP%CFM==:7B35                   ;SPECIAL MESSAGE RETURN FIELD
 .IPCFN==:1                     ;MESSAGE WAS NOT DELIVERED
.IPCFS==:1                      ;PID OF SENDER
.IPCFR==:2                      ;PID OF RECEIVER
.IPCFP==:3                      ;POINTER TO MESSAGE BLOCK
.IPCFD==:4                      ;LOGGED IN DIR OF SENDER
.IPCFC==:5                      ;ENABLED CAPABILITIES OF SENDER
.IPCSD==:6                      ;CONNECTED DIRECTORY NUMBER OF SENDER
.IPCAS==:7                      ;POINTER TO ACCOUNT STRING OF SENDER
.IPCLL==:10                     ;POINTER TO LOGICAL LOCATION OF SENDER

;Possible values in word 0 of packet data block when received from the system

.IPCSU==:26                     ;SPOOL MESSAGE CODE FROM IPCC
.IPCSL==:27                     ;LOGOUT MESSAGE CODE FROM IPCC
.IPCSA==:30                     ;RESOURCE ALLOCATOR MESSAGE CODE
.IPCDS==:31                     ;STRUCTURE DISMOUNT MESSAGE CODE FROM IPCC
.IPCLI==:32                     ;LOGIN MESSAGE CODE FROM IPCC
.IPCLO==:33                     ;LOGOUT MESSAGE TO CREATOR FROM IPCC
.IPCKP==:34                     ;DELETED PID MESSAGE FROM IPCC
.IPCCA==:35                     ;CREATE AN APPLICATION (RESERVED FOR TPS USE)
.IPCTR==:36                     ;REQUEST FROM TAPE
.IPCMS==:37                     ;STRUCTURE MOUNT MESSAGE CODE FROM IPCC
.IPCRS==:40                     ;STRUCTURE REMOVAL MSSG CODE FROM IPCC
.IPCSR==:41                     ; Archive message code from IPCC

.IPCSS==:15                     ;IPCC REQUEST TO INFO TO DELETE PIDS
```

```
;[SYSTEM] INFO DEFINITIONS

.IPCI0==:0                      ;CODE,,FUNCTION
.IPCIW==:1                      ;FIND PID FOR NAME
.IPCIG==:2                      ;FIND NAME FOR PID
.IPCII==:3                      ;ASSIGN NAME TO PID
.IPCIJ==:4                      ;ASSIGN NAME TO PID
.IPCIK==:5                      ;NOTIFY WHEN SPECIFIED PID IS KILLED
.IPCIS==:15                     ;MONITOR DROP PID FUNCTION
.IPCI1==:1                      ;PID TO GET A COPY OF REPLY
.IPCI2==:2                      ;START OF DATA


;JFNS

JS%DEV==:7B2                    ;DEVICE FIELD OUTPUT CONTROL
JS%DIR==:7B5                    ;DIRECTORY FIELD OUTPUT CONTROL
JS%NAM==:7B8                    ;NAME FIELD OUTPUT CONTROL
JS%TYP==:7B11                   ;FILE TYPE FIELD OUTPUT CONTROL
JS%GEN==:7B14                   ;GENERATION FIELD OUTPUT CONTROL
JS%PRO==:7B17                   ;PROTECTION FIELD OUTPUT CONTROL
JS%ACT==:7B20                   ;ACCOUNT FIELD OUTPUT CONTROL
        ;VALUES FOR ABOVE 7 FIELDS:
        .JSNOF==:0              ;NEVER OUTPUT FIELD
        .JSAOF==:1              ;ALWAYS OUTPUT FIELD
        .JSSSD==:2              ;SUPPRESS IF SYSTEM DEFAULT
JS%TMP==:1B21                   ;RETURN ;T IF TEMP FILE
JS%SIZ==:1B22                   ;RETURN SIZE
JS%CDR==:1B23                   ;RETURN CREATION DATE
JS%LWR==:1B24                   ;RETURN LAST WRITE
JS%LRD==:1B25                   ;RETURN LAST READ
JS%PTR==:1B26                   ;AC 2 HOLDS STRING POINTER NOT JFN
JS%ATR==:1B27                   ;RETURN ATTRIBUTES
JS%AT1==:1B28                   ;RETURN 1 SPECIFIC ATTRIBUTE
JS%OFL==:1B29                   ;RETURN ;OFF-LINE IF OFFLINE FILE
JS%PSD==:1B32                   ;PUNCTUATE SIZE AND DATE
JS%TBR==:1B33                   ;TAB BEFORE FIELDS RETURNED
JS%TBP==:1B34                   ;TAB BEFORE POSSIBLE FIELDS
JS%PAF==:1B35                   ;PUNCTUATE ALL FIELDS

JS%SPC==:<FLD(.JSAOF,JS%DEV)>!<FLD(.JSAOF,JS%DIR)>!<FLD(.JSAOF,JS%NAM)>!^_
<FLD(.JSAOF,JS%TYP)>!<FLD(.JSAOF,JS%GEN)>!JS%PAF      ;MASK FOR WHOLE SPEC

;LNMST

.LNSJB==:0                      ;GET JOB WIDE DEFINITION OF A LN
.LNSSY==:1                      ;GET SYSTEM DEFINITION OF A LOGICAL NAME


;LOCK

LK%CNT==:1B0                    ;USE COUNT IN AC3
LK%PHY==:1B1                    ;USE AC1 AS PHYSICAL PAGE NUMBER
LK%NCH==:1B2                    ;MAP PAGES CACHE INHIBITED
LK%AOL==:1B3                    ;ALLOW LOCKING IN OFFLINE PAGES


;METER JSYS DEFS.

.MEREA==:1                      ;READ EBOX TICKS
.MERMA==:2                      ;READ MBOX TICKS
```

```
;MSTR

.MSRNU==:0                      ;READ STATUS OF NEXT DISK UNIT
.MSRUS==:1                      ;READ STATUS OF A DISK UNIT
  .MSRCH==:0                    ;CHANNEL NUMBER
  .MSRCT==:1                    ;CONTROLLER NUMBER
  .MSRUN==:2                    ;UNIT NUMBER
  .MSRST==:3                    ;STATUS
    MS%MNT==:1B0                ;THIS UNIT IS PART OF A MOUNTED STRUCTURE
    MS%16B==:1B1                ;THIS UNIT WRITTEN IN 16-BIT MODE
                                ; (RESERVED FOR FUTURE)
    MS%DIA==:1B2                ;THIS UNIT IS CURRENTLY IN USE BY AN
                                ;  ON-LINE DIAGNOSTIC
    MS%OFL==:1B3                ;THIS UNIT IS OFF-LINE
    MS%ERR==:1B4                ;THERE WAS AN ERROR READING THIS UNIT
    MS%BBB==:1B5                ;ONE OF THE BAT BLOCKS IS BAD
    MS%HBB==:1B6                ;ONE OF THE HOME BLOCKS IS BAD
    MS%WLK==:1B7                ;UNIT IS WRITE-LOCKED
    MS%TYP==:777B17             ;DISK TYPE CODE
; DEFINED THE SAME AS .UTTXX IN PHYPAR
     .MSRP4==:1                 ;RP04
     .MSRP5==:5                 ;RP05
     .MSRP6==:6                 ;RP06
     .MSRP7==:7                 ;RP07
     .MSRM3==:11                ;RM03
     .MSR20==:24                ;RP20
  .MSRSN==:4                    ;STRUCTURE NAME
  .MSRSA==:5                    ;STRUCTURE ALIAS
  .MSRNS==:6                    ;UNIT # IN STRUCTURE,,# OF UNITS IN STRUCTURE
  .MSRSW==:7                    ;NUMBER OF PAGES FOR SWAPPING
  .MSRUI==:10                   ;UNIT ID
  .MSROI==:13                   ;OWNER ID
  .MSRFI==:16                   ;FILE-SYSTEM ID
  .MSRSP==:21                   ;NUMBER OF SECTORS PER PAGE
  .MSRSC==:22                   ;NUMBER OF SECTORS PER CYLINDER
  .MSRPC==:23                   ;NUMBER OF PAGES PER CYLINDER
  .MSRCU==:24                   ;NUMBER OF CYLINDERS PER UNIT
  .MSRSU==:25                   ;NUMBER OF SECTORS PER UNIT
  .MSRBT==:26                   ;NUMBER OF BIT-WORDS IN BIT TABLE PER CYLINDER
  .MSRSE==:27                   ;CPU SERIAL # IF STRUCTURE IS USED FOR BOOTING
  .MSRLN==:30                   ;MAX LENGTH OF ARGUMENT BLOCK IN WORDS
```

```
.MSMNT==:2                      ;MOUNT A STRUCTURE
  .MSTNM==:0                    ;NAME OF STRUCTURE
  .MSTAL==:1                    ;ALIAS NAME
  .MSTNU==:2                    ;NUMBER OF UNITS IN STRUCTURE
  .MSTFL==:2                    ;FLAGS (LHS)
    MS%FLG==:777777,,0          ;MASK FOR .MSTFL
    MS%NFH==:1B0                ;NO FIX BAD HOME BLOCK
    MS%NFB==:1B1                ;NO FIX BAD BAT BLOCK
    MS%XCL==:1B2                ;MOUNT FOR EXCLUSIVE USE BY JOB
    MS%IGN==:1B3                ;IGNORE ERRORS
  .MSTUI==:3                    ;START OF UNIT INFORMATION
    .MSTCH==:0                  ;CHANNEL NUMBER
    .MSTCT==:1                  ;CONTROLLER NUMBER
    .MSTUN==:2                  ;UNIT NUMBER
    .MSTNO==:3                  ;# OF ARGUMENT WORDS/UNIT

.MSDIS==:3                      ;DISMOUNT A STRUCTURE
  .MSDNM==:0                    ;NAME OF STRUCTURE

.MSGSS==:4                      ;GET STATUS OF A STRUCTURE
  .MSGSN==:0                    ;STRUCTURE NAME (ALIAS)
  .MSGST==:1                    ;STATUS
    MS%PS==:1B0                 ;STRUCTURE IS A PUBLIC STRUCTURE
    MS%DIS==:1B1                ;STRUCTURE IS BEING DISMOUNTED
    MS%DOM==:1B2                ;STRUCTURE IS DOMESTIC
    MS%PPS==:1B3                ;STRUCTURE IS THE PRIMARY PUBLIC STRUCTURE
    MS%INI==:1B4                ;STRUCTURE IS BEING INITIALIZED
    MS%LIM==:1B5                ;STRUCTURE LIMITED TO 2050 SIZES
    MS%NRS==:1B6                ;STRUCTURE IS NOT REGULATED
    MS%RWS==:1B7                ;READ AFTER WRITE FOR SWAP SPACE
    MS%RWD==:1B8                ;READ AFTER WRITE FOR DATA SPACE
  .MSGNU==:2                    ;NUMBER OF UNITS IN STRUCTURE
  .MSGMC==:3                    ;MOUNT COUNT
  .MSGFC==:4                    ;OPEN FILE COUNT
  .MSGSI==:5                    ;STRUCTURE ID
  .MSGLN==:6                    ;LENGTH OF ARGUMENT BLOCK

.MSSSS==:5                      ;SET STATUS OF A STRUCTURE
  .MSSSN==:0                    ;STRUCTURE NAME
  .MSSST==:1                    ;NEW STATUS BITS
  .MSSMW==:2                    ;MASK WORD OF BITS TO BE CHANGED
  .MSSLN==:3                    ;LENGTH OF ARGUMENT BLOCK
```

```
.MSINI==:6                        ;INITIALIZE A STRUCTURE
  .MSINM==:0                      ;NAME OF STRUCTURE
  .MSIAL==:1                      ;ALIAS NAME
  .MSINU==:2                      ;NUMBER OF UNITS IN STRUCTURE
  .MSIFL==:2                      ;FLAGS (LHS)
                                  ;FLAGS DEFINED IN .MSMNT FUNCTION
    MS%FCN==:77B17                ;FUNCTION CODE
      .MSCRE==:1                  ;CREATE NEW FILE SYSTEM
      .MSRRD==:2                  ;RECONSTRUCT THE ROOT-DIRECTORY
      .MSWHB==:3                  ;WRITE THE HOME BLOCKS
      .MSRIX==:4                          ;REBUILD INDEX TABLE (IDXFIL)
  .MSISU==:3                      ;START OF UNIT INFORMATION
    .MSICH==:0                    ;CHANNEL NUMBER
    .MSICT==:1                    ;CONTROLLER NUMBER
    .MSIUN==:2                    ;UNIT NUMBER
    .MSINO==:3                    ;# OF ARGUMENT WORDS/UNIT
  .MSIST==:6                      ;STATUS WORD
  .MSISW==:7                      ;NUMBER OF PAGES FOR SWAPPING ON THIS UNIT
  .MSIFE==:10                     ;NUMBER OF PAGES FOR FRONT-END FILE SYSTEM
  .MSIUI==:11                     ;UNIT ID
  .MSIOI==:14                     ;OWNER ID
  .MSIFI==:17                     ;FILE SYSTEM ID
  .MSIFB==:22                     ;NUMBER OF PAGES FOR  BOOTSTRAP.BIN (OPTIONAL)
  .MSISN==:23                     ;CPU SERIAL # IF STRUCTURE IS USED FOR BOOTING

.MSIMC==:7                        ;INCREMENT MOUNT COUNT
.MSDMC==:10                       ;DECREMENT MOUNT COUNT
  .MSDEV==:0                      ;DEVICE DESIGNATOR OR STRUCTURE
  .MSJOB==:1                      ;JOB NUMBER FOR WHICH TO CHANGE COUNT
```

```
.MSGSU==:11                         ;GET STRUCTURE USERS
  .MSUAL==:0                        ;POINTER TO ALIAS OF STRUCTURE
  .MSUFL==:1                        ;FLAGS,,# OF ITEMS RETURNED
    MS%GTA==:1B0                    ;GET USERS WHO HAVE ACCESSED STRUCTURE
    MS%GTM==:1B1                    ;GET USERS WHO HAVE MOUNTED STRUCTURE
    MS%GTC==:1B2                    ;GET USERS WHO ARE CONNECTED TO STRUCTURE
  .MSUJ1==:2                        ;FIRST JOB NUMBER RETURNED

.MSHOM==:12                         ;MODIFY HOMEBLOCK WORD
  .MSHNM==:0                        ;POINTER TO ALIAS, OR DESIGNATOR FOR ALIAS
  .MSHOF==:1                        ;OFFSET INTO HOMEBLOCK OF WORD BEING CHANGED
  .MSHVL==:2                        ;NEW VALUES FOR BITS BEING CHANGED
  .MSHMK==:3                        ;MASK DECLARING WHICH BITS BEING CHANGED

.MSICF==:13                         ;INCREMENT MOUNT COUNT ON A FORK BASIS
.MSDCF==:14                         ;DECREMENT MOUNT COUNT ON A FORK BASIS
  .MSDEV==:0                        ;DEVICE DESIGNATOR OR STRUCTURE

.MSOFL==:15                         ;ENABLE PSI INTERRUPTS INTERRUPTS FOR
                                    ; DISK (FOR DEVICE ALLOCATOR)
  .MSCHN==:0                        ;CHANNEL ON WHICH TO RECEIVE INTERRUPT


.MSIIC==:16                         ;IGNORE INCREMENT CHECK FOR STRUCTURE USE
```

```
;MTOPR - FUNCTION CODES

.MOCLE==:0                      ;CLEAR ERRORS
.MONOP==:31                     ;NOP (WAIT FOR ACTIVITY TO STOP)
.MOREW==:1                      ;REWIND
.MOEOF==:3                      ;WRITE EOF
.MODTE==:4                      ;ASSIGN FE DEVICE TO A DTE
.MOFWR==:6                      ;FORWARD SPACE RECORD
.MOBKR==:7                      ;BACKSPACE RECORD
.MORUL==:11                     ;REWIND AND UNLOAD
.MOERS==:13                     ;ERASE TAPE
.MOFWF==:16                     ;FORWARD SPACE FILE
.MOBKF==:17                     ;BACKSPACE FILE
.MOSPD==:26                     ;SET TTY SPEED (FOR KL ONLY)
.MORSP==:27                     ;READ LINE SPEED (FOR KL ONLY)
   MO%RMT==:1B0                 ;FLAG TO SAY LINE IS REMOTE
   MO%AUT==:1B1                 ;FLAG TO SAY LINE IS "AUTO" SPEED
                                ; (RSX20F ONLY)
.MOSDR==:2                      ;SET READ DIRECTION
.MORDR==:26                     ;READ READ DIRECTION
.MOEOT==:10                     ;SKIP TO LOGICAL END OF TAPE
.MOSRS==:5                      ;SET RECORD SIZE
.MORRS==:15                     ;READ RECORD SIZE
.MOSDN==:24                     ;SET DENSITY
.MORDN==:12                     ;READ DENSITY
.MOSDM==:4                      ;SET DATA MODE
.MORDM==:14                     ;READ DATA MODE
.MOSPR==:20                     ;SET PARITY
.MORPR==:21                     ;READ PARITY
.MONRB==:22                     ;GET NUMBER OF REMAINING BYTES IN RECORD
.MOFOU==:23                     ;FORCE OUT RECORD
.MOINF==:25                     ;GET INFORMATION ABOUT TAPE
   .MOICT==:0                   ;COUNT OF ARGUMENTS TO BE RETURNED
   .MOITP==:1                   ;MAGTAPE TYPE CODE
; DEFINED THE SAME AS .UTTXX IN PHYPAR
      .MTT45==:3                ;MAGTAPE TYPE TU45
      .MTT77==:13               ;MAGTAPE TYPE TU77
      .MTT78==:15               ;MAGTAPE TYPE TU78
      .MTT70==:17               ;MAGTAPE TYPE TU70
      .MTT71==:20               ;MAGTAPE TYPE TU71
      .MTT72==:21               ;MAGTAPE TYPE TU72
      .MTT73==:22               ;RESERVED FOR 200 IPS STC GCR DRIVE
   .MOIID==:2                   ;MAGTAPE REEL ID
   .MOISN==:3                   ;CHAN,CONTROLLER,UNIT ,, SERIAL #
   .MOIRD==:4                   ;# OF READS DONE
   .MOIWT==:5                   ;# OF WRITES DONE
   .MOIRC==:6                   ;RECORD # FROM BOT
   .MOIFC==:7                   ;FILE COUNT ON TAPE
   .MOISR==:10                  ;# OF SOFT READ ERRORS
   .MOISW==:11                  ;# OF SOFT WRITE ERRORS
   .MOIHR==:12                  ;# OF HARD READ ERRORS
   .MOIHW==:13                  ;# OF HARD WRITE ERRORS
   .MOIRF==:14                  ;# RECORDS READ
   .MOIWF==:15                  ;# OF FRAMES WRITTEN
.MOLOC==:32                     ;ATTACH MT TO MTA
        .MOCNT==:0              ;OFFSET FOR COUNT
        .MOMTN==:1              ;OFFSET FOR MT NUMBER
        .MOLBT==:2              ;LABEL TYPE (.LTxxx)
        .MODNS==:3              ;DENSITY (.SJDxx)
        .MOAVL==:4              ;ADDRESS OF VOLUME LABELS
        .MONVL==:5              ;# OF VOLUME LABELS (VOL1 + UVLSs)
        .MOCVN==:6              ;CURRENT VOLUME NUMBER WITHIN SET
        .MOVSN==:7              ;VOLUME SET NAME
```

```
.MOSTA==:37                         ;CURRENT MAGTAPE STATUS
  .MODDN==:1                        ;1ST WORD OF .MOSTA DENSITIES CAPABLE

    SJ%CP2==:1B1                    ;200 BPI
    SJ%CP5==:1B2                    ;556 BPI
    SJ%CP8==:1B3                    ;800 BPI
    SJ%C16==:1B4                    ;1600 BPI
    SJ%C62==:1B5                    ;6250 BPI

  .MODDM==:2                        ;2ND WORD OF .MOSTA DATA MODES CAPABLE
    SJ%CMC==:1B1                    ;CORE DUMP MODE
    SJ%CM6==:1B2                    ;SIXBIT
    SJ%CMA==:1B3                    ;ANSI ASCII
    SJ%CM8==:1B4                    ;INDUSTRY COMPATABLE
    SJ%CMH==:1B5                    ;HIGH DENSITY MODE
  .MOTRK==:3                        ;3RD WORD OF .MOSTA NUMBER OF TRACKS
    SJ%7TR==:1B1                    ;7 TRACK DRIVE
    SJ%9TR==:1B2                    ;9 TRACK DRIVE

  .MOCST==:4                        ;4TH WORD OF .MOSTA TAPE STATUS
    SJ%OFS==:1B0                    ;OFF LINE
    SJ%MAI==:1B1                    ;MAINTENANCE MODE ENABLED
    SJ%MRQ==:1B2                    ;MAINTENANCE MODE REQUESTED
    SJ%BOT==:1B3                    ;BOT
    SJ%REW==:1B4                    ;REWINDING
    SJ%WLK==:1B5                    ;WRITE LOCKED

  .MODVT==:5                        ;5TH WORD OF .MOSTA DEVICE TYPE

; DEFINITIONS FOR THIS ARE SAME AS USED IN .MTALN
.MOOFL==:40                         ;PSI FOR MAGTAPES
.MOPST==:42                         ;PSI FOR EOT ON MT'S
;       T3/ PSI ASSIGNMENT (-1 => CLEAR)
.MORVS==:.MOREW                     ;REWIND VOLUME SET
.MORVL==:43                         ;REWIND CURRENT VOLUME
.MOVLS==:44                         ;VOLUME SWITCH FOR UNLABELED TAPES
.MONTR==:45                         ;SET/CLEAR NO TRANSLATE FLAG
;       T3/ -1 => DON'T CONVERT EBCDIC TO ASCII
;       T3/0=> CONVERT
.MORDL==:46                         ;READ USER LABELS
;       T2/ GETS LABEL I.D.
;       T3/ SP TO WHERE 76 CHARCTERS ARE TO BE PLACED
.MOWUL==:47                         ;WRITE USER LABELS
;       T2/ LABEL I.D.
;       T3/ SP TO 76 CHARACTERS OF DATA
.MORLI==:50                         ;READ LABEL INFORMATION FOR MT
        .MOMTP==:1                  ;TYPE OF LABEL
        .MOMVN==:2                  ;VOLUME NAME
        .MOMOW==:3                  ;OWNER
        .MOMFM==:4                  ;FORMAT OF TAPE FILE
        .MOMRL==:5                  ;RECORD LENGTH
        .MOMBL==:6                  ;BLOCK LENGTH
        .MOMCD==:7                  ;CREATION DATE
        .MOMED==:10                 ;EXPIRATION DATE
        .MOMFI==:11                 ;FILE NAME
        .MOMGN==:12                 ;GENERATION NUMBER
        .MOMGV==:13                 ;GENERATION VERSION NUMBER
        .MOVMB==:14                 ;VALUE OF MODE BYTE
.MOSMV==:51                         ;SET MODE VALUE
        .TPFST==:0                  ;STREAM MODE
        .TPFCP==:1                  ;ALL FORMATTING CONTROLS PRESENT
        .TPFFC==:2                  ;FORTRAN CONTROLS PRESENT
        .TPFNC==:3                  ;NO CONTROLS PRESENT
```

```
        .TPFMX==:3                  ;MAX VALUE OF FIELD
.MOSDS==:52                         ;SET DEFERRED VOLUME-SWITCH MODE
.MOPSI==:27                         ;SET ERROR PSI FOR LPT AND CDR
   MO%MSG==:1B0                     ;SUPPRESS STANDARD CTY MESSAGES
.MOSID==:27                         ;SET REEL I.D.
.MOIEL==:30                         ;INHIBIT ERROR LOGGING
.MOSHV==:45                         ;SET HDR1 AND HDR2 VALUES FOR MT
        .MOFMT==:1                  ;OFFSET FOR FORMAT
        .MOEPD==:2                  ;EXPIRATION DATE
        .MOBSZ==:3                  ;BLOCK SIZE
        .MORSZ==:4                  ;RECORD SIZE

;DEF FOR IPCF MESSAGE SENT ON A VOLUME SWITCH OR OTHER CONDITION
;MESSAGE CODE IS .IPCTR. OFFSETS THAT FOLLOW ARE
;RELATIVE TO WORD CONTAINING .IPCTR.

.VMCOD==:0                          ;CODE FOR THIS MESSAGE
                                    ; IS SUBCODE OF .IPCTR FUNCTION
        .VMABT==:1                  ;ABORT CLOSE
        .VMICN==:2                  ;INTERNAL ERROR (HOPEFULLY NOT USED)
        .VMERR==:3                  ;LABEL R/W ERROR
        .VMVSM==:4                  ;VOLUME SWITCH
        .VMSTS==:5                  ;UNIT STATUS CHANGE (NOT USED YET)
        .VMUNL==:6                  ;UNIT UNLOAD
        .VMREW==:7                  ;REWIND
.VSMTN==:1                          ;MT NUMBER
.VSFLG==:2                          ;FLAGS
 VS%FLG==:-1B17                     ;FLAGS PART OF WORD
        VS%WRT==:1B0                ;WRITE PREVIOUS VOLUME WAS OPENED FOR WRITE
 VS%COD==:777777                    ;CODE
        .VSMNV==:1                  ;MOUNT NTH VOLUME
        .VSFST==:2                  ;MOUNT FIRST VOLUME
        .VSLST==:3                  ;MOUNT LAST VOLUME
        .VSMRV==:4                  ;MOUNT RELATIVE VOLUME NUMBER (SIGNED)
        .VSFLS==:5                  ;FORCE LABELED TAPE VOLUME-SWITCH
.VSCNT==:3                          ;VOLUME NUMBER (SIGNED IF VS%MRV IS ON)
```

```
.MOLVF==:32                       ;LOAD DEVICE'S VFU
.MORVF==:33                       ;READ VFU FILE NAME
.MOLTR==:34                       ;LOAD TRANSLATION RAM
.MORTR==:35                       ;READ RAM FILE NAME
.MOSTS==:36                       ;SET SOFTWARE STATUS
.MORST==:37                       ;READ SOFTWARE STATUS
   MO%LPC==1                      ;PAGE COUNTER OVERFLOW
   MO%LCI==2                      ;CHARACTER INTERRUPT (HARD ERROR)
   MO%LVF==4                      ;VFU ERROR.  PAPER MUST BE RE-ALIGNED
   MO%LVU==20                     ;LINE PRINTER HAS OPTICAL VFU
   MO%RPE==40                     ;RAM PARITY ERROR

   MO%RCK==:1                     ;READ CHECK
   MO%PCK==:2                     ;PICK CHECK
   MO%SCK==:4                     ;STACK CHECK
   MO%HEM==:10                    ;HOPPER EMPTY
   MO%SFL==:20                    ;STACKER FULL

   MO%FNX==:1B17                  ;NON-EXISTENT DEVICE
   MO%OL==:1B16                   ;DEVICE IS OFF-LINE
   MO%HE==:1B15                   ;HARDWARE ERROR
   MO%SER==:1B14                  ;SOFTWARE ERROR
   MO%IOP==:1B13                  ;I/O IN PROGRESS
   MO%EOF==:1B12                  ;END OF FILE
;  1B11                           ;RESERVED
   MO%FER==:1B10                  ;FATAL ERROR
   MO%LCP==:1B0                   ;LOWER CASE PRINTER
   MO%RLD==:1B1                   ;FRONT-END WAS RELOADED
.MOFLO==:40                       ;FLUSH OUTPUT

;SEE SETJB FOR VARIOUS ARGUMENT VALUES

.MOSNT==:34                       ;SET TTY NON-TERMINAL STATUS
   .MOSMN==:1                     ;NO SYSTEM MESSAGES(I.E. SUPPRESS)
   .MOSMY==:0                     ;YES SYSTEM MESSAGES(DEFAULT)
.MORNT==:35                       ;READ TTY NON-TERMINAL STATUS

;PTY MTOPR NUMBERS

.MOAPI==:24                       ;ASSIGN PTY INTERRUPT CHANNELS
   MO%WFI==:1B0                   ;ENABLE WAITING FOR INPUT
   MO%OIR==:1B1                   ;ENABLE OUTPUT IS WAITING
   MO%SIC==:77B17                 ;SOFTWARE INTERRUPT CHANNEL
.MOPIH==:25                       ;TEST PTY INPUT HUNGRY
   .MONWI==:0                     ;NOT WAITING FOR INPUT
   .MOWFI==:-1                    ;WAITING FOR INPUT
.MOBAT==:26                       ;SET BATCH BIT
  .MOJCB==:1                      ;JOB CONTROLLED BY BATCH
  .MONCB==:0                      ;JOB NOT CONTROLLED BY BATCH
```

```
;TTY MODE DEFINITIONS

.MORLW==:30                     ;READ WIDTH
.MOSLW==:31                     ;SET WIDTH
.MORLL==:32                     ;READ LENGTH
.MOSLL==:33                     ;SET LENGTH
.MOSIG==:36                     ;SET "IGNORE INPUT WHEN INACTIVE" BIT
.MORBM==:37                     ;READ 128 CHARACTER BREAK MASK

        MO%WN1==:776117,,777740 ;BIT DEFINITIONS FOR NON-FORMATTING CONTROL
        MO%WN2==:0              ;FOR ASCII CODES 40-777
        MO%WN3==:0              ;FOR ASCII CODES 100-137
        MO%WN4==:20             ;FOR ASCII CODES 137-177

        MO%WF1==:001260,,000420 ;FORMATTING CONTROL BITS
        MO%WF2==:0              ;FOR ASCII CODES 40-77
        MO%WF3==:0              ;FOR ASCII CODES 100-137
        MO%WF4==:20             ;FOR ASCII CODES 140-177

        MO%WP1==:000400,,400    ;PUNCTUATION BIT DEFINITIONS
        MO%WP2==:777774,,001760 ; FOR ASCII CODES 40-77
        MO%WP3==:400000,,000760 ; FOR ASCII CODES 100-137
        MO%WP4==:400000,,000760 ; FOR ASCII CODES 140-177

        MO%WA1==:400            ;ALPHANUMERICS DEFINITIONS
        MO%WA2==:000003,,776000 ; FOR ASCII CODES 40-77
        MO%WA3==:377777,,777000 ; FOR ASCII CODES 100-137
        MO%WA4==:377777,,777020 ; FOR ASCII CODES 140-177
.MOSBM==:40                     ;SET 128 CHARACTER BREAK MASK
.MORFW==:41                     ;READ FIELD WIDTH
.MOSFW==:42                     ;SET FIELD WIDTH
.MOXOF==:43                     ;SET/CLEAR XOFF/XON HANDLING
        .MOOFF==:0              ;TURN OFF XON/XOFF PROCESSING
        .MOONX==:1              ;TURN ON XON/XOFF PROCESSING
.MORXO==:44                     ;READ VALUE OF XOFF BIT
.MOSLC==:45                     ;SET LINE COUNTER
.MORLC==:46                     ;READ LINE COUNTER
.MOSLM==:47                     ;SET LINE COUNTER MAXIMUM
.MORLM==:50                     ;READ LINE COUNTER MAXIMUM
.MOTPS==:51                     ;PSI FOR NON-CONTROLLING TERMINAL
.MOPCS==:52                     ;SET PAGE PAUSE CHARACTER
.MOPCR==:53                     ;READ PAGE PAUSE CHARACTER

;NET MTOPR NUMBERS

.MOACP==:20           ;TOPS20AN ;ACCEPT CONNECTION ON SOCKET
.MOSND==:21           ;TOPS20AN ;SEND ALL CURENTLY BUFFERED BYTES
.MOSIN==:22           ;TOPS20AN ;SEND INS/INR COMMAND
.MOAIN==:24           ;TOPS20AN ;ASSIGN INS/INR AND FSM PSI CHANNELS
   MO%NIN==:77B5      ;TOPS20AN ;INS/INR SOFTWARE INTERRUPT CHANNEL
   MO%FSM==:77B17     ;TOPS20AN ;FSM CHANGE OF STATE INTERRUPT CHANNEL


;DEFINITIONS FOR DECNET

.MOACN==:24                     ;ASSIGN CONNECT INTERRUPT CHANNEL
   MO%CDN==:777B8               ;CONNECT INTERRUPT CHANNEL
   MO%INA==:777B17              ;INTERRUPT MESSAGE CHANNEL
   MO%DAV==:777B26              ;DATA AVAILABLE CHANNEL
     .MONCI==:777               ;NO CHANGE
     .MOCIA==:776               ;CLEAR INTERRUPT ASSIGNMENT

.MORLS==:25                     ;READ LINK STATUS
```

```
   MO%CON==:1B0                  ;LINK IS CONNECTED
   MO%SRV==:1B1                  ;LINK IS A SERVER
   MO%WFC==:1B2                  ;WAITING FOR A CONNECT
   MO%WCC==:1B3                  ;WAITING FOR THIS LINK TO CONFIRM
   MO%EOM==:1B4                  ;EOM PRESENT IN INPUT BUFFER
   MO%ABT==:1B5                  ;CONNECTION ABORTED
   MO%SYN==:1B6                  ;SYNCH DI RECIEVED
   MO%INT==:1B7                  ;INT MESSAGE AVAILABLE
   MO%LWC==:1B8                  ;LINK WAS CONNECTED
.MORHN==:26                      ;READ HOST NAME
.MORTN==:27                      ;READ TASK NAME
.MORUS==:30                      ;READ USER DATA
.MORPW==:31                      ;READ PASSWORD
.MORAC==:32                      ;READ ACCOUNT
.MORDA==:33                      ;READ OPTIONAL DATA
.MORCN==:34                      ;READ CONNECT OBJECT NUMBER
.MORIM==:35                      ;READ INTERRUPT MESSAGE
.MOSIM==:36                      ;SEND INTERRUPT MESSAGE
.MOROD==:37                      ;READ OBJ-DESC OF CONNECTION
.MOCLZ==:40                      ;CLOSE/REJECT A CONNECTION
.MOCC==:41                       ;ACCEPT A CONNECTION
.MORSS==:42                      ;READ SEGMENT SIZE
.MOANT==:43                      ;ATTACH NETWORK TERMINAL
.MOSNH==:44                      ;SET NETWORK HOST
 .SHTTY==:1                      ;ARG BLOCK - TTY IDENT
 .SHESC==:2                      ; - FLAGS,,ESC CHAR
  SH%LPM==:1B0                   ; FLAG - LOCAL PAGE MODE

;DEFINITIONS FOR ATS

;FUNCTION CODES FOR MTOPR ARE IN COLUMN 1

.MOAMO==:1                       ;SET MODE WORD
        .MOAMM==:1               ;MESSAGE MODE
        .MOADM==:2               ;DATA MODE
.MOAAT==:2                       ;ACQUIRE TERMINAL
        MO%AER==:1B0             ;HTN FIELD CONTAINS AN ERROR CODE
.MOASI==:3                       ;ENABLE INTERRUPTS
        MO%IFL==:777B8           ;FUNCTION TO BE PERFORMED
          .MOAAI==:0             ;ASSIGN INTERRUPT CHANNEL
          .MOADI==:1             ;DEASSIGN INTERRUPT CHANNEL
        MO%IEV==:777B17          ;EVENT BEING ASSIGNED OR DEASSIGNED
          .MOADT==:0             ;DATA ARRIVAL
          .MOAST==:1             ;STATUS ARRIVAL
        MO%ACH==:777777B35       ;CHANNEL NUMBER
.MORCD==:4                       ;GET STATUS
        MO%WDV==:777B35          ;WHICH DEVICES TO REPORT ON
          .MOALD==:0             ;ALL TERMINALS
          .MOCHG==:1             ;TERMINALS WHOSE STATUS HAS CHANGED
          .MOLST==:2             ;TERMINALS SPECIFIED IN LIST
        MO%ARM==:1B0             ;ASK THE RESOURCE MANAGER
        MO%MDA==:1B1             ;MORE DATA AVAILABLE FOR THIS JFN
                AT%OPN==:1B0     ;HTN IS OPEN AND USABLE
                AT%TCL==:1B1     ;NRM CLOSED TERMINAL VIA STATUS-REPORT
                AT%DHT==:1B2     ;DEASSIGNING HTN
                AT%TXF==:1B3     ;TERMINAL IS XOFF'D
                AT%UND==:1B4     ;DEVICE REQUESTED IS UNDEFINED
                AT%NAV==:1B5     ;DEVICE REQUESTED IS NOT AVAILABLE
                AT%OFL==:1B6     ;DEVICE REQUESTED IS OFFLINE
                AT%FUL==:1B7     ;SERVER IS FULL
                AT%UNS==:1B8     ;DEVICE TYPE IS UNSUPPORTED
                AT%REJ==:1B9     ;NODE NRM REJECTED THE REQUEST
                AT%MIE==:1B10    ;MONITOR INTERNAL ERROR (NODE OR HOST)
```

```
                AT%STF==:1B11    ;VT62 START-UP FAILED
                AT%CRJ==:1B12    ;CONNECTION WAS REJECTED
                AT%NDP==:1B13    ;DATA PIPE IS NOT OPEN
                AT%SER==:777777B35 ;STATUS REPORT ERROR CODE (18 BITS)
.MOADE==:5                       ;DEASSIGN TERMINAL
        MO%AAB==:1B0             ;DON'T SEND REMAINING DATA

;FUNCTION CODES FOR AYDIN DISPLAY MTOPR

.MOFLE==:0                       ;FLUSH ERRORS
.MORER==:1                       ;RETURN AYDIN ERROR CODE
.MOWAT==:2                       ;WAIT FOR ACTIVITY TO STOP
        MO%RWC==:777777B17       ;REMAINING WORD COUNT
        MO%LER==:777777B35       ;LAST AYDIN ERROR CODE
```

```
;DEFS FOR MTU JSYS

;FUNCTIONS:

.MTNVV==:1                              ;SET NO VOLUME VALID
        .MTCNT==:0                      ;COUNT WORD
        .MTCOD==:1                      ;ERROR CODE
        .MTPTR==:2                      ;SP TO OPERATOR RESPONSE

.MTRAL==:2                              ;READ ALL LABELS
        .MTVL1==:1                      ;SP TO VOL1 AREA
        .MTVL2==:2                      ;SP TO VOL2 AREA
        .MTHD1==:3                      ;SP TO HDR1 AREA
        .MTHD2==:4                      ;SP TO HDR2 AREA
.MTASI==:3                              ;RETURN MT TO MTA ASSOCIATION
        .MTPHU==:1                      ;RETURN MTA UNIT NUMBER HERE
                .MTNUL==:-1             ;NO ASSIGNMENT CODE
.MTCVV==:4                              ;CLEAR VV
```

```
;MUTIL JSYS FUNCTION CODES

.MUENB==:1                      ;ENABLE PID FOR RECEIVING
.MUDIS==:2                      ;DISABLE PID FROM RECEIVING
.MUGTI==:3                      ;GET PID OF [SYSTEM]INFO
.MUCPI==:4                      ;CREATE A PRIVATE INFO FOR A JOB
.MUDES==:5                      ;DESTROY A PID
.MUCRE==:6                      ;CREATE A PID
.MUSSQ==:7                      ;SET SEND AND RECEIVE QUOTAS
.MUCHO==:10                     ;CHANGE OWNER OF A PID
.MUFOJ==:11                     ;FIND OWNER'S JOB NUMBER
.MUFJP==:12                     ;FIND JOB'S PIDS
.MUFSQ==:13                     ;FIND SEND AND RECEIVE QUOTAS
.MUFFP==:15                     ;FIND FORK'S PIDS
.MUSPQ==:16                     ;SET PID QUOTA
.MUFPQ==:17                     ;FIND PID QUOTA
.MUQRY==:20                     ;QUERY
.MUAPF==:21                     ;ASSOCIATE A PID WITH A FORK
.MUPIC==:22                     ;PUT PID ON AN INTERRUPT CHANNEL
.MUDFI==:23                     ;DEFINE PID OF [SYSTEM]INFO
.MUSSP==:24                     ;SET SYSTEM PID TABLE
.MURSP==:25                     ;READ SYSTEM PID TABLE
.MUMPS==:26                     ;GET MAXIMUM PACKET SIZE
.MUSKP==:27                     ;SET PID TO RECEIVE KILLED PID MESSAGE
.MURKP==:30                     ;READ PID THAT RECEIVES KILLED PID MESSAGES
.MUSPS==:31                     ;Get system maximum packet size


;SYSTEM PID TABLE INDEX VALUES

.SPIPC==:0                      ;PID OF IPCC
.SPINF==:1                      ;PID OF INFO
.SPQSR==:2                      ;PID OF QUASAR
.SPMDA==:3                      ;PID OF QSRMDA
.SPOPR==:4                      ;PID OF OPERATOR JOB (ORION)
.SPNSR==:5                      ;PID OF NETSER


;NODE

.NDSLN==:0                      ;SET LOCAL NODE NAME
.NDGLN==:1                      ;GET LOCAL NODE NAME
   .NDNOD==:0                   ;POINTER TO NODE NAME
.NDSNM==:2                      ;SET LOCAL NODE NUMBER
        .NDMAX==:377            ;MAXIMUM NODE NUMBER
.NDGNM==:3                      ;GET LOCAL NODE NUMBER
.NDSLP==:4                      ;SET LOOPBACK ON PORT
        .NDPRT==:0              ;PORT TO SET IN LOOPBACK
.NDCLP==:5                      ;CLEAR LOOPBACK ON PORT
.NDFLP==:6                      ;FIND LOOPBACK PORT
        ND%LPR==1B0             ;LOOPBACK RUNNING
        ND%LPA==1B1             ;LOOPBACK ASSIGNED TO PORT
.NDSNT==:7                      ;SET NETWORK TOPOLOGY INFORMATION
        .NDNNO==:0              ;NUMBER OF NODES REPRESENTED IN BIT MASK
        .NDMSK==:1              ;FIRST WORD OF REACHABLE NODES BIT MASK
.NDGNT==:10                     ;GET NETWORK TOPOLOGY INFORMATION
        .NDNND==:0              ;NUMBER OF NODE BLOCK POINTERS FOLLOWING
        .NDCNT==:1              ;NUMBER OF WORDS IN A NODE BLOCK
        .NDBK1==2               ;FIRST ADDRESS OF A NODE BLOCK

        ;NODE BLOCK DEFINITIONS
        .NDNAM==:0              ;POINTER TO ASCIZ NODE NAME
        .NDSTA==:1              ;NODE STATE
```

```
                .NDSON==:0          ;ON
              , .NDSOF==:1          ;OFF
        .NDNXT==:2                  ;POINTER TO ASCIZ NEARER NEIGHBOR STRING
        .NDNBS==:3                  ;NODE BLOCK SIZE

.NDSIC==:11                         ;SET TOPOLOGY CHANGE INTERRUPT CHANNEL
        .NDCHN==:0                  ;CHANNEL NUMBER
.NDCIC==:12                         ;CLEAR NETWORK TOPOLOGY INTERRUPT
.NDGVR==:13                         ;GET NSP VERSION INFORMATION
        .NDNVR==:0                  ;NUMBER OF VERSIONS RETURNED
        .NDCVR==:1                  ;POINTER TO COMMUNICATONS VERSION BLOCK
        .NDRVR==:2                  ;POINTER TO ROUTING VERSION BLOCK

        .NDVER==:0                  ;VERSION NUMBER
        .NDECO==:1                  ;ECO NUMBER
        .NDCST==:2                  ;CUSTOMER LEVEL
.NDGLI==:14                         ;GET LINE INFORMATION
        .NDNLN==:0                  ;<# OF ENTRIES FOLLOWING>,,<# LINE RETURNED>
        .NDCNT==:1                  ;NUMBER OF WORDS IN A LINE BLOCK

        ; LINE BLOCK DEFINITION
        .NDLNM==:0                  ;NSP PORT (LINE) NUMBER
        .NDLST==:1                  ;STATE OF LINE
                .NDLON==:1          ;ON
                .NDLOF==:2          ;OFF
                .NDLCN==:3          ;CONTROLLER LOOPBACK
                .NDLCB==:4          ;CABLE LOOPBACK
        .NDLND==:2                  ;BYTE POINTER NODE AT END OF LINE
        .NDLSZ==:3                  ;SIZE OF BLOCK
.NDVFY==:15                         ;VERIFY NODE NAME
        .NDFLG==:1                  ;FLAGS RETURNED BY MONITOR
                ND%EXM==:1B0        ;NODE SPECFIED EXACTLY MATCHES A KNOWN NODE
.NDRNM==:16                         ;GIVEN A NODE NUMBER, RETURN THE NODE NAME

;NOUT

NO%MAG==:1B0                        ;OUTPUT MAGNITUDE
NO%SGN==:1B1                        ;OUTPUT SIGN
NO%LFL==:1B2                        ;LEADING FILLER
NO%ZRO==:1B3                        ;FILL WITH ZERO'S
NO%OOV==:1B4                        ;OUTPUT ON COLUMN OVERFLOW
NO%AST==:1B5                        ;OUTPUT ASTERISKS ON OVERFLOW
NO%COL==:177B17                     ;NUMBER OF COLUMNS TO USE
NO%RDX==:777777                     ;RADIX


;NTMAN% ARGUMENT BLOCK

.NTCNT==:0                          ;NUMBER OF WORDS IN ARGUMENT BLOCK
.NTENT==:1                          ;ENTITY
        .NTNOD==:0                          ;NODE
        .NTLIN==:1                          ;LINE
        .NTLOG==:2                          ;LOGGING
        .NTCKT==:3                          ;CIRCUIT
        .NTMOD==:4                          ;MODULE
.NTEID==:2                          ;BYTE POINTER TO ENTITY ID
.NTFNC==:3                          ;FUNCTION
        LOWFNC==:-2                         ;VALUE OF FIRST FUNCTION VALUE
        .NTMAP==:-2                         ;MAP NODE NUMBER/NODE NAME
        .NTREX==:-1                         ;RETURN EXECUTOR NODE ID
        .NTSET==:0                          ;SET PARAMETER
        .NTCLR==:1                          ;CLEAR PARAMETER
        .NTZRO==:2                          ;ZERO ALL COUNTERS
```

```
        .NTSHO==:3                      ;SHOW SELECTED ITEMS
        .NTSZC==:4                      ;SHOW AND ZERO ALL COUNTERS
        .NTRET==:5                      ;RETURN LIST OF ITEMS
.NTSEL==:4                      ;SELECTION CRITERION

        ;SELECTORS FOR .NTSHO FUNCTION

        .NTSUM==:0                      ;SUMMARY
        .NTSTA==:1                      ;STATUS
        .NTCHA==:2                      ;CHARACTERISTICS
        .NTCOU==:3                      ;COUNTERS
        .NTEVT==:4                      ;EVENT

        ;SELECTORS FOR .NTRET FUNCTION

        .NTKNO==:-1                     ;KNOWN ITEMS
        .NTACT==:-2                     ;ACTIVE ITEMS
        .NTLOP==:-3                     ;LOOP

.NTQUA==:5                      ;BYTE POINTER TO FUNCTION QUALIFIER
.NTBPT==:6                      ;BYTE POINTER TO PARAMETER OR LIST DATA
.NTBYT==:7                      ;NUMBER OF BYTES IN RETURNED DATA
.NTERR==:10                     ;ERROR RETURN STATUS

        ;MISCELLANEOUS NTMAN% SYMBOLS

        .NTARG==:11             ;LENGTH OF NTMAN% ARGUMENT BLOCK
        .NDALN==:2              ;NUMBER OF BYTES IN A NODE ADDRESS
        .NDPLN==:2              ;NUMBER OF BYTES IN A PARAMETER NUMBER
        .NDAMX==:^D255          ;MAXIMUM NODE ADDRESS
        .NDNMX==:7              ;MAXIMUM NUMBER OF BYTES IN A NODE NAME
```

```
OF%FDT==:1B33                 ;FORCE DATE UPDATE

;ODCNV -- SEE IDCNV FOR BITS


;ODTIM

OT%NDA==:1B0                  ;DO NOT OUTPUT DATE
OT%DAY==:1B1                  ;OUTPUT DAY OF WEEK
OT%FDY==:1B2                  ;OUTPUT NUMERIC MONTH
OT%NMN==:1B3                  ;OUTPUT NUMERIC MONTH
OT%FMN==:1B4                  ;OUTPUT MONTH IN FULL
OT%4YR==:1B5                  ;OUTPUT 4-DIGIT YEAR
OT%DAM==:1B6                  ;OUTPUT DAY AFTER MONTH
OT%SPA==:1B7                  ;OUTPUT SPACES IN DATE
OT%SLA==:1B8                  ;OUTPUT SLASHES IN DATE
OT%NTM==:1B9                  ;DO NOT OUTPUT TIME
OT%NSC==:1B10                 ;DO NOT OUTPUT SECONDS
OT%12H==:1B11                 ;OUTPUT 12-HOUR FORMAT
OT%NCO==:1B12                 ;DO NOT OUTPUT COLON
OT%TMZ==:1B13                 ;OUTPUT TIME ZONE
OT%SCL==:1B17                 ;SUPPRESS COLUMNIZATION


;ODTNC -- SEE IDCNV FOR BITS


;OPENF

OF%BSZ==:77B5                 ;BYTE SIZE
OF%MOD==:17B9                 ;MODE
OF%HER==:1B18                 ;HALT ON IO ERROR
OF%RD==:1B19                  ;READ
OF%WR==:1B20                  ;WRITE
OF%EX==:1B21                  ;EXECUTE (RESERVED FOR THE FUTURE)
OF%APP==:1B22                 ;APPEND
OF%RDU==:1B23                 ;READ UNRESTRICTED
OF%THW==:1B25                 ;THAWED
OF%AWT==:1B26                 ;ALWAYS WAIT
OF%PDT==:1B27                 ;PRESERVE DATES
OF%NWT==:1B28                 ;NEVER WAIT
OF%RTD==:1B29                 ;RESTRICTED
OF%PLN==:1B30                 ;SET TO DISABLE LINE NUMBER CHECKING FOR
                              ; NON-LINE NUMBER FILES
OF%DUD==:1B31                 ;DON'T UPDATE TO DISK BY DDMP
OF%OFL==:1B32                 ;ALLOW OPENING THE DEVICE EVEN IF OFFLINE
OF%FDT==:1B33                 ;FORCE DATE UPDATE
OF%RAR==:1B34                 ; Wait if file is off-line
```

```
;PDVOP MANIPULATES PROGRAM DATA VECTORS

;FUNCTION CODES ACCEPTED IN AC1:

.POGET==:0                      ;GET A SET OF PDVAS (PROGRAM DATA VECTOR ADDRESS
.POADD==:1                      ;ADD A SET OF PDVAS
.POREM==:2                      ;REMOVE A SET
.PONAM==:3                      ;GET NAME OF A PROGRAM
.POVER==:4                      ;GET VERSION NUMBER OF A PROGRAM
.POLOC==:5                      ;LOCATE PDVS HAVING SPECIFIED NAME

;ARG BLOCK OFFSETS FOR BLOCK ADDRESSED BY AC2

.POCT1==:0                      ;SIZE OF ARG BLOCK INCLUDING THIS WORD
.POPHD==:1                      ;PROCESS HANDLE
.POCT2==:2                      ;SIZE OF DATA BLOCK (AND SIZE OF RETURNED DATA)
.PODAT==:3                      ;ADDRESS OF DATA BLOCK
.POADR==:4                      ;SMALL ADDRESS OF DATA VECTOR
.POADE==:5                      ;LARGE ADDRESS OF DATA VECTOR ADDRESS RANGE

;OFFSETS DEFINED WITHIN PROGRAM DATA VECTORS

.PVCNT==:0                      ;Length of vector
.PVNAM==:1                      ;Address of a word-aligned ASCIZ program name
.PVSTR==:2                      ;Program starting address
.PVREE==:3                      ;Program reenter address
.PVVER==:4                      ;Program version number
.PVMEM==:5                      ;Address of a block describing program memory
.PVSYM==:6                      ;Address of the program symbol table
.PVCTM==:7                      ;Time of program compilation
.PVCVR==:10                     ;Version number of compiler
.PVLTM==:11                     ;Time of program loading
.PVLVR==:12                     ;Version number of LINK
.PVMON==:13                     ;Address of a monitor data block
.PVPRG==:14                     ;Address of a program data block
.PVCST==:15                     ;Address of a customer-defined block

;PMAP BIT DEFINITIONS

PM%CNT==:1B0                    ;RH WORD CONTAINS A COUNT
PM%MVP==:1B1                    ;MOVE PAGE INSTEAD OF INDIRECT POINTER
                                ; (NOT IMPLEMENTED
PM%RD==:1B2                     ;READ
PM%WT==:1B3                     ;WRITE
PM%WR==:1B3                     ; (ANOTHER NAME FOR ABOVE)
PM%EX==:1B4                     ;EXECUTE (RESERVED FOR THE FUTURE)
PM%RWX==:7B4                    ;CONVENIENT ABBREV FOR RD+WT+EX
PM%PLD==:1B5                    ;PRELOAD PAGES BEING MAPPED
PM%IND==:1B6                    ;USE INDIRECT PTRS (RESERVED FOR THE FUTURE)
PM%TPU==:1B8                    ;TRAP TO USER
                                ; (NOT IMPLEMENTED -- OBSOLETE)
PM%CPY==:1B9                    ;COPY ON WRITE
PM%EPN==:1B10                   ;EXTENDED PAGE NUMBER (18 BITS)
PM%ABT==:1B11                   ;ABORT UNMAP.
PM%RPT==:777777B35              ;REPEAT COUNT


;PMCTL - PHYSICAL MEMORY CONTROL

.MCRCE==:0                      ;READ CACHE ENABLE
.MCSCE==:1                      ;SET CACHE ENABLE
 .MCCST==:0                     ;ARGLIST OFFSET FOR CACHE STATE
 MC%CEN==:1                     ;CACHE ENABLED
```

```
.MCRPS==:2                      ;READ PAGE STATUS
.MCSPS==:3                      ;SET PAGE STATUS
 .MCPPN==:0                     ;ARGLIST OFFSET FOR PHYSICAL PAGE NUMBER
 .MCPST==:1                     ;ARGLIST OFFSET FOR PAGE STATE
 .MCPSA==:0                     ;PAGE AVAILABLE
 .MCPSS==:1                     ;PAGE IN TRANSITION STATE
 .MCPSO==:2                     ;PAGE OFFLINE
 .MCPSE==:3                     ;PAGE OFFLINE DUE TO ERROR
.MCRME==:4                      ;READ MEMORY ERROR INFORMATION
 .PMMER==:1                     ;MOS MEMORY ERROR
 .PMMTP==:0                     ;ENTRY HEADER AND TYPE
 .PMMRG==:1                     ;ERROR REGISTER
 .PMMSY==:2                     ;SYNDROME
 .PMMBN==:3                     ;BLOCK NUMBER
 .PMMSB==:4                     ;SPARE BIT NUMBER
 .PMMEA==:5                     ;ERROR ADDRESS
 .PMMSN==:6                     ;START OF SERIAL NUMBERS
  .PMMNS==:4                    ;# OF SERIAL NUMBERS TO STORE


;PRARG - PROCESS ARGUMENTS

;FUNCTION CODE DEFINITIONS

.PRARD==:1                      ;READ ARGUMENT BLOCK
.PRAST==:2                      ;SET ARGUMENT BLOCK
```

```
;RCUSR AND RCDIR

; FLAGS SUPPLIED ON CALL

RC%PAR==:1B14                   ;PARTIAL RECOGNITION IS ALLOWED
RC%STP==:1B15                   ;STEP WILDCARD (RCDIR ONLY)
RC%AWL==:1B16                   ;ALLOW WILDCARDS (RCDIR ONLY)
RC%EMO==:1B17                   ;EXACT MATCH ONLY

; FLAGS RETURNED

RC%DIR==1B0                     ;FILES-ONLY DIRECTORY
RC%ANA==1B1                     ;ALPHANUMERIC ACCOUNTS ALLOWED
RC%RLM==1B2                     ;REPEAT LOGIN MESSAGE
RC%NOM==:1B3                    ;NO MATCH FOUND
RC%AMB==:1B4                    ;AMBIGUOUS
RC%NMD==:1B5                    ;NO MORE DIRS - RETURNED IF STP IS REQUESTED
RC%WLD==:1B6                    ;WILDCARD DIR WAS INPUT


;RCVOK

.RCFCJ==:0                      ;FUNCTION CODE,, JOB NUMBER
.RCUNO==:1                      ;USER NUMBER
.RCCDR==:2                      ;CONNECTED DIRECTORY
.RCRQN==:3                      ;REQUEST NUMBER
.RCNUA==:4                      ;NUMBER OF USER ARGS
.RCARA==:5                      ;POINTER TO USER ARGS
.RCCAP==:6                      ;CURRENT CAPABILITIES
.RCTER==:7                      ;TERMINAL NUMBER
.RCRJB==:10                     ;REQUESTED JOB

;RDTTY AND TEXTI

RD%BRK==:1B0                    ;BREAK ON REGULAR BREAK SET
RD%TOP==:1B1                    ;BREAK ON TOPS10 BREAK SET
RD%PUN==:1B2                    ;BREAK ON PUNCTUATION
RD%BEL==:1B3                    ;BREAK ON END OF LINE
RD%CRF==:1B4                    ;SUPPRESS CR (RETURNS LF ONLY)
RD%RND==:1B5                    ;RETURN IF NOTHING TO DELETE
RD%JFN==:1B6                    ;JFNS GIVEN FOR SOURCE
RD%RIE==:1B7                    ;RETURN ON INPUT (BUFFER) EMPTY
RD%BBG==:1B8                    ;BEGINNING OF (DEST) BUFFER GIVEN
RD%BEG==:1B9                    ;RETURN IMMEDIATELY WHEN TYPIST EDITS TO .RDBKL
RD%RAI==:1B10                   ;RAISE LOWERCASE INPUT
RD%SUI==:1B11                   ;SUPPRESS ^U INDICATION
RD%BTM==:1B12                   ;BREAK CHARACTER TERMINATED INPUT
RD%BFE==:1B13                   ;RETURNED BECAUSE BUFFER EMPTY
RD%BLR==:1B14                   ;BACKUP LIMIT REACHED

;TEXTI ARG BLOCK

.RDCWB==:0                      ;COUNT OF WORDS IN BLOCK
.RDFLG==:1                      ;FLAGS
.RDIOJ==:2                      ;IO JFNS
.RDDBP==:3                      ;DEST BYTE POINTER
.RDDBC==:4                      ;DEST BYTE COUNT
.RDBFP==:5                      ;TOP OF BUFFER POINTER
.RDRTY==:6                      ;RETYPE (^R) POINTER
.RDBRK==:7                      ;BREAK SET MASK POINTER
.RDBKL==:10                     ;BACKUP LIMIT POINTER
```

```
;RFSTS

RF%LNG==:1B0                     ;LONG FORM OF RFSTS CALL, ARG BLOCK IN 2
RF%PRH==:777777B35               ;PROCESS HANDLE

;RFSTS ARG BLOCK

.RFCNT==:0                       ;XWD COUNT OF WORDS RETURNED,
                                 ; MAXIMUM WORDS TO RETURN
.RFPSW==:1                       ;PROCESS STATUS WORD
.RFPFL==:2                       ;PROCESS' PC FLAGS
.RFPPC==:3                       ;PROCESS' PC
.RFSFL==:4                       ;STATUS FLAGS FOR PROCESS:
  RF%EXO==1B0                    ;PROCESS IS EXECUTE-ONLY

;PROCESS STATUS WORD

RF%FRZ==:1B0                     ;PROCESS IS FROZEN
RF%STS==:377777B17               ;PROCESS STATUS CODE
   .RFRUN==:0                    ;RUNNABLE
   .RFIO==:1                     ;DISMISSED FOR I/O
   .RFHLT==:2                    ;HALTED
   .RFFPT==:3                    ;FORCED PROCESS TERMINATION
   .RFWAT==:4                    ;WAITING FOR INFERIOR PROCESS
   .RFSLP==:5                    ;SLEEP
   .RFTRP==:6                    ;JSYS TRAPPED
   .RFABK==:7                    ;ADDRESS BREAK FREEZE
RF%SIC==:777777B35               ;SOFTWARE INTERRUPT CHANNNEL


;RFTAD/SFTAD

.RSWRT==:0                       ;WRITE DATE WORD
.RSCRV==:1                       ;CREATION DATE WORD
.RSREF==:2                       ;REFERENCE DATE WORD
.RSCRE==:3                       ;INTERNAL SYSTEM WRITE DATE WORD
.RSTDT==:4                       ; Tape write date word
.RSNET==:5                       ; Online expiration date/interval word
.RSFET==:6                       ; Offline expiration date/interval word


;RMAP

RM%RD==1B2                       ;READ ACCESS ALLOWED
RM%WR==:1B3                      ;WRITE ACCESS ALLOWED
RM%EX==:1B4                      ;EXECUTE ACCESS ALLOWED
RM%PEX==:1B5                     ;PAGE EXISTS
RM%CPY==:1B9                     ;COPY ON WRITE

;RSMAP/SMAP

SM%RD==:1B2                      ;READ ACCESS ALLOWED
SM%WR==:1B3                      ;WRITE ACCESS ALLOWED
SM%EX==:1B4                      ;EXECUTE ACCESS ALLOWED
SM%IND==:1B6                     ;INDIRECT POINTER


;RPACS/SPACS BIT DEFINITIONS

PA%RD==:1B2                      ;READ ACCESS ALLOWED
PA%WT==:1B3                      ;WRITE ACCESS ALLOWED
PA%WR==:1B3                      ; (ANOTHER NAME FOR ABOVE)
PA%EX==:1B4                      ;EXECUTE ACCESS ALLOWED
```

```
                                ; (RESERVED FOR THE FUTURE)
PA%PEX==:1B5                    ;PAGE EXISTS
PA%IND==:1B6                    ;INDIRECT POINTER
PA%TPU==:1B8                    ;TRAP TO USER
                                ; (NOT IMPLEMENTED -- OBSOLETE)
PA%CPY==:1B9                    ;COPY ON WRITE
PA%PRV==:1B10                   ;PRIVATE
P1%RD==:1B20                    ;READ ACCESS ALLOWED IN 1ST POINTER
P1%WR==:1B21                    ;WRITE ACCESS ALLOWED IN 1ST POINTER
P1%WT==:1B21                    ; (ANOTHER NAME FOR ABOVE)
P1%EX==:1B22                    ;EXECUTE ACCESS ALLOWED IN 1ST POINTER
                                ; (RESERVED FOR THE FUTURE)
P1%PEX==:1B23                   ;PAGE EXISTS IN 1ST POINTER
P1%CPY==:1B27                   ;COPY-ON-WRITE IN 1ST POINTER
```

```
;RSCAN

.RSINI==:0                       ;MAKE RESCAN BUFFER AVAILABLE FOR INPUT
.RSCNT==:1                       ;COUNT CHARACTERS LEFT TO READ FROM
                                 ; RESCAN BUFFER


;RTIW

RT%DIM==:1B0                     ;DEFERRED TERMINAL INTERRUPT MASK GIVEN
RT%PRH==:777777                  ;PROCESS HANDLE


;SCTTY

.SCRET==:0                       ;RETURN DESIGNATOR (CTTY) FOR FORK
.SCSET==:1                       ;SET SCTTY FOR FORK
.SCRST==:2                       ;CLEAR FORK CTTY (RESTORE JOB CTTY)

;SCVEC

.SVEAD==:0                       ;ENTRY ADDRESS
.SVINE==:1                       ;INITIAL ENTRY FOR SETUP
.SVGET==:2                       ;ENTRY ADDRESS FOR GET SHARE FILE ROUTINE
.SV40==:3                        ;ADDRESS TO GET LOCATION 40
.SVRPC==:4                       ;ADDRESS TO GET RETURN PC
.SVMAK==:5                       ;ENTRY FOR MAKE SHARE FILE ROUTINE
.SVCST==:6                       ;2 WORD BLOCK FOR CONTROL-C/START PROCESSING


;SDVEC

.SDEAD==:0                       ;ENTRY ADDRESS
.SDINE==:1                       ;INITIAL ENTRY
.SDVER==:2                       ;DMS VERSION
.SDDMS==:3                       ;ADDRESS TO STORE DMS JSYS
.SDRPC==:4                       ;ADDRESS TO STORE RETURN PC
```

```
;SETJB FUNCTION CODES

.SJDEN==:0                      ;SET DEFAULT MAGTAPE DENSITY
 .SJDDN==:0                     ;SYSTEM DEFAULT DENSITY
 .SJDN2==:1                     ;200 BPI
 .SJDN5==:2                     ;556 BPI
 .SJDN8==:3                     ;800 BPI
 .SJD16==:4                     ;1600 BPI
 .SJD62==:5                     ;6250 BPI
.SJPAR==:1                      ;SET DEFAULT MAGTAPE PARITY
 .SJPRO==:0                     ;ODD PARITY
 .SJPRE==:1                     ;EVEN PARITY
.SJDM==:2                       ;SET DEFAULT MAGTAPE DATA MODE
 .SJDDM==:0                     ;SYSTEM DEFAULT DATA MODE
 .SJDMC==:1                     ;CORE DUMP MODE
 .SJDM6==:2                     ;SIX BIT BYTE MODE (FOR 7-TRACK DRIVES)
 .SJDMA==:3                     ;ANSI ASCII MODE (7 BITS IN 8 BIT BYTE)
 .SJDM8==:4                     ;INDUSTRY COMPATIBLE MODE
 .SJDMH==:5                     ;HI-DENSITY MODE (9 EIGHT BIT
                                ; BYTES IN 2 WORDS)
.SJRS==:3                       ;SET DEFAULT MAGTAPE RECORD SIZE
.SJDFS==:4                      ;SET DEFERRED SPOOLING
 .SJSPI==:0                     ;IMMEDIATE MODE SPOOLING
 .SJSPD==:1                     ;DEFERRED MODE SPOOLING
.SJSRM==:5                      ;SET JOB SESSION REMARK
.SJT20==:6                      ;DECLARE WHETHER TOPS20 COMMAND LEVEL OR NOT
.SJDFR==:7                      ; Set default job retrieval mode
 .SJRFA==:0                     ; OPENF should always fail
 .SJRWA==:1                     ; OPENF should always request & wait
.SJBAT==:10                     ;SET BATCH FLAGS AND STREAM
                                ;SEE .JIBCH FOR FIELD DEFINITIONS
.SJLLO==:11                     ;SET JOB LOCATION


;SFORK

SF%CON==:1B0                    ;CONTINUE PROCESS, IGNORE PC IN AC2
SF%PRH==:777777B35              ;PROCESS HANDLE


;SFUST

.SFAUT==:0                      ;SET AUTHOR STRING
.SFLWR==:1                      ;SET LAST WRITER STRING
```

```
;SMON FUNCTION CODES AND BIT DEFINITIONS (SYSTEM FLAGS)

.SFFAC==:0                      ;ALLOW FACT ENTRIES
.SFCDE==:1                      ;CHECKDISK FOUND ERRORS
.SFCDR==:2                      ;CHECKDISK RUNNING
.SFMST==:3                      ;MANUAL START IN PROGRESS
.SFRMT==:4                      ;REMOTE LOGINS ALLOWED
.SFPTY==:5                      ;PTY LOGINS ALLOWED
.SFCTY==:6                      ;CTY LOGIN ALLOWED
.SFOPR==:7                      ;OPERATOR IN ATTENDANCE
.SFLCL==:10                     ;LOCAL LOGINS ALLOWED
.SFBTE==:11                     ;BIT TABLE ERRORS FOUND ON STARTUP
.SFCRD==:12                     ;USER CAN CHANGE DIRECTORY CHARACTERISTICS
.SFNVT==:13           ;TOPS20AN ;NVT LOGIN ALLOWED
.SFWCT==:14                     ;WHEEL LOGIN ON CTY ALLOWED
.SFWLC==:15                     ;WHEEL LOGIN ON LOCAL TERMINALS ALLOWED
.SFWRM==:16                     ;WHEEL LOGIN ON REMOTE TERMINALS ALLOWED
.SFWPT==:17                     ;WHEEL LOGIN ON PTY'S ALLOWED
.SFWNV==:20           ;TOPS20AN ;WHEEL LOGIN ON NVT'S ALLOWED
.SFUSG==:21                     ;USAGE FILE IN USE
.SFFLO==:22                     ;FULL LATENCY OPTIMIZATION
                                ;CAUTION: SETTING THIS REQUIRES THAT THE
                                ;  SYSTEM BE AT REVISION LEVEL 10, AND
                                ;  THAT RH20 BOARD M8555 BE AT REVISION LEVEL I
                                ;  OTHERWISE, THE FILE-SYSTEM MAY BE DAMAGED.

.SFMTA==:23                     ;MAGTAPE ALLOCATION ENABLED
.SFMS0==:24                     ;SYSTEM MESSAGE LEVEL 0
.SFMS1==:25                     ;SYSTEM MESSAGE LEVEL 1
;BELOW ARE FUNCTION CODES THAT DO NOT MAP DIRECTLY INTO BITS


.SFNTN==:44           ;TOPS20AN ;NETWORK ON/OFF CONTROL
.SFNDU==:45           ;TOPS20AN ;NET DOWN/UP REQUEST
.SFNHI==:46           ;TOPS20AN ;NET HOST TABLE INITIALIZE
.SFTMZ==:47                     ;SET TIME ZONE THIS SYSTEM IS IN
.SFLHN==:50           ;TOPS20AN ;SET LOCAL HOST NUMBER OF THIS NET SITE
.SFAVR==:51                     ;ACCOUNT VALIDATION ON/OFF
.SFSTS==:52                     ;ENABLE/DISABLE STATUS REPORTING
.SFSOK==:53                     ;GETOK/GIVOK DEFAULT SETTING
.SFMCY==:54                     ;SET MAX ORDINARY OFFLINE EXP PERIOD
.SFRDU==:55                     ;READ DATE UPDATE FUNCTION
.SFACY==:56                     ;SET MAX ARCHIVE EXP PERIOD
.SFRTW==:57                     ;SET [NO] RETRIEVAL WAITS NON-0 => NO WAIT
.SFTDF==:60                     ;TAPE MOUNT CONTROLS
  MT%UUT==1B0                   ;UNLOAD UNREADABLE TAPES
.SFWSP==:61                     ;WORKING SET PRELOADING

SF%FAC==:1B<.SFFAC>             ;FACT ENTRIES ALLOWED
SF%CDE==:1B<.SFCDE>             ;CHECKDISK FOUND ERRORS
SF%CDR==:1B<.SFCDR>             ;CHECKDISK RUNNING
SF%MST==:1B<.SFMST>             ;MANUAL START IN PROGRESS
SF%RMT==:1B<.SFRMT>             ;REMOTE LOGINS ALLOWED
SF%PTY==:1B<.SFPTY>             ;PTY LOGINS ALLOWED
SF%CTY==:1B<.SFCTY>             ;CTY LOGIN ALLOWED
SF%OPR==:1B<.SFOPR>             ;OPERATOR IN ATTENDANCE
SF%LCL==:1B<.SFLCL>             ;LOCAL LOGINS ALLOWED
SF%BTE==:1B<.SFBTE>             ;BIT TABLE ERRORS FOUND ON STARTUP
SF%CRD==:1B<.SFCRD>             ;USER CAN CHANGE DIRECTORY CHARACTERISTICS
SF%NVT==:1B<.SFNVT>   ;TOPS20AN ;NVT LOGINS ALLOWED
SF%USG==:1B<.SFUSG>             ;USAGE FILE IN USE
SF%FLO==:1B<.SFFLO>             ;FULL LATENCY OPTIMIZATION IN USE
                                ;CAUTION: SETTING THIS REQUIRES THAT THE
```

```
                                ;  SYSTEM BE AT REVISION LEVEL 10, AND
                                ;  THAT RH20 BOARD M8555 BE AT REVISION LEVEL D.
                                ;  OTHERWISE, THE FILE-SYSTEM MAY BE DAMAGED.
SF%MTA==:1B<.SFMTA>             ;MAGTAPE ALLOCATION ENABLED
SF%MS0==:1B<.SFMS0>             ;SYSTEM MESSAGE LEVEL 0
SF%MS1==:1B<.SFMS1>             ;SYSTEM MESSAGE LEVEL 1


SF%EOK==:1B0                    ;ENABLE ACCESS CHECKING
SF%DOK==:1B1                    ;ALLOW ACCESS IF CHECKING DISABLED


;SINM JSYS DEFINITIONS

SI%TMG==:1B0                    ;TRUNCATE MESSAGE
SI%EOM==:1B1                    ;END-OF-MESSAGE FOUND

;SIR JSYS (NEW FORM)

SI%VER==:7B17                   ;VERSION OF SIR IN T1
SI%LEV==:77B5                   ;LEVEL FIELD IN CHNTAB
SI%ADR==:7777,,-1               ;ADDRESS OF INTERRUPT ROUTINE IN CHNTAB

;SKED JSYS

        .SACNT==:0              ;ARGUMENT BLOCK OFFSET FOR COUNT

        ;FUNCTION CODES

.SKRBC==:1                      ;READ BIAS CONTROL KNOB
        .SAKNB==:1              ;OFFSET FOR KNOB VALUE
.SKSBC==:2                      ;SET BIAS CONTROL KNOB
.SKRCS==:3                      ;READ SHARE OF A CLASS
        .SACLS==:1              ;CLASS
        .SASHR==:2              ;SHARE
        .SAUSE==:3              ;USE
        .SA1ML==:4              ;1 MINUTE LOAD AVERAGE
        .SA5ML==:5              ;5 MINUTE LOAD AVERAGE
        .SA15L==:6              ;15 MINUTE LOAD
.SKSCS==:4                      ;SET SHARE OF A CLASS
.SKICS==:5                      ;START OR STOP CLASS SCHEDULING
        .SACTL==:1              ;WORD FOR CONTROL BITS
.SKSCJ==:6                      ;SET CLASS OF A JOB
        .SAJOB==:1              ;JOB
        .SAJCL==:2              ;CLASS OF JOB
        .SAWA==:3               ;WA ON/OFF SWITCH
.SKRJP==:7                      ;READ CLASS PARAMETERS FOR A JOB
        .SAJSH==:3              ;JOB'S SHARE ALLOTMENT
        .SAJUS==:4              ;JOB'S CURRENT USE
.SKBCR==:10                     ;READ CLASS SETTING FOR BATCH JOBS
        .SABCL==:1              ;BATCH CLASS
.SKBCS==:11                     ;SET CLASS FOR BATCH JOBS
.SKBBG==:12                     ;RUN BATCH JOBS ON DREGS QUEUE
        .SADRG==:1              ;WORD TO SPECIFY DREGS OR NOT
.SKDDC==:13                     ;SET SYSTEM CLASS DEFAULT
        .SADCL==:1              ;DEFAULT CLASS WORD
.SKRCV==:14                     ;READ STATUS
        SK%ACT==:1B0            ;CLASS BY ACCOUNTS
        SK%WDF==:1B1            ;WITHHOLD WINDFALL
        SK%STP==:1B2            ;CLASS SCHEDULER OFF
        SK%DRG==:1B3            ;BATCH JOBS ARE BEING RUN ON DREGS QUEUE

;SJPRI, SPRIW - PRIORITY WORD
```

```
JP%RTG==:177B17                    ;RUN TIME GUARANTEE PERCENTAGE
JP%SYS==:1B18                      ;SYSTEM FORK (PRIORITY ABOVE ALL CLASSES)
JP%MNQ==:77B29                     ;MINIMUM QUEUE
JP%MXQ==:77B35                     ;MAXIMUM QUEUE
```

```
;SNOOP JSYS DEFINITIONS

;SNOOP FUNCTION CODES

.SNPLC==:0                      ;LOCK CODE INTO MONITOR VIRT MEMORY
.SNPLS==:1                      ;LOCK DOWN THE SWAPPABLE MONITOR
.SNPDB==:2                      ;DEFINE A BREAK POINT
.SNPIB==:3                      ;INSERT THE BREAK POINTS
.SNPRB==:4                      ;REMOVE THE BREAK POINTS
.SNPUL==:5                      ;UNLOCK AND RELEASE ALL SNOOP RESOURCES
.SNPSY==:6                      ;LOOK UP A MONITOR SYMBOL
.SNPAD==:7                      ;LOOK UP ADDRESS IN SYMBOL TABLE

;SOUTM JSYS DEFINITIONS

SO%WMG==1B0                     ;WRITE END-OF-MESSAGE

;SPOOL JSYS FUNCTION CODES

.SPLDI==:0                      ;DEFINE AN INPUT SPOOLING DEVICE
.SPLSD==:1                      ;SET DIRECTORY OF SPOOLED DEVICE
.SPLRD==:2                      ;READ DIRECTORY OF SPOOLED DEVICE

;FLAGS IN SPOOL MESSAGE ON LOGOUT AND SPOOLED FILE CLOSE

SP%BAT==:1B0                    ;JOB IS A BATCH JOB
SP%DFS==:1B1                    ;SPOOLING IS DEFERRED
SP%ELO==:1B2                    ;JOB EXECUTED LGOUT JSYS ITSELF
SP%FLO==:1B3                    ;JOB FORCED TO LOG OUT BY TRAP IN TOP FK
SP%OLO==:1B4                    ;OTHER JOB AIMED LGOUT AT THIS ONE

;SPOOL ARGUMENT BLOCK

.SPLDV==:0                      ;DEVICE DESIGNATOR
.SPLNA==:1                      ;NAME STRING
.SPLDR==:1                      ;DIRECTORY NUMBER
.SPLGN==:2                      ;GENERATION NUMBER


;SSAVE

SS%NNP==777777B17               ;NEGATIVE NUMBER OF PAGES
SS%CPY==:1B18                   ;ALLOW COPY-ON-WRITE
SS%UCA==:1B19                   ;USE CURRENT ACCESS
SS%RD==:1B20                    ;ALLOW READ ACCESS
SS%WR==:1B21                    ;ALLOW WRITE ACCESS
SS%EXE==:1B22                   ;ALLOW EXECUTE ACCESS
SS%EPN==:1B23                   ;TABLE ENTRY IS TWO WORDS
                                ; (PAGE NUMBER IN SECOND WORD)
SS%FPN==:1B27+377B35T           ;FIRST PAGE NUMBER


;STCMP

SC%LSS==:1B0                    ;T1 LESS THAN T2
SC%SUB==:1B1                    ;T1 SUBSTRING OF T2
SC%GTR==:1B2                    ;T1 GREATER THAN T2
```

```
;STDIR

ST%DIR==:1B0                    ;FILES ONLY DIRECTORY
ST%ANA==:1B1                    ;ALPHANUMERIC ACCOUNTS
ST%RLM==:1B2                    ;REPEAT LOGIN MESSAGE


;STIW

ST%DIM==:1B0                    ;SET DEFERRED INTERRUPT MASK
ST%PRH==:777777B35              ;PROCESS HANDLE

;SWTRP DEFINITIONS

.SWART==:0                      ;SET ARITHMETIC TRAP
.SWRAT==:1                      ;READ ARITHMETIC TRAP
.SWLUT==:2                      ;SET LUUO ADDRESS
.SWRLT==:3                      ;READ LUUO ADDRESS
        .ARPFL==:0              ;OFFSET IN TRAP BLOCK FOR PC FLAGS
        .AROPC==:1              ;OFFSET FOR OLD PC VALUE
        .AREFA==:2              ;OFFSET FOR E
        .ARNPC==:3              ;OFFSET FOR NEW PC WORD

;TBLUK

TL%NOM==:1B0                    ;NO MATCH
TL%AMB==:1B1                    ;AMBIGUOUS
TL%ABR==:1B2                    ;LEGAL ABBREVIATION
TL%EXM==:1B3                    ;EXACT MATCH


;TFORK

;FUNCTION CODES IN LH AC1

.TFSET==:0                      ;SET TRAPS AS SPEC'D BY BIT TABLE
.TFRAL==:1                      ;REMOVE ALL TRAPS SET BY THIS FORK
.TFRTP==:2                      ;REMOVE TRAPS SET BY THIS FORK
.TFSPS==:3                      ;SET JSYS TRAP PSI CHAN IN LH(2)
.TFRPS==:4                      ;READ JSYS TRAP PSI CHAN INTO LH(2)
.TFTST==:5                      ;TEST IF SELF MONITORED
.TFRES==:6                      ;REMOVE TRAPS FROM ALL INFERIORS, CLR PSI
.TFUUO==:7                      ;SET UUO TRAPS FOR FORK
.TFSJU==:8                      ;SET BOTH UUO AND JSYS TRAPS
.TFRUU==:9                      ;REMOVE UUO TRAPS

;TIMER DEFINITIONS

.TIMRT==:0                      ;SET TIME LIMIT
.TIMEL==:1                      ;SET ELAPSED TIME CLOCK
.TIMDT==:2                      ;SET DATE & TIME CLOCK
.TIMDD==:3                      ;DELETE AN EXPLICT DATE & TIME CLOCK
.TIMBF==:4                      ;DELETE ALL ENTIRES BEFORE D&T
.TIMAL==:5                      ;DELETE ALL (INCLUDES TIME LIMIT)
```

```
;TLINK

TL%CRO==:1B0                    ;CLEAR REMOTE TO OBJECT LINK
TL%COR==:1B1                    ;CLEAR OBJECT TO REMOTE LINK
TL%EOR==:1B2                    ;ESTABLISH OBJECT TO REMOTE LINK
TL%ERO==:1B3                    ;ESTABLISH REMOTE TO OBJECT LINK
TL%SAB==:1B4                    ;SET ACCEPT BIT FOR OBJECT
TL%ABS==:1B5                    ;ACCEPT BIT STATE
TL%STA==:1B6                    ;SET OR CLEAR ADVICE
TL%AAD==:1B7                    ;ACCEPT ADVICE
TL%OBJ==:777777B35              ;OBJECT DESIGNATOR


;UFPGS

UF%NOW==:1B0                    ;NO WAIT ON UPDATE

;UTEST FUNCTION CODES

.UTSET==:0                      ;START TESTING
.UTCLR==:1                      ;STOP TESTING AND RETURN RESULTS

;UTEST ARGUMENT BLOCK

.UTADR==:0                      ;STARTING ADDRESS OF CODE
.UTLEN==:1                      ;LENGTH OF CODE
.UTMAP==:2                      ;START OF BIT MAP

;USAGE

.USENT==:0                      ;WRITE ENTRY
.USCLS==:1                      ;CLOSE OUT CURRENT FILE
.USCKP==:2                      ;PERFORM CHECKPOINT
.USLGI==:3                      ;LOGIN
.USLGO==:4                      ;LOGOUT
.USSEN==:5                      ;SESSION END
.USCKI==:6                      ;SET CHECKPOINT INTERVAL
.USENA==:7                      ;ENABLE ACCOUNT VALIDATION
.USCAS==:10                     ;CHANGE ACCOUNTING SHIFT NOW
.USSAS==:11                     ;SET AUTOMATIC ACCOUNTING SHIFT CHANGE TIMES
.USRAS==:12                     ;READ AUTOMATIC ACCOUNTING SHIFT CHANGE TIMES
                                ;TABLE ENTRY FORMAT FOR .USSAS/.USRAS:
 US%DOW==:177B6                 ;DAY-OF-WEEK BITS
 US%SSM==:777777                ;TIME IN SECONDS SINCE MIDNIGHT

;UTFRK

UT%TRP==:1B0                    ;ITRAP (OR DO ERJMP/ERCAL) TRAPPED JSYS
```

```
;WILD FUNCTIONS

.WLSTR==:0                      ;COMPARE TWO STRINGS
.WLJFN==:1                      ;COMPARE TWO JFNS

;WILD FLAGS AND BITS

WL%LCD==:1B0                    ;DON'T CONVERT LOWER CASE TO UPPER CASE
WL%NOM==:1B0                    ;STRINGS DID NOT MATCH
WL%ABR==:1B1                    ;NON-WILD STRING IS ABBREVIATION OF WILD STRIN
WL%DEV==:1B1                    ;DEVICE FIELD DID NOT MATCH
WL%DIR==:1B2                    ;DIRECTORY FIELD DID NOT MATCH
WL%NAM==:1B3                    ;NAME FIELD DID NOT MATCH
WL%EXT==:1B4                    ;FILE TYPE DID NOT MATCH
WL%GEN==:1B5                    ;GENERATION NUMBER DID NOT MATCH

;ARGUMENT BLOCK OFFSETS FOR XSIR AND XRIR JSYS'S

.SICNT==:0                      ;LENGTH OF BLOCK
.SILVT==:1                      ;ADDRESS OF LEVEL TABLE
.SICHT==:2                      ;ADDRESS OF CHANNEL TABLE


;SCHEDULER CONTROL FLAGS (JSYS NOT YET DEFINED)

SK%CYT==:1B18                   ;CYCLE TIME
SK%IOC==:1B19                   ;IO QUANTUM CHARGE
SK%HT1==:1B20                   ;LIMIT HOLD TIME
SK%HT2==:1B21                   ;NO HOLD TIME AFTER SKIPPED FORK
SK%HQR==:1B22                   ;HIGH QUEUE FORK HAVE PRIORITY UNDER LOAD
SK%CL1==:1B23                   ;CLASS SKED, USE NORMAL QUEUE PRIORITIES IF 1
;SK%BQE==:1B24                  ;BALSET QUEUE ON ENTRY
SK%RSQ==:1B25                   ;QUICK RESCHEDULE ON WAKEUPS
SK%RQ1==:1B26                   ;REQUEUE TO QUEUE 1
SK%TTP==:1B27                   ;TTY PREFERENCE
SK%WCF==:1B28                   ;WAIT CREDIT PROPORTIONAL TO LOAD AV
SK%TOP==:1B29                   ;TTY OUTPUT PREFERENCE
SK%RQM==:1B30                   ;REQUEUE DEPENDS ON MEM DEMAND
```

```
;***********************************************
;GENERAL FIELD AND VALUE DEFINITIONS
;USED BY MANY JSYSES
;***********************************************


;GENERAL FORK HANDLES

.FHSLF==:400000                 ;SELF
 FH%EPN==:1B19                  ;EXTENDED PAGE NUMBER
.FHSUP==:<Z -1>                 ;SUPERIOR
.FHTOP==:<Z -2>                 ;TOP IN JOB
.FHSAI==:<Z -3>                 ;SELF AND INFERIORS
.FHINF==:<Z -4>                 ;INFERIORS
.FHJOB==:<Z -5>                 ;ALL IN JOB


;FIELDS OF JFN MODE WORD

TT%OSP==:1B0                    ;OUTPUT SUPPRESS
TT%MFF==:1B1                    ;MECHANICAL FORMFEED PRESENT
TT%TAB==:1B2                    ;MECHANICAL TAB PRESENT
TT%LCA==:1B3                    ;LOWER CASE CAPABILITIES PRESENT
TT%LEN==:177B10                 ;PAGE LENGTH
TT%WID==:177B17                 ;PAGE WIDTH
TT%WAK==:17B23                  ;WAKEUP FIELD
TT%WK0==:1B18                   ;WAKEUP CLASS 0 (UNUSED)
TT%IGN==:1B19                   ;IGNORE TT%WAK ON SFMOD
TT%WKF==:1B20                   ;WAKEUP ON FORMATING CONTROL CHARS
TT%WKN==:1B21                   ;WAKEUP ON NON-FORMATTING CONTROLS
TT%WKP==:1B22                   ;WAKEUP ON PUNCTUATION
TT%WKA==:1B23                   ;WAKEUP ON ALPHANUMERICS
TT%ECO==:1B24                   ;ECHOS ON
TT%ECM==:1B25                   ;ECHO MODE
TT%ALK==:1B26                   ;ALLOW LINKS
TT%AAD==:1B27                   ;ALLOW ADVICE (NOT IMPLEMENTED)
TT%DAM==:3B29                   ;DATA MODE
.TTBIN==:0                      ;BINARY
.TTASC==:1                      ;ASCII
.TTATO==:2                      ;ASCII AND TRANSLATE OUTPUT ONLY
.TTATE==:3                      ;ASCII AND TRANSLATE ECHOS ONLY
TT%UOC==:1B30                   ;UPPER CASE OUTPUT CONTROL
TT%LIC==:1B31                   ;LOWER CASE INPUT CONTROL
TT%DUM==:3B33                   ;DUPLEX MODE
.TTFDX==:0                      ;FULL DUPLEX
.TT0DX==:1                      ;NOT USED, RESERVED
.TTHDX==:2                      ;HALF DUPLEX (CHARACTER)
.TTLDX==:3                      ;LINE HALF DUPLEX
TT%PGM==:1B34                   ;PAGE MODE
TT%CAR==:1B35                   ;CARRIER STATE
```

```
;DIRECTORY PROTECTION DEFINITIONS (3 6-BIT FIELDS: OWNER, GROUP, WORLD)

DP%RD==:40                    ;READING DIRECTORY IS ALLOWED
DP%CN==:10                    ;CONNECT TO DIR, OR CHANGE PROT/ACCOUNT
DP%CF==:4                     ;CREATING FILES IN DIR IS ALLOWED

;FILE PROTECTION DEFINITIONS (3 6-BIT FIELDS: OWNER, GROUP, WORLD)

FP%DIR==:2                    ;DIRECTORY LISTING
FP%APP==:4                    ;APPEND
FP%EX==:10                    ;EXECUTE
FP%WR==:20                    ;WRITE
FP%RD==:40                    ;READ


;INPUT AND OUTPUT IDENTIFIERS

.PRIIN==:100                  ;PRIMARY INPUT
.PRIOU==:101                  ;PRIMARY OUTPUT
.NULIO==:377777               ;NULL DESIGNATOR
.CTTRM==:777777               ;JOB'S CONTROLLING TERMINAL
.DVDES==:600000               ;UNIVERSAL DEVICE CODE
.TTDES==:400000               ;UNIVERSAL TERMINAL CODE


;MAGTAPE DEVICE STATUS BITS

MT%ILW==:1B18                 ;ILLEGAL WRITE
MT%DVE==:1B19                 ;DEVICE ERROR
MT%DAE==:1B20                 ;DATA ERROR
MT%SER==:1B21                 ;SUPPRESS ERROR RECOVERY PROCEDURES
MT%EOF==:1B22                 ;EOF (FILE MARK)
MT%IRL==:1B23                 ;INCORRECT RECORD LENGTH
MT%BOT==:1B24                 ;BEGINNING OF TAPE
MT%EOT==:1B25                 ;END OF TAPE
MT%EVP==:1B26                 ;EVEN PARITY
MT%DEN==:3B28                 ;DENSITY (0 IS 'NORMAL')
.MTLOD==:1                    ;LOW DENSITY (200 BPI)
.MTMED==:2                    ;MEDIUM DENSITY (556 BPI)
.MTHID==:3                    ;HIGH DENSITY (800 BPI)
MT%CCT==:7B31                 ;CHARACTER COUNTER
MT%NSH==:1B32                 ;DATA MODE OR DENSITY NOT SUPPORTED BY HARDWARE


;DEVICE DATA MODES

.DMASC==:1                    ;ASCII
.DMIMG==:10                   ;IMAGE
.DMIMB==:13                   ;IMAGE BINARY
.DMBIN==:14                   ;BINARY
```

```
;DEFINED PSI CHANNELS

RADIX 5+5

.ICAOV==:6                      ;ARITHMETIC OVERFLOW
.ICFOV==:7                      ;FLOATING OVERFLOW
.ICPOV==:9                      ;PDL OVERFLOW
.ICEOF==:10                     ;END OF FILE
.ICDAE==:11                     ;DATA ERROR
.ICQTA==:12                     ;QUOTA/DISK EXCEEDED
.ICTOD==:14                     ;TIME OF DAY (NOT IMPLEMENTED)
.ICILI==:15                     ;ILLEG INSTRUCTION
.ICIRD==:16                     ;ILLEGAL READ
.ICIWR==:17                     ;ILLEGAL WRITE
.ICIEX==:18                     ;ILLEGAL EXECUTE (NOT IMPLEMENTED)
.ICIFT==:19                     ;INFERIOR FORK TERMINATION
.ICMSE==:20                     ;MACHINE SIZE EXCEEDED
.ICTRU==:21                     ;TRAP TO USER (NOT IMPLEMENTED)
.ICNXP==:22                     ;NONEXISTENT PAGE REFERENCED
```

```
;TERMINAL TYPE NUMBERS

.TT33==:0                        ;MODEL 33
.TT35==:1                        ;MODEL 35
.TT37==:2                        ;MODEL 37
.TTEXE==:3                       ;EXECUPORT
.TTDEF==:^D8                     ;DEFAULT
.TTIDL==:^D9                     ;IDEAL
.TTV05==:^D10                    ;VT05
.TTV50==:^D11                    ;VT50
.TTL30==:^D12                    ;LA30
.TTG40==:^D13                    ;GT40
.TTL36==:^D14                    ;LA36
.TTV52==:^D15                    ;VT52
.TT100==:^D16                    ;VT100
.TTL38==:^D17                    ;LA38
.TT120==:^D18                    ;LA120
.TT125==:^D35                    ;VT125
.TTK10==:^D36                    ;VK100 - GIGI

;DEFINED TERMINAL CODES

.TICBK==:0                       ;BREAK
.TICCA==:1                       ;^A
.TICCB==:2                       ;^B
.TICCC==:3                       ;^C
.TICCD==:4                       ;^D
.TICCE==:5                       ;^E
.TICCF==:6                       ;^F
.TICCG==:7                       ;^G
.TICCH==:8                       ;^H
.TICCI==:9                       ;^I
.TICCJ==:10                      ;^J
.TICCK==:11                      ;^K
.TICCL==:12                      ;^L
.TICCM==:13                      ;^M
.TICCN==:14                      ;^N
.TICCO==:15                      ;^O
.TICCP==:16                      ;^P
.TICCQ==:17                      ;^Q
.TICCR==:18                      ;^R
.TICCS==:19                      ;^S
.TICCT==:20                      ;^T
.TICCU==:21                      ;^U
.TICCV==:22                      ;^V
.TICCW==:23                      ;^W
.TICCX==:24                      ;^X
.TICCY==:25                      ;^Y
.TICCZ==:26                      ;^Z
.TICES==:27                      ;ESC
.TICRB==:28                      ;RUBOUT
.TICSP==:29                      ;SPACE
.TICRF==:30                      ;CARRIER OFF
.TICTI==:31                      ;TYPEIN
.TICTO==:32                      ;TYPEOUT
```

```
RADIX 8

;CAPABILITIES

SC%CTC==:1B0                        ;CONTROL-C
SC%GTB==:1B1                        ;GETAB
SC%MMN==:1B2                        ;MAP MONITOR
SC%LOG==:1B3                        ;LOGGING FUNCTIONS
SC%MPP==:1B4                        ;MAP PRIVILEGED PAGES
SC%SDV==:1B5                        ;SPECIAL DEVICES
SC%SCT==:1B6                        ;ASSIGN TTY AS CONTROLLING FOR FORK (SCTTY)

SC%SUP==:1B9                        ;SUPERIOR ACCESS

SC%FRZ==:1B17                       ;FREEZE ON TERMINATING CONDITIONS

SC%WHL==:1B18                       ;WHEEL
SC%OPR==:1B19                       ;OPERATOR
SC%CNF==:1B20                       ;CONFIDENTIAL INFORMATION ACCESS
SC%MNT==:1B21                       ;MAINTENANCE
SC%IPC==:1B22                       ;IPCF PRIVILEGES
SC%ENQ==:1B23                       ;ENQ/DEQ PRIVILEGES
SC%NWZ==:1B24             ;TOPS20AN ;NET WIZARD PRIVILEGES (ASNSQ, ETC.)
SC%NAS==:1B25             ;TOPS20AN ;NETWORK ABSOLUTE SOCKET PRIVILEGE
SC%DNA==:1B26                       ;DECNET ACCESS ALLOWED
SC%ANA==:1B27             ;TOPS20AN ;ARPANET ACCESS ALLOWED


;OUTMODED NAMES FOR BITS IN DIRECTORY MODE WORD - USE CD%XXX
;EQUIVALENTS

MD%FO==:CD%DIR                      ;FILES ONLY DIRECTORY
MD%SA==:CD%ANA                      ;STRING ACCOUNT ALLOWED
MD%RLM==:CD%RLM                     ;REPEAT LOGIN MESSAGE
```

```
;FDB DEFINITIONS

.FBHDR==:0                      ;HEADER WORD
        FB%LEN==:177B35         ;LENGTH OF THIS FDB
.FBCTL==:1                      ;FLAGS
        FB%TMP==:1B0            ;FILE IS TEMPORARY
        FB%PRM==:1B1            ;FILE IS PERMANENT
        FB%NEX==:1B2            ;FILE DOES NOT HAVE AN EXTENSION YET
        FB%DEL==:1B3            ;FILE IS DELETED
        FB%NXF==:1B4            ;FILE IS NONEXISTENT
        FB%LNG==:1B5            ;FILE IS A LONG FILE
        FB%SHT==:1B6            ;FILE HAS COMPRESSED PAGE TABLE
        FB%DIR==:1B7            ;FILE IS A DIRECTORY FILE
        FB%NOD==:1B8            ;FILE IS NOT TO BE DUMPED BY BACKUP SYSTEM
        FB%BAT==:1B9            ;FILE HAS AT LEAST ONE BAD PAGE IN IT
        FB%SDR==:1B10           ;THIS DIRECTORY HAS SUBDIRECTORIES
        FB%ARC==:1B11           ; File has archive status
        FB%INV==:1B12           ; File is invisible
        FB%OFF==:1B13           ; File is offline
        FB%FCF==:17B17          ;FILE CLASS FIELD
          .FBNRM==:0            ;NON-RMS
          .FBRMS==:1            ;RMS FILES
        FB%NDL==:1B18           ;FILE CANNOT BE DELETED
        FB%WNC==:1B19           ;LAST WRITE NOT CLOSED
.FBEXL==:2                      ;LINK TO FDB OF NEXT EXTENSION
.FBADR==:3                      ;DISK ADDRESS OF INDEX BLOCK
.FBPRT==:4                      ;PROTECTION OF THE FILE
.FBCRE==:5                      ;TIME AND DATE OF LAST WRITE
.FBUSE==:6                      ;LAST WRITER ,, AUTHOR (OBS)
.FBAUT==:6                      ;POINTER TO AUTHOR STRING
.FBGEN==:7                      ;GENERATION ,, DIR #
        FB%GEN==:777777B17      ;GENERATION NUMBER
.FBDRN==:7                      ;GENERATION ,, DIR #
        FB%DRN==:777777         ;DIR NUMBER
.FBACT==:10                     ;ACCOUNT
.FBBYV==:11                     ;RETENTION+BYTE SIZE+MODE ,, # OF PAGES
        FB%RET==:77B5           ;RETENTION COUNT
        FB%BSZ==:77B11          ;BYTE SIZE
        FB%MOD==:17B17          ;LAST OPENF MODE
        FB%PGC==:777777 ;PAGE COUNT
.FBSIZ==:12                     ;EOF POINTER
.FBCRV==:13                     ;TIME AND DATE OF CREATION OF FILE
.FBWRT==:14                     ;TIME AND DATE OF LAST USER WRITE
.FBREF==:15                     ;TIME AND DATE OF LAST NON-WRITE ACCESS
.FBCNT==:16                     ;# OF WRITES ,, # OF REFERENCES
.FBBK0==:17                     ;BACKUP WORDS (5)
.FBBK1==:20
.FBBK2==:21
.FBBBT==:22                     ; Bits,,#pages in offline file
        AR%RAR==:1B1            ; Request archive by user
        AR%RIV==:1B2            ; Request invol migration by system
        AR%NDL==:1B3            ; Do not delete contents of file when archived
        AR%NAR==:1B4            ; Please don't migrate this file
        AR%EXM==:1B5            ; File exempt from migration
        AR%1ST==:1B6            ; 1st pass of archive/collection run complete
        AR%RFL==:1B7            ; Retrieve failed
        AR%WRN==:1B8            ; USER WARNED OF APPROACHING EXPIRATION
        AR%RSN==:7B17           ; Reason pushed offline
          .AREXP==:1            ; File expired
          .ARARR==:2            ; Archive was requested
          .ARRIR==:3            ; Migration was requested
        AR%PSZ==:777777 ; RH is pg count when file went offline
```

```
.FBNET==:23                      ; On-line expiration date/interval
.FBUSW==:24                      ;USER SETTABLE WORD
.FBGNL==:25                      ;LINK TO NEXT GENERATION FILE
.FBNAM==:26                      ;POINTER TO NAME BLOCK
.FBEXT==:27                      ;POINTER TO EXTENSION BLOCK
.FBLWR==:30                      ;POINTER TO LAST WRITER STRING
.FBTDT==:31                      ; Archive or collection date & time
.FBFET==:32                      ; Offline expiration date/interval
.FBTP1==:33                      ; Tape ID for run 1 tape
.FBSS1==:34                      ; Saveset #,,Tape file # for run 1 tape
.FBTP2==:35                      ; Tape ID for run 2 tape
.FBSS2==:36                      ; Saveset #,,Tape file # for run 2 tape

.FBLN0==:30                      ;LENGTH OF VERSION 0 FDB
.FBLN1==:31                      ;LENGTH OF VERSION 1 FDB
.FBLXT==:37                      ; Minimum length for archive/virtual dsk sys
.FBLEN==:37                      ;LENGTH OF THE FDB
```

```
;CARD READER DEFINITIONS

.CRILC==:"\"                    ;ILLEGAL CHARACTER CODE

;A WORD IS DISTINGUISHED FROM A BYTE POINTER BY THE VALUE 5 IN BITS 0-2
;USE THESE DEFINITIONS TO TEST FOR A NUMBER AS FOLLOWS:
;       LOAD AC,NMFLG,LOC
;       CAIE AC,NUMVAL

NMFLG==:7B2
NUMVAL==:5


;MAGTAPE LABEL TYPES

.LTUNL==:1                      ;UNLABELED
.LTANS==:2                      ;ANSI STANDARD
.LTEBC==:3                      ;EBCDIC
.LTT20==:4                      ;TOPS-20
.LTMAX==:4                      ;MAXIMUM LABEL TYPE

;MAGTAPE LABEL STATES

.LSUNL==:0                      ;UNLABELLED VOLUME
.LSPRI==:1                      ;PRIVATE VOLUME
.LSSCR==:2                      ;SCRATCH VOLUME
.LSUSC==:3                      ;USER SCRATCH VOLUME

; MAGTAPE DRIVE TYPES

.TMDR9==:1                      ;9-TRACK
.TMDR7==:2                      ;7-TRACK
.TMDMX==:2                      ;MAXIMUM DRIVE-TYPE VALUE


;DEFINITIONS FOR COMMUNICATIONS PROTOCOLS

;DEFINE THE SUPPORTED PROTOCOL TYPES

.VN20F==:0                      ;RSX20F PROTOCOL
.VNMCB==:1                      ;MCB DECNET PROTOCOL
.VND60==:2                      ;DN60 PROTOCOL
.VNDDC==:2                      ;DDCMP PROTOCOL
.VNMOP==:3                      ;MOP (DDCMP MAINTENANCE) MODE
.VNCNL==:4                      ;CONTROLLER LOOPBACK
.VNCBL==:5                      ;CABLE LOOPBACK

;DEFINE BITS USED WHEN RELOADING AN -11

RM%ROM==:1B0                    ;IF SET, ACTIVATE ROM
```

```
;***********************************************
;GENERAL FIELD AND VALUE DEFINITIONS
;USED BY TOPS20AN JSYS'S
;***********************************************

;STATES OF A CONNECTION IN ARPANET NCP
; RETURNED IN B0-B3 OF GDSTS ON A NET CONNECTION
; ALSO AVAILABLE IN A GETAB, BUT THAT'S NOT THE PREFERRED WAY
; TO READ THEM, IF YOU HAVE A JFN FOR THE CONNECTION.

.NSCZD==:01                     ;CLOSED
.NSPND==:02                     ;PENDING
.NSLSN==:03                     ;LISTENING
.NSRCR==:04                     ;REQUEST FOR CONNECTION RECEIVED
.NSCW1==:05                     ;CLOSE WAIT SUB ONE (NCP CLOSE)
.NSRCS==:06                     ;REQUEST FOR CONNECTION SENT
.NSOPN==:07                     ;OPENED
.NSCSW==:10                     ;CLOSE WAIT (NCP CLOSE)
.NSDTW==:11                     ;FINAL DATA WAIT
.NSRF1==:12                     ;RFNM WAIT SUB ONE (NORMAL NCP CLOSE)
.NSCZW==:13                     ;CLOSE WAIT (PROGRAM CLOSE)
.NSRF2==:14                     ;RFNM WAIT SUB TWO (UNEXPECTED NCP CLOSE)
.NSFRE==:16                     ;FREE

;HOST STATUS BITS

HS%UP==1B0                      ;HOST IS UP
HS%VAL==1B1                     ;VALID STATUS
HS%DAY==7B4                     ;DAY WHEN UP IF DOWN
HS%HR==37B9                     ;HOUR
HS%MIN==17B13                   ;5 MIN INTERVAL
HS%RSN==17B17                   ;REASON
HS%SRV==1B18                    ;HOST IS SERVER
HS%USR==1B19                    ;HOST IS USER
HS%NCK==1B20                    ;HOST NAME STRING WAS NICKNAME
HS%STY==77B26                   ;SYSTEM TYPE MASK
HS%NEW==1B27                    ;HOST DOES NEW PROTOCOL
HS%NAM==1B28                    ;HOST HAS NAME

.HS10X==1B26                    ;TENEX
.HSITS==2B26                    ;ITS
.HSDEC==3B26                    ;TOPS-10
.HSTIP==4B26                    ;TIP
.HSMTP==5B26                    ;MTIP
.HSELF==6B26                    ;ELF
.HSANT==7B26                    ;ANTS
.HSMLT==10B26                   ;MULTICS
.HST20==11B26                   ;TOPS-20
.HSUNX==12B26                   ;UNIX
```

```
;ERROR CODE DEFINITIONS

        .ERBAS==:600000         ;BASE VALUE FOR ALL ERROR CODES

DEFINE .ERCOD <

.ERR (10,LGINX1,<Invalid account identifier>)
.ERR (11,LGINX2,<Directory is "files-only" and cannot be logged in to>)
.ERR (12,LGINX3,<Internal format of directory is incorrect>)
.ERR (13,LGINX4,<Invalid password>)
.ERR (14,LGINX5,<Job is already logged in>)
.ERR (20,CRJBX1,<Invalid parameter or function bit combination>)
.ERR (21,CRJBX2,<Illegal for created job to enter MINI-EXEC>)
.ERR (22,CRJBX3,<Reserved>)
.ERR (23,CRJBX4,<Terminal is not available>)
.ERR (24,CRJBX5,<Unknown name for LOGIN>)
.ERR (25,CRJBX6,<Insufficient system resources>)
.ERR (26,CRJBX7,<Reserved>)
.ERR (35,LOUTX1,<Illegal to specify job number when logging out own job>)
.ERR (36,LOUTX2,<Invalid job number>)
.ERR (45,CACTX1,<Invalid account identifier>)
.ERR (46,CACTX2,<Job is not logged in>)
.ERR (50,EFCTX1,<WHEEL or OPERATOR capability required>)
.ERR (51,EFCTX2,<Entry cannot be longer than 64 words>)
.ERR (52,EFCTX3,<Fatal error when accessing FACT file>)
.ERR (55,GJFX1,<Desired JFN invalid>)
.ERR (56,GJFX2,<Desired JFN not available>)
.ERR (57,GJFX3,<No JFN available>)
.ERR (60,GJFX4,<Invalid character in filename>)
.ERR (61,GJFX5,<Field cannot be longer than 39 characters>)
.ERR (62,GJFX6,<Device field not in a valid position>)
.ERR (63,GJFX7,<Directory field not in a valid position>)
.ERR (64,GJFX8,<Directory terminating delimiter is not preceded by a valid
      beginning delimiter>)
.ERR (65,GJFX9,<More than one name field is not allowed>)
.ERR (66,GJFX10,<Generation number is not numeric>)
.ERR (67,GJFX11,<More than one generation number field is not allowed>)
.ERR (70,GJFX12,<More than one account field is not allowed>)
.ERR (71,GJFX13,<More than one protection field is not allowed>)
.ERR (72,GJFX14,<Invalid protection>)
.ERR (73,GJFX15,<Invalid confirmation character>)
.ERR (74,GJFX16,<No such device>)
.ERR (75,GJFX17,<No such directory name>)
.ERR (76,GJFX18,<No such filename>)
.ERR (77,GJFX19,<No such file type>)
.ERR (100,GJFX20,<No such generation number>)
.ERR (101,GJFX21,<File was expunged>)
.ERR (102,GJFX22,<Insufficient system resources (Job Storage Block full)>)
.ERR (103,GJFX23,<Exceeded maximum number of files per directory>)
.ERR (104,GJFX24,<File not found>)
.ERR (107,GJFX27,<File already exists (new file required)>)
.ERR (110,GJFX28,<Device is not on line>)
.ERR (111,GJFX29,<Device is not available to this job>)
.ERR (112,GJFX30,<Account is not numeric>)
.ERR (113,GJFX31,<Invalid wildcard designator>)
.ERR (114,GJFX32,<No files match this specification>)
.ERR (115,GJFX33,<Filename was not specified>)
.ERR (116,GJFX34,<Invalid character "?" in file specification>)
.ERR (117,GJFX35,<Directory access privileges required>)
.ERR (120,OPNX1,<File is already open>)
.ERR (121,OPNX2,<File does not exist>)
.ERR (122,OPNX3,<Read access required>)
.ERR (123,OPNX4,<Write access required>)
```

```
.ERR (124,OPNX5,<Execute access required>)
.ERR (125,OPNX6,<Append access required>)
.ERR (126,OPNX7,<Device already assigned to another job>)
.ERR (127,OPNX8,<Device is not on line>)
.ERR (130,OPNX9,<Invalid simultaneous access>)
.ERR (131,OPNX10,<Entire file structure full>)
.ERR (133,OPNX12,<List access required>)
.ERR (134,OPNX13,<Invalid access requested>)
.ERR (135,OPNX14,<Invalid mode requested>)
.ERR (136,OPNX15,<Read/write access required>)
.ERR (137,OPNX16,<File has bad index block>)
.ERR (140,OPNX17,<No room in job for long file page table>)
.ERR (141,OPNX18,<Unit Record Devices are not available>)
.ERR (142,OPNX19,<IMP is not up>) ;TOPS20AN
.ERR (143,OPNX20,<Host is not up>) ;TOPS20AN
.ERR (144,OPNX21,<Connection refused>) ;TOPS20AN
.ERR (145,OPNX22,<Connection byte size does not match>) ;TOPS20AN
.ERR (150,DESX1,<Invalid source/destination designator>)
.ERR (151,DESX2,<Terminal is not available to this job>)
.ERR (152,DESX3,<JFN is not assigned>)
.ERR (153,DESX4,<Invalid use of terminal designator or string pointer>)
.ERR (154,DESX5,<File is not open>)
.ERR (155,DESX6,<Device is not a terminal>)
.ERR (156,DESX7,<Illegal use of parse-only JFN or output
        wildcard-designators>)
.ERR (157,DESX8,<File is not on disk>)
.ERR (160,CLSX1,<File is not open>)
.ERR (161,CLSX2,<File cannot be closed by this process>)
.ERR (165,RJFNX1,<File is not closed>)
.ERR (166,RJFNX2,<JFN is being used to accumulate filename>)
.ERR (167,RJFNX3,<JFN is not accessible by this process>)
.ERR (170,DELFX1,<Delete access required>)
.ERR (175,SFPTX1,<File is not open>)
.ERR (176,SFPTX2,<Illegal to reset pointer for this file>)
.ERR (177,SFPTX3,<Invalid byte number>)
.ERR (200,CNDIX1,<Invalid password>)
.ERR (202,CNDIX3,<Invalid directory number>)
.ERR (204,CNDIX5,<Job is not logged in>)
.ERR (210,SFBSX1,<Illegal to change byte size for this opening of file>)
.ERR (211,SFBSX2,<Invalid byte size>)
.ERR (215,IOX1,<File is not opened for reading>)
.ERR (216,IOX2,<File is not opened for writing>)
.ERR (217,IOX3,<File is not open for random access>)
.ERR (220,IOX4,<End of file reached>)
.ERR (221,IOX5,<Device or data error>)
.ERR (222,IOX6,<Illegal to write beyond absolute end of file>)
.ERR (240,PMAPX1,<Invalid access requested>)
.ERR (241,PMAPX2,<Invalid use of PMAP>)
.ERR (245,SPACX1,<Invalid access requested>)
.ERR (250,FRKHX1,<Invalid process handle>)
.ERR (251,FRKHX2,<Illegal to manipulate a superior process>)
.ERR (252,FRKHX3,<Invalid use of multiple process handle>)
.ERR (253,FRKHX4,<Process is running>)
.ERR (254,FRKHX5,<Process has not been started>)
.ERR (255,FRKHX6,<All relative process handles in use>)
.ERR (260,SPLFX1,<Process is not inferior or equal to self>)
.ERR (261,SPLFX2,<Process is not inferior to self>)
.ERR (262,SPLFX3,<New superior process is inferior to intended inferior>)
.ERR (267,GTABX1,<Invalid table number>)
.ERR (270,GTABX2,<Invalid table index>)
.ERR (271,GTABX3,<GETAB capability required>)
.ERR (273,RUNTX1,<Invalid process handle -3 or -4>)
.ERR (275,STADX1,<WHEEL or OPERATOR capability required>)
```

```
.ERR (276,STADX2,<Invalid date or time>)
.ERR (300,ASNDX1,<Device is not assignable>)
.ERR (301,ASNDX2,<Illegal to assign this device >)
.ERR (302,ASNDX3,<No such device>)
.ERR (320,ATACX1,<Invalid job number>)
.ERR (321,ATACX2,<Job already attached>)
.ERR (322,ATACX3,<Incorrect user number>)
.ERR (323,ATACX4,<Invalid password>)
.ERR (324,ATACX5,<This job has no controlling terminal>)
.ERR (332,STDVX1,<No such device>)
.ERR (335,DEVX1,<Invalid device designator>)
.ERR (336,DEVX2,<Device already assigned to another job>)
.ERR (337,DEVX3,<Device is not on line>)
.ERR (345,MNTX1,<Internal format of directory is incorrect>)
.ERR (346,MNTX2,<Device is not on line>)
.ERR (347,MNTX3,<Device is not mountable>)
.ERR (350,TERMX1,<Invalid terminal code>)
.ERR (351,TLNKX1,<Illegal to set remote to object before object to remote>)
.ERR (352,ATIX1,<Invalid software interrupt channel number>)
.ERR (353,ATIX2,<Control-C capability required>)
.ERR (356,TLNKX2,<Link was not received within 15 seconds>)
.ERR (357,TLNKX3,<Links full>)
.ERR (360,TTYX1,<Device is not a terminal>)
.ERR (361,RSCNX1,<Overflowed rescan buffer, input string truncated>)
.ERR (362,RSCNX2,<Invalid function code>)
.ERR (363,CFRKX3,<Insufficient system resources>)
.ERR (365,KFRKX1,<Illegal to kill top level process>)
.ERR (366,KFRKX2,<Illegal to kill self>)
.ERR (367,RFRKX1,<Processes are not frozen>)
.ERR (370,HFRKX1,<Illegal to halt self with HFORK>)
.ERR (371,GFRKX1,<Invalid process handle>)
.ERR (373,GETX1,<Invalid save file format>)
.ERR (374,GETX2,<System Special Pages Table full>)
.ERR (375,TFRKX1,<Undefined function code>)
.ERR (376,TFRKX2,<Unassigned fork handle or not immediate inferior>)
.ERR (377,SFRVX1,<Invalid position in entry vector>)
.ERR (407,NOUTX1,<Radix is not in range 2 to 36 >)
.ERR (410,NOUTX2,<Column overflow>)
.ERR (411,TFRKX3,<Fork(s) not frozen>)
.ERR (414,IFIXX1,<Radix is not in range 2 to 10>)
.ERR (415,IFIXX2,<First nonspace character is not a digit>)
.ERR (416,IFIXX3,<Overflow (number is greater than 2**35 )>)
.ERR (424,GFDBX1,<Invalid displacement>)
.ERR (425,GFDBX2,<Invalid number of words>)
.ERR (426,GFDBX3,<List access required>)
.ERR (430,CFDBX1,<Invalid displacement>)
.ERR (431,CFDBX2,<Illegal to change specified bits>)
.ERR (432,CFDBX3,<Write or owner access required>)
.ERR (433,CFDBX4,<Invalid value for specified bits>)
.ERR (440,DUMPX1,<Command list error>)
.ERR (441,DUMPX2,<JFN is not open in dump mode>)
.ERR (442,DUMPX3,<Address error (too big or crosses end of memory)>)
.ERR (443,DUMPX4,<Access error (cannot read or write data in memory)>)
.ERR (450,RNAMX1,<Files are not on same device>)
.ERR (451,RNAMX2,<Destination file expunged>)
.ERR (452,RNAMX3,<Write or owner access to destination file required>)
.ERR (453,RNAMX4,<Quota exceeded in destination of rename>)
.ERR (454,BKJFX1,<Illegal to back up terminal pointer twice>)
.ERR (460,TIMEX1,<Time cannot be greater than 24 hours>)
.ERR (461,ZONEX1,<Time zone out of range>)
.ERR (462,ODTNX1,<Time zone must be USA or Greenwich>)
.ERR (464,DILFX1,<Invalid date format>)
.ERR (465,TILFX1,<Invalid time format>)
```

```
.ERR (466,DATEX1,<Year out of range>)
.ERR (467,DATEX2,<Month is not less than 12>)
.ERR (470,DATEX3,<Day of month too large>)
.ERR (471,DATEX4,<Day of week is not less than 7>)
.ERR (472,DATEX5,<Date out of range>)
.ERR (473,DATEX6,<System date and time are not set>)
.ERR (516,SMONX1,<WHEEL or OPERATOR capability required>)
.ERR (530,SACTX1,<File is not on multiple-directory device>)
.ERR (531,SACTX2,<Insufficient system resources (Job Storage Block full)>)
.ERR (532,SACTX3,<Directory requires numeric account>)
.ERR (533,SACTX4,<Write or owner access required>)
.ERR (540,GACTX1,<File is not on multiple-directory device>)
.ERR (541,GACTX2,<File expunged>)
.ERR (544,FFUFX1,<File is not open>)
.ERR (545,FFUFX2,<File is not on multiple-directory device>)
.ERR (546,FFUFX3,<No used page found>)
.ERR (555,DSMX1,<File(s) not closed>)
.ERR (560,RDDIX1,<Illegal to read directory for this device>)
.ERR (570,SIRX1,<Table address is not greater than 20>)
.ERR (600,SSAVX1,<Illegal to save files on this device>)
.ERR (601,SSAVX2,<Page count (left half of table entry) must be negative>)
.ERR (610,SEVEX1,<Entry vector length is not less than 1000>)
.ERR (614,WHELX1,<WHEEL or OPERATOR capability required>)
.ERR (615,CAPX1,<WHEEL or OPERATOR capability required>)
.ERR (617,PEEKX2,<Read access failure on monitor page>)
.ERR (620,CRDIX1,<WHEEL or OPERATOR capability required>)
.ERR (621,CRDIX2,<Illegal to change number of old directory>)
.ERR (622,CRDIX3,<Insufficient system resources (Job Storage Block full)>)
.ERR (623,CRDIX4,<Superior directory full>)
.ERR (624,CRDIX5,<Directory name not given>)
.ERR (626,CRDIX7,<File(s) open in directory>)
.ERR (640,GTDIX1,<WHEEL or OPERATOR capability required>)
.ERR (641,GTDIX2,<Invalid directory number>)
.ERR (650,FLINX1,<First character is not blank or numeric>)
.ERR (651,FLINX2,<Number too small>)
.ERR (652,FLINX3,<Number too large>)
.ERR (653,FLINX4,<Invalid format>)
.ERR (660,FLOTX1,<Column overflow in field 1 or 2>)
.ERR (661,FLOTX2,<Column overflow in field 3>)
.ERR (662,FLOTX3,<Invalid format specified>)
.ERR (670,HPTX1,<Undefined clock number>)
.ERR (700,FDFRX1,<Not a multiple-directory device>)
.ERR (701,FDFRX2,<Invalid directory number>)
.ERR (704,GTHSX1,<Unknown host number>) ;TOPS20AN
.ERR (705,GTHSX2,<No number for that host name>) ;TOPS20AN
.ERR (707,GTHSX3,<No string for that Host number>) ;TOPS20AN
.ERR (710,ATNX1,<Invalid receive JFN>) ;TOPS20AN
.ERR (711,ATNX2,<Receive JFN not opened for read>) ;TOPS20AN
.ERR (712,ATNX3,<Receive JFN not open>) ;TOPS20AN
.ERR (713,ATNX4,<Receive JFN is not a NET connection>) ;TOPS20AN
.ERR (714,ATNX5,<Receive JFN has been used>) ;TOPS20AN
.ERR (715,ATNX6,<Receive connection refused>) ;TOPS20AN
.ERR (716,ATNX7,<Invalid send JFN>) ;TOPS20AN
.ERR (717,ATNX8,<Send JFN not opened for write>) ;TOPS20AN
.ERR (720,ATNX9,<Send JFN not open>) ;TOPS20AN
.ERR (721,ATNX10,<Send JFN is not a NET connection>) ;TOPS20AN
.ERR (722,ATNX11,<Send JFN has been used>) ;TOPS20AN
.ERR (723,ATNX12,<Send connection refused>) ;TOPS20AN
.ERR (724,ATNX13,<Insufficient system resources (No NVT's)>) ;TOPS20AN
.ERR (727,CVHST1,<No string for that Host number>) ;TOPS20AN
.ERR (730,CVSKX1,<Invalid network JFN>) ;TOPS20AN
.ERR (731,CVSKX2,<Local socket invalid in this context>) ;TOPS20AN
.ERR (732,SNDIX1,<Invalid message size>) ;TOPS20AN
```

```
.ERR (733,SNDIX2,<Insufficient system resources (No buffers available)>)
     ;TOPS20AN
.ERR (734,SNDIX3,<Illegal to specify NCP links 0 - 72>) ;TOPS20AN
.ERR (735,SNDIX4,<Invalid header value for this queue>) ;TOPS20AN
.ERR (736,SNDIX5,<IMP down>) ;TOPS20AN
.ERR (737,NTWZX1,<NET WIZARD capability  required>) ;TOPS20AN
.ERR (740,ASNSX1,<Insufficient system resources (All special queues in use)>)
     ;TOPS20AN
.ERR (741,ASNSX2,<Link(s) assigned to another special queue>) ;TOPS20AN
.ERR (742,SQX1,<Special network queue handle out of range>) ;TOPS20AN
.ERR (743,SQX2,<Special network queue not assigned>) ;TOPS20AN
.ERR (746,GTNCX1,<Invalid network JFN>) ;TOPS20AN
.ERR (747,GTNCX2,<Invalid or inactive NVT>) ;TOPS20AN
.ERR (750,RNAMX5,<Destination file is not closed>)
.ERR (751,RNAMX6,<Destination file has bad page table>)
.ERR (752,RNAMX7,<Source file expunged>)
.ERR (753,RNAMX8,<Write or owner access to source file required>)
.ERR (754,RNAMX9,<Source file is nonexistent>)
.ERR (755,RNMX10,<Source file is not closed>)
.ERR (756,RNMX11,<Source file has bad page table>)
.ERR (757,RNMX12,<Illegal to rename to self>)
.ERR (760,GJFX36,<Internal format of directory is incorrect>)
.ERR (770,ILINS1,<Undefined operation code>)
.ERR (771,ILINS2,<Undefined JSYS>)
.ERR (772,ILINS3,<UUO simulation facility not available>)
.ERR (1000,CRLNX1,<Logical name is not defined>)
.ERR (1001,INLNX1,<Index is beyond end of logical name table>)
.ERR (1002,LNSTX1,<No such logical name>)
.ERR (1003,MLKBX1,<Lock facility already in use>)
.ERR (1004,MLKBX2,<Too many pages to be locked>)
.ERR (1005,MLKBX3,<Page is not available>)
.ERR (1006,MLKBX4,<Illegal to remove previous contents of user map>)
.ERR (1007,VBCX1,<Display data area not locked in core>)
.ERR (1010,RDTX1,<Invalid string pointer>)
.ERR (1011,GFKSX1,<Area too small to hold process structure>)
.ERR (1013,GTJIX1,<Invalid index>)
.ERR (1014,GTJIX2,<Invalid terminal line number>)
.ERR (1015,GTJIX3,<Invalid job number>)
.ERR (1016,IPCFX1,<Length of packet descriptor block cannot be less than 4>)
.ERR (1017,IPCFX2,<No message for this PID>)
.ERR (1020,IPCFX3,<Data too long for user's buffer>)
.ERR (1021,IPCFX4,<Receiver's PID invalid>)
.ERR (1022,IPCFX5,<Receiver's PID disabled>)
.ERR (1023,IPCFX6,<Send quota exceeded>)
.ERR (1024,IPCFX7,<Receiver quota exceeded>)
.ERR (1025,IPCFX8,<IPCF free space exhausted>)
.ERR (1026,IPCFX9,<Sender's PID invalid>)
.ERR (1027,IPCF10,<WHEEL capability required>)
.ERR (1030,IPCF11,<WHEEL or IPCF capability required>)
.ERR (1031,IPCF12,<No free PID's available>)
.ERR (1032,IPCF13,<PID quota exceeded>)
.ERR (1033,IPCF14,<No PID's available to this job>)
.ERR (1034,IPCF15,<No PID's available to this process>)
.ERR (1035,IPCF16,<Receive and message data modes do not match>)
.ERR (1036,IPCF17,<Argument block too small>)
.ERR (1037,IPCF18,<Invalid MUTIL JSYS function>)
.ERR (1040,IPCF19,<No PID for [SYSTEM] INFO>)
.ERR (1041,IPCF20,<Invalid process handle>)
.ERR (1042,IPCF21,<Invalid job number>)
.ERR (1043,IPCF22,<Invalid software interrupt channel number>)
.ERR (1044,IPCF23,<[SYSTEM] INFO already exists>)
.ERR (1045,IPCF24,<Invalid message size>)
.ERR (1046,IPCF25,<PID does not belong to this job>)
```

```
.ERR (1047,IPCF26,<PID does not belong to this process>)
.ERR (1050,IPCF27,<PID is not defined>)
.ERR (1051,IPCF28,<PID not accessible by this process>)
.ERR (1052,IPCF29,<PID already being used by another process>)
.ERR (1053,IPCF30,<Job is not logged in>)
.ERR (1054,GNJFX1,<No more files in this specification>)
.ERR (1055,ENQX1,<Invalid function>)
.ERR (1056,ENQX2,<Level number too small>)
.ERR (1057,ENQX3,<Request and lock level numbers do not match>)
.ERR (1060,ENQX4,<Number of pool and lock resources do not match>)
.ERR (1061,ENQX5,<Lock already requested>)
.ERR (1062,ENQX6,<Requested locks are not all locked>)
.ERR (1063,ENQX7,<No ENQ on this lock>)
.ERR (1064,ENQX8,<Invalid access change requested>)
.ERR (1065,ENQX9,<Invalid number of blocks specified>)
.ERR (1066,ENQX10,<Invalid argument block length>)
.ERR (1067,ENQX11,<Invalid software interrupt channel number>)
.ERR (1070,ENQX12,<Invalid number of resources requested>)
.ERR (1071,ENQX13,<Indirect or indexed byte pointer not allowed>)
.ERR (1072,ENQX14,<Invalid byte size>)
.ERR (1073,ENQX15,<ENQ/DEQ capability required>)
.ERR (1074,ENQX16,<WHEEL or OPERATOR capability required>)
.ERR (1075,ENQX17,<Invalid JFN>)
.ERR (1076,ENQX18,<Quota exceeded>)
.ERR (1077,ENQX19,<String too long>)
.ERR (1100,ENQX20,<Locked JFN cannot be closed>)
.ERR (1101,ENQX21,<Job is not logged in>)
.ERR (1102,IPCF31,<Invalid page number>)
.ERR (1103,IPCF32,<Page is not private>)
.ERR (1104,PMAPX3,<Illegal to move shared page into file>)
.ERR (1105,PMAPX4,<Illegal to move file page into process>)
.ERR (1106,PMAPX5,<Illegal to move special page into file>)
.ERR (1107,PMAPX6,<Disk quota exceeded>)
.ERR (1110,SNOPX1,<WHEEL or OPERATOR capability required>)
.ERR (1111,SNOPX2,<Invalid function>)
.ERR (1112,SNOPX3,<.SNPLC function must be first>)
.ERR (1113,SNOPX4,<Only one .SNPLC function allowed>)
.ERR (1114,SNOPX5,<Invalid page number>)
.ERR (1115,SNOPX6,<Invalid number of pages to lock>)
.ERR (1116,SNOPX7,<Illegal to define breakpoints after inserting them>)
.ERR (1117,SNOPX8,<Breakpoint is not set on instruction>)
.ERR (1120,SNOPX9,<No more breakpoints allowed>)
.ERR (1121,SNOP10,<Breakpoints already inserted>)
.ERR (1122,SNOP11,<Breakpoints not inserted>)
.ERR (1123,SNOP12,<Invalid format for program name symbol>)
.ERR (1124,SNOP13,<No such program name symbol>)
.ERR (1125,SNOP14,<No such symbol>)
.ERR (1126,SNOP15,<Not enough free pages for snooping>)
.ERR (1127,SNOP16,<Multiply defined symbol>)
.ERR (1130,IPCF33,<Invalid index into system PID table>)
.ERR (1131,SNOP17,<Breakpoint already defined>)
.ERR (1132,OPNX23,<Disk quota exceeded>)
.ERR (1133,GJFX37,<Input deleted>)
.ERR (1134,CRLNX2,<WHEEL or OPERATOR capability required>)
.ERR (1135,INLNX2,<Invalid function>)
.ERR (1136,LNSTX2,<Invalid function>)
.ERR (1137,ALCX1,<Invalid function>)
.ERR (1140,ALCX2,<WHEEL or OPERATOR capability required>)
.ERR (1141,ALCX3,<Device is not assignable>)
.ERR (1142,ALCX4,<Invalid job number>)
.ERR (1143,ALCX5,<Device already assigned to another job>)
.ERR (1144,SPLX1,<Invalid function>)
.ERR (1145,SPLX2,<Argument block too small>)
```

```
.ERR (1146,SPLX3,<Invalid device designator>)
.ERR (1147,SPLX4,<WHEEL or OPERATOR capability required>)
.ERR (1150,SPLX5,<Illegal to specify 0 as generation number for first file>)
.ERR (1151,CLSX3,<File still mapped>)
.ERR (1152,CRLNX3,<Invalid function>)
.ERR (1153,ALCX6,<Device assigned to user job, but will be given to allocator
      when released>)
.ERR (1154,CKAX1,<Argument block too small>)
.ERR (1155,CKAX2,<Invalid directory number>)
.ERR (1156,CKAX3,<Invalid access code>)
.ERR (1157,TIMX1,<Invalid function>)
.ERR (1160,TIMX2,<Invalid process handle>)
.ERR (1161,TIMX3,<Time limit already set>)
.ERR (1162,TIMX4,<Illegal to clear time limit>)
.ERR (1163,SNOP18,<Data page is not private or copy-on-write>)
.ERR (1164,GJFX38,<File not found because output-only device was specified>)
.ERR (1165,GJFX39,<Logical name loop detected>)
.ERR (1166,CRDIX8,<Invalid directory number>)
.ERR (1167,CRDIX9,<Internal format of directory is incorrect>)
.ERR (1170,CRDI10,<Maximum directory number exceeded;  index table needs
      expanding>)
.ERR (1171,DELDX1,<WHEEL or OPERATOR capability required>)
.ERR (1172,DELDX2,<Invalid directory number>)
.ERR (1173,GACTX3,<Internal format of directory is incorrect>)
.ERR (1174,DIAGX1,<Invalid function>)
.ERR (1175,DIAGX2,<Device is not assigned>)
.ERR (1176,DIAGX3,<Argument block too small>)
.ERR (1177,DIAGX4,<Invalid device type>)
.ERR (1200,DIAGX5,<WHEEL, OPERATOR, or MAINTENANCE capability required>)
.ERR (1201,DIAGX6,<Invalid channel command list>)
.ERR (1202,DIAGX7,<Illegal to do I/O across page boundary>)
.ERR (1203,DIAGX8,<No such device>)
.ERR (1204,DIAGX9,<Unit does not exist>)
.ERR (1205,DIAG10,<Subunit does not exist>)
.ERR (1206,SYEX1,<Unreasonable SPEAR block size>)
.ERR (1207,SYEX2,<No buffer space available for SPEAR>)
.ERR (1210,MTOX1,<Invalid function>)
.ERR (1211,IOX7,<Insufficient system resources (Job Storage Block full)>)
.ERR (1212,IOX8,<Monitor internal error>)
.ERR (1213,MTOX5,<Invalid hardware data mode for magnetic tape>)
.ERR (1214,DUMPX5,<No-wait dump mode not supported for this device>)
.ERR (1215,DUMPX6,<Dump mode not supported for this device>)
.ERR (1216,IOX9,<Function legal for sequential write only>)
.ERR (1217,CLSX4,<Device still active>)
.ERR (1220,MTOX2,<Record size was not set before I/O was done>)
.ERR (1221,MTOX3,<Function not legal in dump mode>)
.ERR (1222,MTOX4,<Invalid record size>)
.ERR (1223,MTOX6,<Invalid magnetic tape density>)
.ERR (1224,OPNX25,<Device is write locked>)
.ERR (1225,GJFX40,<Undefined attribute in file specification>)
.ERR (1226,MTOX7,<WHEEL or OPERATOR capability required>)
.ERR (1227,LOUTX3,<WHEEL or OPERATOR capability required>)
.ERR (1230,LOUTX4,<LOG capability required>)
.ERR (1231,CAPX2,<WHEEL, OPERATOR, or MAINTENANCE capability required>)
.ERR (1232,SSAVX3,<Insufficient system resources (Job Storage Block
      full)>)
.ERR (1233,SSAVX4,<Directory area of EXE file is more than one page>)
.ERR (1234,TDELX1,<Table is empty>)
.ERR (1235,TADDX1,<Table is full>)
.ERR (1236,TADDX2,<Entry is already in table>)
.ERR (1237,TLUKX1,<Internal format of table is incorrect>)
.ERR (1240,IOX10,<Record is longer than user requested>)
.ERR (1241,CNDIX2,<WHEEL or OPERATOR capability required>)
```

```
.ERR (1242,CNDIX4,<Invalid job number>)
.ERR (1243,CNDIX6,<Job is not logged in>)
.ERR (1244,SJBX1,<Invalid function>)
.ERR (1245,SJBX2,<Invalid magnetic tape density>)
.ERR (1246,SJBX3,<Invalid magnetic tape data mode>)
.ERR (1247,TMONX1,<Invalid TMON function>)
.ERR (1250,SMONX2,<Invalid SMON function>)
.ERR (1251,SJBX4,<Invalid job number>)
.ERR (1252,SJBX5,<Job is not logged in>)
.ERR (1253,SJBX6,<WHEEL or OPERATOR capability required>)
.ERR (1254,GTJIX4,<No such job>)
.ERR (1255,ILINS4,<UUO simulation is disabled>)
.ERR (1256,ILINS5,<RMS facility is not available>)
.ERR (1257,COMNX1,<Invalid COMND function code>)
.ERR (1260,COMNX2,<Field too long for internal buffer>)
.ERR (1261,COMNX3,<Command too long for internal buffer>)
.ERR (1262,COMNX4,<Invalid character in input>)
.ERR (1263,PRAX1,<Invalid PRARG function code>)
.ERR (1264,PRAX2,<No room in monitor data base for argument block>)
.ERR (1265,COMNX5,<Invalid string pointer argument>)
.ERR (1266,COMNX6,<Problem in indirect file>)
.ERR (1267,COMNX7,<Error in command>)
.ERR (1270,PRAX3,<PRARG argument block too large>)
.ERR (1271,CKAX4,<File is not on disk>)
.ERR (1272,GACCX1,<Invalid job number>)
.ERR (1273,GACCX2,<No such job>)
.ERR (1274,MTOX8,<Argument block too long>)
.ERR (1275,DBRKX1,<No interrupts in progress>)
.ERR (1276,SJPRX1,<Job is not logged in>)
.ERR (1277,GJFX41,<File name must not exceed 6 characters>)
.ERR (1300,GJFX42,<File type must not exceed 3 characters>)
.ERR (1301,GACCX3,<Confidential Information Access capability required>)
.ERR (1302,TIMEX2,<Downtime cannot be more than 7 days in the future>)
.ERR (1303,DELFX2,<File cannot be expunged because it is currently open>)
.ERR (1304,DELFX3,<System scratch area depleted; file not deleted>)
.ERR (1305,DELFX4,<Directory symbol table could not be rebuilt>)
.ERR (1306,DELFX5,<Directory symbol table needs rebuilding>)
.ERR (1307,DELFX6,<Internal format of directory is incorrect>)
.ERR (1310,DELFX7,<FDB formatted incorrectly; file not deleted>)
.ERR (1311,DELFX8,<FDB not found; file not deleted>)
.ERR (1312,FRKHX7,<Process page cannot exceed 777>)
.ERR (1313,DIRX1,<Invalid directory number>)
.ERR (1314,DIRX2,<Insufficient system resources>)
.ERR (1315,DIRX3,<Internal format of directory is incorrect>)
.ERR (1316,UFPGX1,<File is not open for write>)
.ERR (1317,LNGFX1,<Page table does not exist and file not open for write>)
.ERR (1320,IPCF34,<Cannot receive into an existing page>)
.ERR (1321,COMNX8,<Number base out of range 2-10>)
.ERR (1322,MTOX9,<Output still pending>)
.ERR (1323,MTOX10,<VFU or RAM file cannot be OPENed>)
.ERR (1324,MTOX11,<Data too large for buffers>)
.ERR (1325,MTOX12,<Input error or not all data read>)
.ERR (1326,MTOX13,<Argument block too small>)
.ERR (1327,MTOX14,<Invalid software interrupt channel number>)
.ERR (1330,SAVX1,<Illegal to save files on this device>)
.ERR (1331,MTOX15,<Device does not have Direct Access (programmable) VFU>)
.ERR (1332,MTOX16,<VFU or Translation Ram file must be on disk>)
.ERR (1333,LPINX1,<Invalid unit number>)
.ERR (1334,LPINX2,<WHEEL or OPERATOR capability required>)
.ERR (1335,LPINX3,<Illegal to load RAM or VFU while device is OPEN>)
.ERR (1336,MTOX17,<Device is not on line>)
.ERR (1337,LGINX6,<No more job slots available for logging-in>)
.ERR (1340,DESX9,<Invalid operation for this device>)
```

```
.ERR (1341,ACESX1,<Argument block too small>)
.ERR (1342,ACESX2,<Insufficient system resources>)
.ERR (1343,DSKOX1,<Channel number too large>)
.ERR (1344,DSKOX2,<Unit number too large>)
.ERR (1345,MSTRX1,<Invalid function>)
.ERR (1346,MSTRX2,<WHEEL or OPERATOR capability required>)
.ERR (1347,MSTRX3,<Argument block too small>)
.ERR (1350,MSTRX4,<Insufficient system resources>)
.ERR (1351,MSTRX5,<Drive is not on-line>)
.ERR (1352,MSTRX6,<Home blocks are bad>)
.ERR (1353,MSTRX7,<Invalid structure name>)
.ERR (1354,MSTRX8,<Could not get OFN for ROOT-DIRECTORY>)
.ERR (1355,MSTRX9,<Could not MAP ROOT-DIRECTORY>)
.ERR (1356,MSTX10,<ROOT-DIRECTORY bad>)
.ERR (1357,MSTX11,<Could not initialize Index Table>)
.ERR (1360,MSTX12,<Could not OPEN Bit Table File>)
.ERR (1361,MSTX13,<Backup copy of ROOT-DIRECTORY is bad>)
.ERR (1362,MSTX14,<Invalid channel number>)
.ERR (1363,MSTX15,<Invalid unit number>)
.ERR (1364,MSTX16,<Invalid controller number>)
.ERR (1365,DSKX01,<Invalid structure number>)
.ERR (1366,DSKX02,<Bit table is being initialized>)
.ERR (1367,DSKX03,<Bit table has not been initialized>)
.ERR (1370,DSKX04,<Bit table being initialized by another job>)
.ERR (1371,GFUSX1,<Invalid function>)
.ERR (1372,GFUSX2,<Insufficient system resources>)
.ERR (1373,SFUSX1,<Invalid function>)
.ERR (1374,SFUSX2,<Insufficient system resources>)
.ERR (1375,SFUSX3,<No such user name>)
.ERR (1376,RCDIX1,<Insufficient system resources>)
.ERR (1377,RCDIX2,<Invalid directory specification>)
.ERR (1400,RCDIX3,<Invalid structure name>)
.ERR (1401,RCDIX4,<Monitor internal error>)
.ERR (1402,RCUSX1,<Insufficient system resources>)
.ERR (1403,TDELX2,<Invalid table entry location>)
.ERR (1404,TIMX5,<Invalid software interrupt channel number>)
.ERR (1405,LSTRX1,<Process has not encountered any errors>)
.ERR (1406,SWJFX1,<Illegal to swap same JFN>)
.ERR (1407,MTOX18,<Invalid software interrupt channel number>)
.ERR (1410,OPNX26,<Illegal to open a string pointer>)
.ERR (1411,DELFX9,<File is not a directory file>)
.ERR (1412,CRDIX6,<Directory file is mapped>)
.ERR (1413,COMNX9,<End of input file reached>)
.ERR (1414,STYPX1,<Invalid terminal type>)
.ERR (1415,PMAPX7,<Illegal to map file on dismounted structure>)
.ERR (1416,DSKOX3,<Invalid structure number>)
.ERR (1417,DESX10,<Structure is dismounted>)
.ERR (1420,DSKOX4,<Invalid address type specified>)
.ERR (1421,MSTX17,<All units in a structure must be of the same type>)
.ERR (1422,MSTX18,<No more units in system>)
.ERR (1423,MSTX19,<Unit is already part of a mounted structure>)
.ERR (1424,MSTX20,<Data error reading HOME blocks>)
.ERR (1425,MSTX21,<Structure is not mounted>)
.ERR (1426,MSTX22,<Illegal to change specified bits>)
.ERR (1427,CRDI11,<Invalid terminating bracket on directory>)
.ERR (1430,MSTX23,<Could not write HOME blocks>)
.ERR (1431,ACESX3,<Password is required>)
.ERR (1432,ACESX4,<Function not allowed for another job>)
.ERR (1433,ACESX5,<No function specified for ACCES>)
.ERR (1434,STRX05,<No such user name>)
.ERR (1435,ACESX6,<Directory is not accessed>)
.ERR (1436,STRX01,<Structure is not mounted>)
.ERR (1437,STRX02,<Insufficient system resources>)
```

```
.ERR (1440,IOX11,<Quota exceeded>)
.ERR (1441,IOX12,<Insufficient system resources (Swapping space full)>)
.ERR (1442,STRX03,<No such directory name>)
.ERR (1443,STRX04,<Ambiguous directory specification>)
.ERR (1444,PPNX1,<Invalid PPN>)
.ERR (1445,PPNX2,<Structure is not mounted>)
.ERR (1446,PPNX3,<Insufficient system resources>)
.ERR (1447,PPNX4,<Invalid directory number>)
.ERR (1450,SPLX6,<No directory to write spooled files into>)
.ERR (1451,CRDI12,<Structure is not mounted>)
.ERR (1452,GFUSX3,<File expunged>)
.ERR (1453,GFUSX4,<Internal format of directory is incorrect>)
.ERR (1454,RNMX13,<Insufficient system resources>)
.ERR (1455,SJBX8,<Illegal to perform this function>)
.ERR (1456,DECRSV,<DEC reserved bits not zero>)
.ERR (1457,FFFFX1,<No free pages in file>)
.ERR (1460,WILDX1,<Second JFN cannot be wild>)
.ERR (1461,MSTX41,<Channel does not exist>)
.ERR (1462,MSTX42,<Controller does not exist>)
.ERR (1463,CIMXND,<Maximum memory driver nodes assigned>)
.ERR (1464,CINOND,<No LCS node slots availble>)
.ERR (1465,CIBDOF,<BAD BDT offset given >)
.ERR (1466,CINOFQ,<No CI free queue entries left>)
.ERR (1467,CINOPG,<No BDT page slots left>)
.ERR (1470,CINPTH,<Target CI LCS node is dead, no path to it>)
.ERR (1471,CIBDCD,<Bad CI op code>)
.ERR (1472,CIUNOP,<Undefined op code (in range but not yet defined>)
.ERR (1473,CINOND,<Dead LCS node>)
.ERR (1474,CILNER,<CI length error>)
.ERR (1475,LCBDBP,<Bad byte pointer passed to LCS>)
.ERR (1476,LCLNER,<LCS length error>)
.ERR (1477,LCNOND,<LCS No such node>)
.ERR (1500,SSAVX5,<Number of PDVs grew during save>)
.ERR (1501,CIBDFQ,<BAD CI FREE QUEUE>)
.ERR (1502,ATACX6,<Terminal is already attached to a job>)
.ERR (1503,ATACX7,<Illegal terminal number>)

; ERROR CODES 1504-1532 ARE AVAILABLE******

.ERR (1533,DSKOX5,<Invalid word count>)
.ERR (1534,DSKOX6,<Invalid buffer address>)
.ERR (1535,TIMX6,<Time has already passed>)
.ERR (1536,TIMX7,<No space available for a clock>)
.ERR (1537,TIMX8,<User clock allocation exceeded>)
.ERR (1540,TIMX9,<No such clock entry found>)
.ERR (1541,TIMX10,<No system date and time>)

.ERR (1550,SCTX1,<Invalid function code>)
.ERR (1551,SCTX2,<Terminal already in use as controlling terminal>)
.ERR (1552,SCTX3,<Illegal to redefine the job's controlling terminal>)
.ERR (1553,SCTX4,<SC%SCT capability required>)

.ERR (1554,PDVX01,<Address in .POADE must be as large as address in
     .POADR>)
.ERR (1555,PDVX02,<Addresses in .PODAT block must be in strict ascending
     order>)
.ERR (1556,PDVX03,<Address in .POADR must be a program data vector
     address>)
.ERR (1557,GETX4,<Illegal to relocate (via .GBASE) a multi-section exe
     file>)
.ERR (1560,GETX5,<Exe file directory entry specifies a section-crossing>)

; Note: Error codes are available here!
```

```
.ERR (1700,SFUSX4,<File expunged>)
.ERR (1701,SFUSX5,<Write or owner access required>)
.ERR (1702,SFUSX6,<No such user name>)
.ERR (1703,GETX3,<Illegal to overlay existing pages>)
.ERR (1704,FILX01,<File is not open>)
.ERR (1705,ARGX01,<Invalid password>)
.ERR (1706,CAPX3,<WHEEL capability required>)
.ERR (1707,CAPX4,<WHEEL or IPCF capability required>)
.ERR (1711,CAPX6,<ENQ/DEQ capability required>)
.ERR (1712,CAPX7,<Confidential Information Access Capability required>)
.ERR (1713,ARGX02,<Invalid function>)
.ERR (1714,ARGX03,<Illegal to change specified bits>)
.ERR (1715,ARGX04,<Argument block too small>)
.ERR (1716,ARGX05,<Argument block too long>)
.ERR (1717,ARGX06,<Invalid page number>)
.ERR (1720,ARGX07,<Invalid job number>)
.ERR (1721,ARGX08,<No such job>)
.ERR (1722,ARGX09,<Invalid byte size>)
.ERR (1723,ARGX10,<Invalid access requested>)
.ERR (1724,ARGX11,<Invalid directory number>)
.ERR (1725,ARGX12,<Invalid process handle>)
.ERR (1726,ARGX13,<Invalid software interrupt channel number>)
.ERR (1727,MONX01,<Insufficient system resources>)
.ERR (1730,MONX02,<Insufficient system resources (JSB full)>)
.ERR (1731,MONX03,<Monitor internal error>)
.ERR (1732,MONX04,<Insufficient system resources (Swapping space full)>)
.ERR (1733,ARGX14,<Invalid account identifier>)
.ERR (1734,ARGX15,<Job is not logged in>)
.ERR (1735,FILX02,<Write or owner access required>)
.ERR (1736,FILX03,<List access required>)
.ERR (1737,DEVX4,<Device is not assignable>)
.ERR (1740,FILX04,<File is not on multiple-directory device>)
.ERR (1741,ARGX16,<Password is required>)
.ERR (1742,ARGX17,<Invalid argument block length>)
.ERR (1743,ARGX18,<Invalid structure name>)
.ERR (1744,DEVX5,<No such device>)
.ERR (1745,DIRX4,<Invalid directory specification>)
.ERR (1746,FILX05,<File expunged>)
.ERR (1747,STRX06,<No such user number>)
.ERR (1750,MSTX24,<Illegal to dismount the System Structure>)
.ERR (1751,MSTX25,<Invalid number of swapping pages>)
.ERR (1752,MSTX26,<Invalid number of Front-End-Filesystem pages>)
.ERR (1753,LOUTX5,<Illegal to log out job 0>)
.ERR (1754,GJFX43,<More than one ;T specification is not allowed>)
.ERR (1755,MTOX19,<Invalid terminal page width>)
.ERR (1756,MTOX20,<Invalid terminal page length>)
.ERR (1757,MSTX27,<Specified unit is not a disk>)
.ERR (1760,MSTX28,<Could not initialize bit table for structure>)
.ERR (1761,MSTX29,<Could not reconstruct ROOT-DIRECTORY>)
.ERR (1763,DSKX05,<Disk assignments and deassignments are currently prohibited>
.ERR (1764,DSKX06,<Invalid disk address>)
.ERR (1765,DSKX07,<Address cannot be deassigned because it is not assigned>)
.ERR (1766,DSKX08,<Address cannot be assigned because it is already assigned>)
.ERR (1767,COMX10,<Invalid default string>)
.ERR (1770,MSTX30,<Incorrect Bit Table counts on structure>)
.ERR (1771,LOCKX1,<Illegal to lock other than a private page>)
.ERR (1772,LOCKX2,<Requested page unavailable>)
.ERR (1773,LOCKX3,<Attempt to lock too much memory>)
.ERR (1774,ILLX01,<Illegal memory read>)
.ERR (1775,ILLX02,<Illegal memory write>)
.ERR (1776,ILLX03,<Memory data parity error >)
.ERR (1777,ILLX04,<Reference to non-existent page>)
.ERR (2000,MSTX31,<Structure already mounted>)
```

```
.ERR (2001,MSTX32,<Structure was not mounted>)
.ERR (2002,MSTX33,<Structure is unavailable for mounting>)
.ERR (2003,STDIX1,<The STDIR JSYS has been replaced by RCDIR and RCUSR>)
.ERR (2004,CNDIX7,<The CNDIR JSYS has been replaced by ACCES>)
.ERR (2005,PMCLX1,<Illegal page state or state transition>)
.ERR (2006,PMCLX2,<Requested physical page is unavailable>)
.ERR (2007,PMCLX3,<Requested physical page contains errors>)
.ERR (2010,DLFX10,<Cannot delete directory; file still mapped>)
.ERR (2011,DLFX11,<Cannot delete directory file in this manner>)
.ERR (2012,GJFX44,<Account string does not match>)
.ERR (2013,UTSTX1,<Invalid function code>)
.ERR (2014,UTSTX2,<Area of code too large to test>)
.ERR (2015,UTSTX3,<UTEST facility in use by another process>)
.ERR (2016,BOTX01,<Invalid DTE-20 number>)
.ERR (2017,BOTX02,<Invalid byte size>)
.ERR (2020,DCNX1,<Invalid network file name>)
.ERR (2021,DCNX5,<No more logical links available>)
.ERR (2022,DCNX3,<Invalid object>)
.ERR (2023,DCNX4,<Invalid task name>)
.ERR (2024,DCNX9,<Object is already defined>)
.ERR (2025,DCNX8,<Invalid network operation>)
.ERR (2026,DCNX11,<Link aborted>)
.ERR (2027,DCNX12,<String exceeds 16 bytes>)
.ERR (2030,TTYX01,<Line is not active>)
.ERR (2031,BOTX03,<Invalid protocol version number>)
.ERR (2032,MONX05,<Insufficient system resources (no resident free space)>)
.ERR (2033,ARGX19,<Invalid unit number>)
.ERR (2035,COMX11,<Invalid CMRTY pointer>)
.ERR (2036,COMX12,<Invalid CMBFP pointer>)
.ERR (2037,COMX13,<Invalid CMPTR pointer>)
.ERR (2040,COMX14,<Invalid CMABP pointer>)
.ERR (2041,COMX15,<Invalid default string pointer>)
.ERR (2042,COMX16,<Invalid help message pointer>)
.ERR (2043,COMX17,<Invalid byte pointer in function block>)
.ERR (2044,NPXAMB,<Ambiguous>)
.ERR (2045,NPXNSW,<Not a switch - does not begin with slash>)
.ERR (2046,NPXNOM,<Does not match switch or keyword>)
.ERR (2047,NPXNUL,<Null switch or keyword given>)
.ERR (2050,NPXINW,<Invalid guide word>)
.ERR (2051,NPXNC,<Not confirmed>)
.ERR (2052,NPXICN,<Invalid character in number>)
.ERR (2053,NPXIDT,<Invalid device terminator>)
.ERR (2054,NPXNQS,<Not a quoted string - quote missing at beginning or
      end>)
.ERR (2055,NPXNMT,<Does not match token>)
.ERR (2056,NPXNMD,<Does not match directory or user name>)
.ERR (2057,NPXCMA,<Comma not given>)
.ERR (2060,GJFX45,<Illegal to request multiple specifications for the same
      attribute>)
.ERR (2061,GJFX46,<Attribute value is required>)
.ERR (2062,GJFX47,<Attribute does not take a value>)
.ERR (2063,MSTX34,<Unit is write-locked>)
.ERR (2064,GJFX48,<GTJFN input buffer is empty>)
.ERR (2065,GJFX49,<Invalid attribute for this device>)
.ERR (2077,SJBX7,<Remark exceeds 39 characters>)
.ERR (2100,DELF10,<Directory still contains subdirectory>)
.ERR (2101,CRDI13,<Request exceeds superior directory working quota>)
.ERR (2102,CRDI14,<Request exceeds superior directory permanent quota>)
.ERR (2103,CRDI15,<Request exceeds superior directory subdirectory quota>)
.ERR (2104,CRDI16,<Invalid user group>)
.ERR (2105,ENACX1,<Account validation data base file not completely
      closed>)
.ERR (2106,ENACX2,<Cannot get a JFN for <SYSTEM>ACCOUNTS-TABLE.BIN>)
```

```
.ERR (2107,ENACX3,<Account validation data base file too long>)
.ERR (2110,ENACX4,<Cannot get an OFN for <SYSTEM>ACCOUNTS-TABLE.BIN>)
.ERR (2111,VACCX0,<Invalid account>)
.ERR (2112,VACCX1,<Account string exceeds 39 characters>)
.ERR (2113,USGX01,<Invalid USAGE entry type code>)
.ERR (2114,BOTX04,<Byte count is not positive>)
.ERR (2115,NODX01,<Node name exceeds 6 characters>)
.ERR (2116,USGX02,<Item not found in argument list>)
.ERR (2117,CRDI17,<Illegal to create non-files-only subdirectory under
      files-only directory>)
.ERR (2120,ENQX23,<Mismatched mask block lengths>)
.ERR (2121,ENQX22,<Invalid mask block length>)
.ERR (2122,DCNX2,<Interrupt message must be read first>)
.ERR (2123,ABRKX1,<Address break not available on this system>)
.ERR (2124,USGX03,<Default item not allowed>)
.ERR (2125,IPCF35,<Invalid IPCF quota>)
.ERR (2126,VACCX2,<Account has expired>)
.ERR (2127,CRDI18,<Illegal to delete logged-in directory>)
.ERR (2130,CRDI19,<Illegal to delete connected directory>)
.ERR (2132,BOTX05,<Protocol initialization failed>)
.ERR (2133,CRDI20,<WHEEL, OPERATOR, or requested capability required>)
.ERR (2134,COMX18,<Invalid character in node name>)
.ERR (2135,COMX19,<Too many characters in node name>)
.ERR (2136,CRDI21,<Working space insufficient for current allocation>)
.ERR (2137,ACESX7,<Directory is "files-only" and cannot be accessed>)
.ERR (2140,CRDI22,<Subdirectory quota insufficient for existing
      subdirectories>)
.ERR (2141,CRDI23,<Superior directory does not exist>)
.ERR (2142,STRX07,<Invalid user number>)
.ERR (2143,STRX08,<Invalid user name>)
.ERR (2144,CRDI24,<Invalid subdirectory quota>)
.ERR (2146,ATSX01,<Invalid mode>)
.ERR (2147,ATSX02,<Illegal to declare mode twice>)
.ERR (2150,ATSX03,<Illegal to declare mode after acquiring terminal>)
.ERR (2151,ATSX04,<Invalid event code>)
.ERR (2152,ATSX05,<Invalid function code for channel assignment>)
.ERR (2153,ATSX06,<JFN is not an ATS JFN>)
.ERR (2154,ATSX07,<Table length too small>)
.ERR (2155,ATSX08,<Table lengths must be the same>)
.ERR (2156,ATSX09,<Table length too large>)
.ERR (2157,ATSX10,<Maximum applications terminals for system already
      assigned>)
.ERR (2160,ATSX11,<Byte count is too large>)
.ERR (2161,ATSX12,<Terminal not assigned to this JFN>)
.ERR (2162,ATSX13,<Terminal is XOFF'd>)
.err (2163,ATSX14,<Terminal has been released>)
.ERR (2164,ATSX15,<Terminal identifier is not assigned>)
.ERR (2165,PMCLX4,<No more error information>)
.ERR (2166,ATSX16,<Invalid Host Terminal Number>)
.ERR (2167,ATSX17,<Output failed -- monitor internal error>)
.ERR (2170,FRKHX8,<Illegal to manipulate an execute-only process>)
.ERR (2171,ARGX20,<Invalid arithmetic trap argument>)
.ERR (2172,ARGX21,<Invalid LUUO trap argument>)
.ERR (2173,ARGX22,<Invalid flags>)
.ERR (2174,ATSX18,<ATS input message too long for internal buffers>)
.ERR (2175,ATSX19,<Monitor internal error - ATS input message truncated>)
.ERR (2176,ATSX20,<Illegal to close JFN with terminal assigned>)
.ERR (2177,ARGX23,<Invalid section number>)
.ERR (2200,ARGX24,<Invalid count>)
.ERR (2201,MSTX35,<Too many units in structure>)
.ERR (2202,DCNX13,<Node not accessible>)
.ERR (2203,DCNX14,<Previous interrupt message outstanding>)
.ERR (2204,DCNX15,<No interrupt message available>)
```

```
.ERR (2205,GJFX50,<Invalid argument for attribute>)
.ERR (2206,KDPX01,<KMC11 not running>)
.ERR (2207,NODX02,<Line not turned off>)
.ERR (2210,NODX03,<Another line already looped>)
.ERR (2211,GJFX51,<Byte count too small>)
.ERR (2212,COMX20,<Invalid node name>)
.ERR (2213,ATSX21,<Maximum applications terminals for job already
      assigned>)
.ERR (2214,ATSX22,<Failed to acquire applications terminal>)
.ERR (2215,ATSX23,<Invalid device name>)
.ERR (2216,ATSX24,<Invalid server name>)
.ERR (2217,ATSX25,<Terminal is already released>)
.ERR (2220,GOKER1,<Illegal function>)
.ERR (2221,GOKER2,<Request denied by Access Control Facility>)
.ERR (2222,STRX09,<Prior structure mount required>)
.ERR (2223,MSTX36,<Illegal while JFNs assigned>)
.ERR (2224,MSTX37,<Illegal while connected to structure>)
.ERR (2225,MSTX40,<Invalid PSI channel number given>)
.ERR (2226,ATSX26,<Invalid host name>)
.ERR (2227,IOX13,<Invalid segment type>)
.ERR (2230,IOX14,<Invalid segment size>)
.ERR (2231,IOX15,<Illegal tape format for dump mode>)
.ERR (2232,IOX16,<Density specified does not match tape density>)
.ERR (2233,IOX17,<Invalid tape label>)
.ERR (2234,IOX20,<Illegal tape record size>)
.ERR (2235,IOX21,<Tape HDR1 missing>)
.ERR (2236,IOX22,<Invalid tape HDR1 sequence number>)
.ERR (2237,IOX23,<Tape label read error>)
.ERR (2240,IOX24,<Logical end of tape encountered>)
.ERR (2241,IOX25,<Invalid tape format>)
.ERR (2242,SWJFX2,<Illegal to swap ATS JFN>)
.ERR (2243,IOX26,<Tape write date has not expired>)
.ERR (2244,IOX27,<Tape is domestic and HDR2 is missing>)
.ERR (2245,IOX30,<Tape has invalid access character>)
.ERR (2246,ARGX25,<Invalid class>)
.ERR (2247,SKDX1,<Cannot change class>)
.ERR (2250,MREQX1,<Request canceled by user>)
.ERR (2251,MREQX2,<Labeled tapes not permitted on 7-track drives>)
.ERR (2252,MREQX3,<Unknown density specified>)
.ERR (2253,MREQX4,<Unknown drive type specified>)
.ERR (2254,MREQX5,<Unknown label type specified>)
.ERR (2255,MREQX6,<Set name illegal or not specified>)
.ERR (2256,MREQX7,<Illegal starting-volume specification>)
.ERR (2257,MREQX8,<Attempt to switch to volume outside set>)
.ERR (2260,MREQX9,<Illegal volume identifier specified>)
.ERR (2261,MREQ10,<Density mismatch between request and volume>)
.ERR (2262,MREQ11,<Drive type mismatch between request and volume>)
.ERR (2263,MREQ12,<Label type mismatch between request and volume>)
.ERR (2264,MREQ13,<Structural error in mount message>)
.ERR (2265,MREQ14,<Setname mismatch between request and volume>)
.ERR (2266,MREQ15,<Mount refused by operator>)
.ERR (2267,MREQ16,<Volume identifiers not supplied by operator>)
.ERR (2270,MREQ17,<Volume-identifier list missing>)
.ERR (2271,MREQ18,<End of volume-identifier list reached while reading>)
.ERR (2272,MREQ19,<Requested tape drive type not available to system>)
.ERR (2273,MREQ20,<Structural error in mount entry>)
.ERR (2274,MREQ21,<Mount requested for unknown device type>)
.ERR (2275,DEVX6,<Job has open JFN on device>)
.ERR (2276,ATSX27,<Terminal is not open>)
.ERR (2277,ATSX28,<Unknown error received>)
.ERR (2300,ATSX29,<Receive error threshold exceeded>)
.ERR (2301,ATSX30,<Reply threshold exceeded>)
.ERR (2302,ATSX31,<NAK threshold exceeded>)
```

```
.ERR (2303,ATSX32,<Terminal protocol error>)
.ERR (2304,ATSX33,<Intervention required at terminal>)
.ERR (2305,ATSX34,<Powerfail>)
.ERR (2306,ATSX35,<Data pipe was disconnected>)
.ERR (2307,ATSX36,<Dialup terminal was attached>)
.ERR (2310,DATEX7,<Julian day is out of range>)
.ERR (2311,MREQ22,<Structure name not specified>)
.ERR (2312,ARCFX2,<File already has archive status>)
.ERR (2313,ARCFX3,<Cannot perform ARCF functions on non-multiple directory
      devices>)
.ERR (2314,ARCFX4,<File is not on-line>)
.ERR (2315,ARCFX5,<Files not on the same device or structure>)
.ERR (2316,ARCFX6,<File does not have archive status>)
.ERR (2317,ARCFX7,<Invalid parameter>)
.ERR (2320,ARCFX8,<Archive not complete>)
.ERR (2321,ARCFX9,<File not off-line>)
.ERR (2322,ARCX10,<Archive prohibited>)
.ERR (2323,ARCX11,<Archive requested, modification prohibited>)
.ERR (2324,ARCX12,<Archive requested, delete prohibited>)
.ERR (2325,ARCX13,<Archive system request not completed>)
.ERR (2326,OPNX30,<File has archive status, modification is prohibited>)
.ERR (2327,OPNX31,<File is off-line>)
.ERR (2330,DELX11,<File has archive status, delete is not permitted>)
.ERR (2331,DELX12,<File has no pointer to offline storage>)
.ERR (2332,ARCX14,<File restore failed>)
.ERR (2333,ARCX15,<Migration prohibited>)
.ERR (2334,ARCX16,<Cannot exempt offline file>)
.ERR (2335,ARCX17,<FDB incorrect format for ARCF JSYS>)
.ERR (2336,ARCX18,<Retrieval request cannot be fulfilled for waiting
      process>)
.ERR (2337,ARCX19,<Migration already pending>)
.ERR (2340,ARGX26,<File is offline>)
.ERR (2341,ARGX27,<Offline expiration time cannot exceed system maximum>)
.ERR (2342,DIRX5,<Directory too large>)
.ERR (2343,IOX31,<Invalid record descriptor in labeled tape>)
.ERR (2344,MREQ23,<Dismount refused by operator>)
.ERR (2345,MREQ24,<Illegal to dismount connected structure>)
.ERR (2346,MREQ25,<Structure not found>)
.ERR (2347,LTLBLX,<Too many user labels>)
.ERR (2350,LTLBX1,<Undefined record format on non-TOPS20 tape>)
.ERR (2351,MREQ26,<Tape mounting function disabled by installation>)
.ERR (2352,METRX1,<METER% not supported on this processor>)
.ERR (2353,NSPX00,<Connection not accepted>)
.ERR (2354,NSPX01,<Resource allocation failure>)
.ERR (2355,NSPX02,<Destination node does not exist>)
.ERR (2356,NSPX03,<Node shutting down>)
.ERR (2357,NSPX04,<Destination process does not exist>)
.ERR (2360,NSPX05,<Invalid process name>)
.ERR (2361,NSPX06,<Destination process queue overflow>)
.ERR (2362,NSPX07,<Unspecified error>)
.ERR (2363,NSPX08,<Connection aborted by third party>)
.ERR (2364,NSPX09,<Link aborted by process>)
.ERR (2365,NSPX10,<NSP Failure - Flow control violation>)
.ERR (2366,NSPX11,<Too many connections to node>)
.ERR (2367,NSPX12,<Too many connections to destination process>)
.ERR (2370,NSPX13,<Access denied due to unacceptable user name or
      password>)
.ERR (2371,NSPX14,<NSP failure - invalid SERVICES field>)
.ERR (2372,NSPX15,<Invalid account>)
.ERR (2373,NSPX16,<NSP failure - invalid SEGSIZ field>)
.ERR (2374,NSPX17,<Process aborted, timed out, or cancelled request>)
.ERR (2375,NSPX18,<No path to destination node>)
.ERR (2376,NSPX19,<NSP failure - flow control failure>)
```

```
.ERR (2377,NSPX20,<NSP failure - invalid DSTADDR>)
.ERR (2400,NSPX21,<Disconnect confirmation>)
.ERR (2401,NSPX22,<NSP failure - image data field too long>)
.ERR (2402,MREQ27,<Structure is set IGNORED>)
.ERR (2403,MREQ28,<Cannot overwrite volume - first file is not expired>)
.ERR (2404,MREQ29,<Cannot overwrite volume - write access required>)
.ERR (2405,MREQ30,<Tape label format error>)
.ERR (2406,DIAG11,<Unit already online>)
.ERR (2407,DIAG12,<Unit not online>)
.ERR (2410,DESX11,<Invalid operation for this label type>)
.ERR (2411,NSPX23,<Invalid NSP reason code>)
.ERR (2412,ARGX28,<not available on this system>)
.ERR (2413,NPX2CL,<Two colons required on node name>)
.ERR (2414,ARGX29,<Invalid class share>)
.ERR (2415,ARGX30,<Invalid KNOB value>)
.ERR (2416,ARGX31,<Class Scheduler already enabled>)
.ERR (2417,DEVX7,<Null device name given>)
.ERR (2420,GJFX52,<End of tape encountered while searching for file>)
.ERR (2421,GOKER3,<JSYS not executed within ACJ fork>)
.ERR (2422,IOX32,<Tape position is indeterminate>)
.ERR (2423,IOX33,<TTY input buffer full>)
.ERR (2424,XSIRX1,<Channel table crosses section boundary>)
.ERR (2425,SIRX2,<SIR JSYS invoked from non-zero section>)
.ERR (2426,RIRX1,<RIR JSYS incompatible with previous XSIR>)
.ERR (2427,XSIRX2,<Level table crosses section boundary>)
.ERR (2430,MREQ31,<Insufficient MOUNTR resources>)
.ERR (2431,SMAPX1,<Attempt to delete a section still shared>)
.ERR (2432,TTMSX1,<Could not send message within timeout interval>)
.ERR (2433,MONX06,<Insufficient system resources (No swappable free
      space)>)
.ERR (2434,BOTX06,<GTJFN failed for dump file>)
.ERR (2435,BOTX07,<OPENF failed for dump file>)
.ERR (2436,BOTX08,<Dump failed>)
.ERR (2437,BOTX09,<To -10 error on dump>)
.ERR (2440,BOTX10,<To -11 error on dump>)
.ERR (2441,BOTX11,<Failed to assign page on dump>)
.ERR (2442,BOTX12,<Reload failed>)
.ERR (2443,BOTX13,<-11 didn't power down>)
.ERR (2444,BOTX14,<-11 didn't power up>)
.ERR (2445,BOTX15,<ROM did not ACK the -10>)
.ERR (2446,BOTX16,<-11 boot program did not make it to -11>)
.ERR (2447,BOTX17,<-11 took more than 1 minute to reload. Will cause retry>)
.ERR (2450,BOTX18,<Unknown BOOT error>)
.ERR (2451,NTMX1,<Network Management unable to complete request>)
.ERR (2452,COMX21,<Node name doesn't contain an alphabetic character>)
.ERR (2453,DELX13,<File is marked "Never Delete">)
.ERR (2454,ANTX01,<No more network terminals available>)
.ERR (2455,TTYX02,<Illegal character specified>)
.ERR (2456,NSPX24,<Node name not assigned to a network node>)
.ERR (2457,NSPX25,<Illegal DECnet node number>)
.ERR (2460,NSPX26,<Table of topology watchers is full>)
.ERR (2461,GJFX53,<Tape label filename specification exceeds 17
      characters>)
.ERR (2462,IOX34,<Disk structure completely full>)
.ERR (2463,IOX35,<Disk structure damaged, cannot allocate space>)
.ERR (2464,PMAPX8,<Indirect page map loop detected>)
.ERR (2465,SMAPX2,<Indirect section map loop detected>)
.ERR (2466,GJFX54,<Node name not first field>)
.ERR (2467,BOTX19,<Overdue TO-11 transfer aborted>)
.ERR (2470,BOTX20,<Overdue TO-10 transfer aborted>)
.ERR (2471,ILLX05,<Illegal memory reference, section greater than 37>)
.ERR (2472,XSEVX1,<Illegal entry vector type>)
.ERR (2473,XSEVX2,<Invalid entry vector length>)
```

```
.ERR (2474,XSEVX3,<Cannot get extended values with this monitor call>)
>                                ;END OF .ERCOD DEFINITION


;DEFINE THE ERROR CODE VALUES

DEFINE .ERR (N,E,S) <
        E==:.ERBAS+N
        IFG <N-.ERMAX>,<.ERMAX==:N>>

        .ERMAX==:0

.ERCOD
```

```
;THIS SECTION CONSISTS OF SPECIAL CODE TO WRITE THE ERRMES.BIN FILE
; THE CODE IS ONLY ASSEMBLED IF .ERBLD IS PREVIOUSLY
; DEFINED TO BE NON-ZERO.

IFNDEF .ERBLD,<.ERBLD==0>

IFN .ERBLD,<

.ERGO:  MOVSI 1,(GJ%FOU!GJ%SHT) ;GET A JFN ON ERROR FILE
        HRROI 2,[ASCIZ/ERRMES.BIN/]
        GTJFN
         JRST .ERER
        MOVE 2,[440000,,OF%WR]
        OPENF                    ;OPEN THE FILE FOR WRITE
         JRST .ERER
        MOVNI 3,.ERSTE-.ERTAB    ;GET LENGTH OF FILE
        MOVE 2,[POINT 36,.ERTAB]
        SOUT                     ;OUTPUT THE ERROR FILE DATA
        CLOSF                    ;CLOSE THE FILE
         JRST .ERER
        HALTF                    ;DONE

.ERER:  MOVEI 1,101              ;TYPE OUT ERROR CODE
        HRLOI 2,400000
        SETZ 3,
        ERSTR
         JFCL
         JFCL
        HALTF

LIT

DEFINE .ERR (N,E,S) <
        .ERQQ==<.-.ERTAB>*5
        .ERQQ2==N&37777
        .ERRM1 \.ERQQ2,N,.ERQQ
        ASCII \S'@\
>

DEFINE .ERRM1 (NN,N,.ERQQ)<
   IF1,<IFDEF EZ'NN,<
        PRINTX ERROR N=NN HAS ALREADY BEEN USED
   >>
   EZ'NN==1
   RELOC .ERTAB+NN
        .ERQQ
   RELOC
>

.ERTAB: .ERMAX                   ;FIRST WORD OF TABLE IS THE LENGTH
                                 ; OF THE TABLE FOR ERSTR TO USE AS
                                 ; A BOUNDS CHECK.
        BLOCK .ERMAX             ;LEAVE ROOM FOR POINTERS

.ERST:  .ERCOD                   ;BUILD STRINGS AND .ERTAB
.ERSTE:                          ;END OF STRINGS

        END .ERGO

>                                ;END OF IFN .ERBLD CONDITIONAL

PURGE .ERR,REL

END
;D$
```

APPENDIX C

MACSYM

```
; UPD ID= 82, <5.UTILITIES>MACSYM.MAC.43,  22-Feb-82 17:57:38 by MURPHY


;THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED
;  OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.
;
;COPYRIGHT (C) 1976,1977,1978,1979,1980,1981 BY DIGITAL EQUIPMENT CORPORATION,
 MAYNARD, MASS.

;VERSION 1

IFNDEF REL,<REL==0>             ;UNIVERSAL UNLESS OTHERWISE DECLARED
   IFE REL,<
        UNIVERSAL MACSYM        COMMON MACROS AND SYMBOLS
   >
   IFN REL,<
        TITLE MACREL            SUPPORT CODE FOR MACSYM
        SEARCH MONSYM
        SALL
   >

;THE STANDARD VERSION WORD CONSTRUCTION
; VERS - PROGRAM VERSION NUMBER
; VUPDAT - PROGRAM UPDATE NUMBER (1=A, 2=B ...)
; VEDIT - PROGRAM EDIT NUMBER
; VCUST - CUSTOMER EDIT CODE (0=DEC DEVELOPMENT, 1=DEC SWS, 2-7 CUST)

DEFINE PGVER.   (VERS,VUPDAT,VEDIT,VCUST)<
        ..PGV0==.               ;;SAVE CURRECT LOCATION AND MODE
        .JBVER=:^O137           ;;WHERE TO PUT VERSION
        LOC     .JBVER          ;;PUT VERSION IN STANDARD PLACE
        BYTE    (3)VCUST(9)VERS(6)VUPDAT(18)VEDIT
        .ORG    ..PGV0          ;;RESTORE LOCATION AND MODE
>

;MASKS FOR THE ABOVE

VI%WHO==:7B2                    ;Customer edit code
VI%MAJ==:777B11                 ;Major version number
VI%MIN==:77B17                  ;Minor version/update
VI%EDN==:777777B35              ;Edit number
;ADDED VI%XXX
```

```
        SUBTTL COMMON DEFS

;DEFINE STANDARD AC'S

DEFINE STDAC. <
F=:0
T1=:1
T2=:2
T3=:3
T4=:4
Q1=:5
Q2=:6
Q3=:7
P1=:10
P2=:11
P3=:12
P4=:13
P5=:14
P6=:15
CX=:16
P=:17
>
        SUBTTL MISC CONSTANTS

;MISC CONSTANTS

.INFIN==:377777,,777777         ;PLUS INFINITY
.MINFI==:1B0                    ;MINUS INFINITY
.LHALF==:777777B17              ;LEFT HALF
.RHALF==:777777                 ;RIGHT HALF
.FWORD==:-1                     ;FULL WORD
```

```
        SUBTTL  SYMBOLS FOR THE CONTROL CHARACTERS

.CHNUL==:000                    ;NULL
.CHCNA==:001
.CHCNB==:002
.CHCNC==:003
.CHCND==:004
.CHCNE==:005
.CHCNF==:006
.CHBEL==:007                    ;BELL
.CHBSP==:010                    ;BACKSPACE
.CHTAB==:011                    ;TAB
.CHLFD==:012                    ;LINE-FEED
.CHVTB==:013                    ;VERTICAL TAB
.CHFFD==:014                    ;FORM FEED
.CHCRT==:015                    ;CARRIAGE RETURN
.CHCNN==:016
.CHCNO==:017
.CHCNP==:020
.CHCNQ==:021
.CHCNR==:022
.CHCNS==:023
.CHCNT==:024
.CHCNU==:025
.CHCNV==:026
.CHCNW==:027
.CHCNX==:030
.CHCNY==:031
.CHCNZ==:032
.CHESC==:033                    ;ESCAPE
.CHCBS==:034                    ;CONTROL BACK SLASH
.CHCRB==:035                    ;CONTROL RIGHT BRACKET
.CHCCF==:036                    ;CONTROL CIRCUMFLEX
.CHCUN==:037                    ;CONTROL UNDERLINE
.CHSPC==:040                    ;SPACE

.CHALT==:175                    ;OLD ALTMODE
.CHAL2==:176                    ;ALTERNATE OLD ALTMODE
.CHDEL==:177                    ;DELETE
```

```
        SUBTTL  HARDWARE BITS OF INTEREST TO USERS

;PC FLAGS

PC%OVF==:1B0                    ;OVERFLOW
PC%CY0==:1B1                    ;CARRY 0
PC%CY1==:1B2                    ;CARRY 1
PC%FOV==:1B3                    ;FLOATING OVERFLOW
PC%BIS==:1B4                    ;BYTE INCREMENT SUPPRESSION
PC%USR==:1B5                    ;USER MODE
PC%UIO==:1B6                    ;USER IOT MODE
PC%LIP==:1B7                    ;LAST INSTRUCTION PUBLIC
PC%AFI==:1B8                    ;ADDRESS FAILURE INHIBIT
PC%ATN==:3B10                   ;APR TRAP NUMBER
PC%FUF==:1B11                   ;FLOATING UNDERFLOW
PC%NDV==:1B12                   ;NO DIVIDE
```

```
;THE FOLLOWING MACRO MAY BE USED TO SUPPRESS CREF ENTRIES FOR
;ALL THE JUNK SYMBOLS USED INTERNALLY WITHIN MACROS IN MACSYM

DEFINE .XCMSY <
        .XCREF
        .XCRF1 <..ACT,..CSC,..CSN,..IFT,..JX1,..MSK,..MX1,..MX2>
        .XCRF1 <..NAC,..NRGS,..NS,..NV,..PST,..STKN,..STKQ,..STKR>
        .XCRF1 <..TRR,..TSA1,..TX1,..TX2,.FP,.FPAC,.NAC,.SAC,.SAV1>
        .XCRF1 <.SAV2,.SAV3,POINTR,POS,WID,..CAS1,..CNS,..CNS2>
        .XCRF1 <..DPB,..GNCS,..ICNS,..JE,..LDB,..STR0,..STR1,..STR2>
        .XCRF1 <..STR4,..TQO,..TQZ,..TSAC,..TSIZ,..TX,..TY,.ACV1,.ACV2>
        .XCRF1 <.ACV3,.CASE,.DECR0,.IF0,.INCR0,.OPST1,.OPST2,.STKV1>
        .XCRF1 <.STKV2,.STKV3,.TRV1,.TRV2,.TRV3>
        .CREF
   >
DEFINE .XCRF1 (SYMS)<
        IRP SYMS,<
         IFDEF SYMS,< .XCREF SYMS>>>

        SUBTTL  MACROS FOR FIELD MASKS

;STANDARD MACROS

;MACROS TO HANDLE FIELD MASKS

;COMPUTE LENGTH OF MASK, I.E. LENGTH OF LEFTMOST STRING OF ONES
;REMEMBER THAT ^L DOES 'JFFO', I.E. HAS VALUE OF FIRST ONE BIT IN WORD

;COMPUTE WIDTH OF MASK, I.E. LENGTH OF LEFTMOST STRING OF ONES

DEFINE WID(MASK)<<^L<-<<MASK>_<^L<MASK>>>-1>>>

;COMPUTE POSITION OF MASK, I.E. BIT POSITION OF RIGHTMOST ONE IN MASK

DEFINE POS(MASK)<<^L<<MASK>&<-<MASK>>>>>

;CONSTRUCT BYTE POINTER TO MASK

DEFINE POINTR(LOC,MASK)<<POINT WID(MASK),LOC,POS(MASK)>>

;PUT RIGHT-JUSTIFIED VALUE INTO FIELD SPECIFIED BY MASK

DEFINE FLD(VAL,MSK)<<<<VAL>B<POS(MSK)>>&<MSK>>>

;MAKE VALUE BE RIGHT JUSTIFIED IN WORD.

DEFINE .RTJST(VAL,MSK)<<<<VAL>&<MSK>>B<^D70-POS(MSK)>>>

;CONSTRUCT MASK FROM BIT AA TO BIT BB. I.E. MASKB 0,8 = 777B8

DEFINE MASKB(AA,BB)<<1B<<AA>-1>-1B<BB>>>

;MODULE - GIVES REMAINDER OF DEND DIVIDED BY DSOR

DEFINE MOD.(DEND,DSOR)<<<DEND>-<<DEND>/<DSOR>>*<DSOR>>>
```

```
;REPEAT WITH SUBSTITUTION OF NUMERIC INDEX

DEFINE FORN. (LOW,HIGH,ARGS,STRING,%MN1)<
  DEFINE %MN1(ARGS)<STRING>
..FORN==LOW
REPEAT HIGH-LOW+1,<
        .FORN1 (%MN1)
        ..FORN=..FORN+1>>

DEFINE .FORN1 (MACN)<
        MACN (\..FORN)>

;REPEAT WITH GENERAL STRING SUBSTITUTION

DEFINE FORX. (ARGS,SYMS,STRING,%MN1)<
  DEFINE %MN1 (SYMS)<STRING>
  IRP ARGS,<
        .FORX1 %MN1,ARGS>>

  DEFINE .FORX1 (MACN,ARGS)<
        MACN ARGS>
```

```
        SUBTTL MOVX

;MOVX - LOAD AC WITH CONSTANT

DEFINE MOVX (AC,MSK)<
   ..MX1==MSK                   ;;EVAL EXPRESSION IF ANY
   .IFN ..MX1,ABSOLUTE,<
        MOVE AC,[MSK]>
   .IF ..MX1,ABSOLUTE,<
        ..MX2==0                ;;FLAG SAYS HAVEN'T DONE IT YET
        IFE <..MX1>B53,<
          ..MX2==1
          MOVEI AC,..MX1>       ;;LH 0, DO AS RH
        IFE ..MX2,<             ;;IF HAVEN'T DONE IT YET,
        IFE <..MX1>B17,<
          ..MX2==1
          MOVSI AC,(..MX1)>>    ;;RH 0, DO AS LH
        IFE ..MX2,<             ;;IF HAVEN'T DONE IT YET,
        IFE <<..MX1>B53-^O777777>,<
          ..MX2==1
          HRROI AC,<..MX1>>>    ;;LH -1
        IFE ..MX2,<             ;;IF HAVEN'T DONE IT YET,
        IFE <<..MX1>B17-^O777777B17>,<
          ..MX2==1
          HRLOI AC,(..MX1-^O777777)>> ;;RH -1
        IFE ..MX2,<             ;;IF STILL HAVEN'T DONE IT,
          MOVE AC,[..MX1]>      ;;GIVE UP AND USE LITERAL
  >>

;MV., MVI. - Move from memory to memory or immediate to memory

DEFINE MV. (FROM,TOO)<
        MOVE .SAC,FROM
        MOVEM .SAC,TOO>

DEFINE MVI. (STUFF,DEST)<
        MOVX .SAC,<STUFF>
        MOVEM .SAC,DEST>
```

```
;VARIENT MNEMONICS FOR TX DEFINITIONS

DEFINE IORX (AC,MSK)<
        TXO AC,<MSK>>

DEFINE ANDX (AC,MSK)<
        TXZ AC,<^-<MSK>>>

DEFINE XORX (AC,MSK)<
        TXC AC,<MSK>>
```

```
        SUBTTL TX -- TEST MASK

;CREATE THE TX MACRO DEFINITIONS

;THIS DOUBLE IRP CAUSES ALL COMBINATIONS OF MODIFICATION AND TESTING
;TO BE DEFINED

DEFINE ..DOTX (M,T)<
        IRP M,<
        IRP T,<
          DEFINE TX'M'T (AC,MSK)<
                ..TX(M'T,AC,<MSK>)>>>>

        ..DOTX (<N,O,Z,C>,<,E,N,A>) ;DO ALL DEFINITIONS
        PURGE ..DOTX

;..TX
;ALL TX MACROS JUST CALL ..TX WHICH DOES ALL THE WORK

DEFINE ..TX(MT,AC,MSK)<
   ..TX1==MSK                   ;;EVAL EXPRESSION IF ANY
   .IFN ..TX1,ABSOLUTE,<
        TD'MT AC,[MSK]>
   .IF ..TX1,ABSOLUTE,<         ;;MASK MUST BE TESTABLE
        ..TX2==0                ;;FLAG SAYS HAVEN'T DONE IT YET
        IFE <..TX1&^O777777B17>,<
          ..TX2==1              ;;LH 0, DO AS RH
          TR'MT AC,..TX1>
        IFE ..TX2,<             ;;IF HAVEN'T DONE IT YET,
        IFE <..TX1&^O777777>,<
          ..TX2==1              ;;RH 0, DO AS LH
          TL'MT AC,(..TX1)>>
        IFE ..TX2,<             ;;IF HAVEN'T DONE IT YET,
          IFE <<..TX1>B53-^O777777>,< ;;IF LH ALL ONES,
            ..TX3 (MT,AC)>>     ;;TRY Z,O,C SPECIAL CASES
        IFE ..TX2,<             ;;IF HAVEN'T DONE IT YET,
          IFE <..TX1+1>,<       ;;TRY WORD ALL ONES
            ..TX4 (MT,AC)>>
        IFE ..TX2,<             ;;IF STILL HAVEN'T DONE IT,
          TD'MT AC,[..TX1]>     ;;MUST GIVE UP AND USE LITERAL
  >>
```

```
;SPECIAL CASE FOR LH ALL ONES

DEFINE ..TX3 (MT,AC)<
        IFIDN <MT><Z>,<             ;;IF ZEROING WANTED
          ..TX2==1
          ANDI AC,^-..TX1>          ;;CAN DO IT WITH ANDI
        IFIDN <MT><O>,<             ;;IF SET TO ONES WANTED
          ..TX2==1
          ORCMI AC,^-..TX1>         ;;CAN DO IT WITH IORCM
        IFIDN <MT><C>,<             ;;IF COMPLEMENT WANTED
          ..TX2==1
          EQVI AC,^-..TX1>>         ;;CAN DO IT WITH EQV

;SPECIAL CASE OF WORD ALL ONES

DEFINE ..TX4 (MT,AC)<
        IFIDN <MT><NN>,<
          ..TX2==1
          CAIN AC,0>                ;;CAN DO FULL WORD COMPARE
        IFIDN <MT><NE>,<
          ..TX2==1
          CAIE AC,0>>
```

```
        SUBTTL JX -- JUMP ON MASK

;JXE -- JUMP IF MASKED BITS ARE EQUAL TO 0
;JXN -- JUMP IF MASKED BITS ARE NOT EQUAL TO 0
;JXO -- JUMP IF MASKED BITS ARE ALL ONES
;JXF -- JUMP IF MASKED BITS ARE NOT ALL ONES (FALSE)

DEFINE JXE (AC,MSK,BA)<
        ..JX1==MSK                      ;;EVAL EXPRESSION IF ANY
        .IFN ..JX1,ABSOLUTE,<PRINTX MSK NOT ABSOLUTE
                ..JX1==0>
        .IF ..JX1,ABSOLUTE,<
        .IF0 <<..JX1>-1B0>,<            ;;IF MASK IS JUST B0,
                JUMPGE AC,BA>,<
        .IF0 <<..JX1>+1>,<              ;;IF MASK IF FULL WORD,
          JUMPE AC,BA>,<                ;;USE GIVEN CONDITION
                TXNN (AC,..JX1)
                JRST BA>>>>

DEFINE JXN (AC,MSK,BA)<
        ..JX1==MSK                      ;;EVAL EXPRESSION IF ANY
        .IFN ..JX1,ABSOLUTE,<PRINTX MSK NOT ABSOLUTE
                ..JX1==0>
        .IF ..JX1,ABSOLUTE,<
        .IF0 <<..JX1>-1B0>,<            ;;IF MASK IS JUST B0,
                JUMPL AC,BA>,<
        .IF0 <<..JX1>+1>,<              ;;IF MASK IF FULL WORD,
          JUMPN AC,BA>,<                ;;USE GIVEN CONDITION
                TXNE (AC,..JX1)
                JRST BA>>>>
```

```
DEFINE JXO (AC,MSK,BA)<
        ..JX1==MSK              ;;EVAL EXPRESSION
        .IFN ..JX1,ABSOLUTE,<PRINTX MSK NOT ABSOLUTE
                ..JX1==0>
        .IF ..JX1,ABSOLUTE,<
        .IF0 <<..JX1>-1B0>,<
                JUMPL AC,BA>,<
        ..ONEB (..BT,MSK)       ;;TEST MASK FOR ONLY ONE BIT ON
        .IF0 ..BT,<
          SETCM .SAC,AC         ;;GENERAL CASE, GET COMPLEMENTS OF BITS
          JXE (.SAC,..JX1,BA)>,< ;;JUMP IF BITS WERE ORIGINALLY ONES
            TXNE AC,..JX1       ;;TEST AND JUMP
            JRST BA>>>>

DEFINE JXF (AC,MSK,BA)<
        ..JX1==MSK              ;;EVAL EXPRESSION
        .IFN ..JX1,ABSOLUTE,<PRINTX MSK NOT ABSOLUTE
                ..JX1==0>
        .IF ..JX1,ABSOLUTE,<
        .IF0 <<..JX1>-1B0>,<
                JUMPGE AC,BA>,<
        ..ONEB (..BT,MSK)       ;;TEST MASK FOR ONLY ONE BIT ON
        .IF0 ..BT,<
          SETCM .SAC,AC         ;;GENERAL CASE, GET COMPLEMENT OF BITS
          JXN (.SAC,..JX1,BA)>,< ;;JUMP IF SOME ZEROS ORIGINALLY
            TXNN AC,..JX1       ;;TEST AND JUMP
            JRST BA>>>>
```

```
        SUBTTL SUBFUNCTION MACROS

;.IF0 CONDITION, ACTION IF CONDITION 0, ACTION OTHERWISE

DEFINE .IF0 (COND,THEN,ELSE)<
        ..IFT==COND                 ;;GET LOCAL VALUE FOR CONDITION
        IFE ..IFT,<
        THEN
        ..IFT==0>                   ;;RESTORE IN CASE CHANGED BY NESTED .IF0
        IFN ..IFT,<
        ELSE>>

;CASE (NUMBER,<FIRST,SECOND,...,NTH>)

DEFINE .CASE (NUM,LIST)<
        ..CSN==NUM
        ..CSC==0
        IRP LIST,<
        IFE ..CSN-..CSC,<
          STOPI
          ..CAS1 (LIST)>
        ..CSC==..CSC+1>>

DEFINE ..CAS1 (LIST)<
        LIST>

;TEST FOR FULL WORD, RH, LH, OR ARBITRARY BYTE

DEFINE ..TSIZ (SYM,MSK)<
        SYM==3                      ;;ASSUME BYTE UNLESS...
        IFE <MSK>+1,<SYM=0>         ;;FULL WORD IF MASK IS -1
        IFE <MSK>-^O777777,<SYM==1> ;;RH IF MASK IS 777777
        IFE <MSK>-^O777777B17,<SYM==2>> ;;LH IF MAST IS 777777,,0

;TEST FOR LOC BEING AN AC -- SET SYM TO 1 IF AC, 0 IF NOT AC

DEFINE ..TSAC (SYM,LOC)<
        SYM==0                      ;;ASSUME NOT AC UNLESS...
        ..TSA1==<Z LOC>             ;;LOOK AT LOC
        .IF ..TSA1,ABSOLUTE,<       ;;SEE IF WE CAN TEST VALUE
          IFE ..TSA1&^O777777777760,<SYM==1>> ;;AC IF VALUE IS 0-17
        >

;TEST FOR SPECIFIC NTH CHARACTER OF ARG

DEFINE ..TSNC (SYM,NTH,STR,CH)<
        SYM==0                      ;;ASSUME NO
        ..TSA1==0                   ;;COUNT CHARS
        IRPC STR,<
          ..TSA1=..TSA1+1
          IFE ..TSA1-NTH,<
           IFIDN <STR><CH>,<
                SYM==1>             ;;YES
           STOPI>>>

;FUNCTION TO TEST FOR MASK CONTAINING EXACTLY ONE BIT. RETURNS
;1 IFF LEFTMOST BIT AND RIGHTMOST BIT ARE SAME

DEFINE ..ONEB (SYM,MSK)<
        SYM==<<<-<MSK>>&<MSK>>&<1B<^L<MSK>>>>>

;DEFAULT SCRACH AC

.SAC=16
```

```
        SUBTTL DEFSTR -- DEFINE DATA STRUCTURE

;DEFINE DATA STRUCTURE
; NAM - NAME OF STRUCTURE AS USED IN CODE
; LOCN - ADDRESS OF DATA
; POS - POSITION OF DATA WITHIN WORD (RIGHTMOST BIT NUMBER)
; SIZ - SIZE OF DATA (IN BITS) WITHIN WORD

DEFINE DEFSTR (NAM,LOCN,POS,SIZ)<
        NAM==<-1B<POS>+1B<POS-SIZ>> ;;ASSIGN SYMBOL TO HOLD MASK
        IF1,<IFDEF %'NAM,<PRINTX ?NAM ALREADY DEFINED>>
        DEFINE %'NAM (OP,AC,Y,MSK)<
        $'NAM==<Z LOCN>             ;;LOCATION SYMBOL FOR DDT
        OP (<AC>,LOCN''Y,MSK)>> ;;DEFINE MACRO TO HOLD LOCATION

;ALTERNATE FORM OF DEFSTR -- TAKES MASK INSTEAD OF POS,SIZ

DEFINE MSKSTR (NAM,LOCN,MASK)<
        NAM==MASK                   ;;ASSIGN SYMBOL TO HOLD MASK
        IF1,<IFDEF %'NAM,<PRINTX ?NAM ALREADY DEFINED>>
        DEFINE %'NAM (OP,AC,Y,MSK)<
        $'NAM==<Z LOCN>             ;;LOCATION SYMBOL FOR DDT
        OP (<AC>,LOCN''Y,MSK)>> ;;DEFINE MACRO TO HOLD LOCATION

;..STR0 - PROCESS INSTANCE OF STRUCTURE USAGE, SINGLE STRUCTURE CASE.

DEFINE ..STR0 (OP,AC,STR,Y)<
        IFNDEF STR,<PRINTX ?STR IS NOT DEFINED
          OP (<AC>,Y,.FWORD)>       ;;RESERVE A WORD, ASSUME WORD MASK
        IFDEF STR,<
        IFNDEF %'STR,<
          OP (<AC>,Y,STR)>          ;;ASSUME NO OTHER LOCN
        IFDEF %'STR,<
          %'STR (OP,<AC>,Y,STR)>>> ;;DO IT
```

```
;..STR1, ..STR2, ..STR3, AND ..STR4 ARE INTERNAL MACROS FOR PROCESSING
;INSTANCES OF STRUCTURE USAGE.

DEFINE ..STR1 (OP,AC,STR,Y,CLL)<
        ..NS==0                 ;;INIT COUNT OF STR'S
        IRP STR,<..NS=..NS+1>   ;;COUNT STR'S
        IFE ..NS,<PRINTX ?EMPTY STRUCTURE LIST, OP>
        IFE ..NS-1,<            ;;THE ONE CASE, CAN DO FAST
          ..STR0 (OP,<AC>,<STR>,Y)>
        IFG ..NS-1,<            ;;MORE THAN ONE, DO GENERAL CASE
        ..ICNS                  ;;INIT REMOTE MACRO
        ..CNS (<CLL (OP,<AC>,,>) ;;CONS ON CALL AND FIRST ARGS
        IRP STR,<               ;;DO ALL NAMES IN LIST
          IFNDEF STR,<PRINTX STR NOT DEFINED>
          IFDEF STR,<
          IFNDEF %'STR,<
          ..CNS (<,STR,Y>)>     ;;ASSUME NO OTHER LOCN
          IFDEF %'STR,<
          %'STR (..STR2,,Y,STR)> ;;STR MACRO WILL GIVE LOCN TO ..STR2
          ..CNS (<)>)           ;;CLOSE ARG LIST
          ..GCNS                ;;DO THIS AND PREVIOUS NAME
          ..ICNS                ;;REINIT CONS
          ..CNS (<CLL (OP,<AC>>) ;;PUT ON FIRST ARGS
          IFNDEF %'STR,<
          ..CNS (<,STR,Y>)>     ;;ASSUME NO OTHER LOCN
          IFDEF %'STR,<
          %'STR (..STR2,,Y,STR)>>> ;;PUT ON THIS ARG, END IRP
        ..CNS (<,,)>)           ;;CLOSE ARG LIST
        ..GCNS>>                ;;DO LAST CALL
```

```
;..STR2 -- CALLED BY ABOVE TO APPEND STRUCTURE NAME AND LOC TO ARG LIST

DEFINE ..STR2 (AA,LOC,STR)<
        ..CNS (<,STR,LOC>)>     ;;CONS ON NEXT ARG PAIR

;..STR3 -- CHECK FOR ALL STRUCTURES IN SAME REGISTER

DEFINE ..STR3 (OP,AC,S1,L1,S2,L2)<
        IFDIF <L1><L2>,<
          IFNB <L1>,<
            OP (<AC>,L1,..MSK)  ;;DO ACCUMULATED STUFF
            IFNB <L2>,<PRINTX S1 AND S2 ARE IN DIFFERENT WORDS>>
          ..MSK==0>             ;;INIT MASK
        IFNB <L2>,<
          ..MSK=..MSK!<S2>>>

;..STR4 -- COMPARE SUCCESSIVE ITEMS, DO SEPARATE OPERATION IF
;DIFFERENT WORDS ENCOUNTERED

DEFINE ..STR4 (OP,AC,S1,L1,S2,L2)<
        IFDIF <L1><L2>,<        ;;IF THIS DIFFERENT FROM PREVIOUS
          IFNB <L1>,<
            OP (<AC>,L1,..MSK)> ;;DO PREVIOUS
          ..MSK==0>             ;;REINIT MASK
        IFNB <L2>,<
          ..MSK=..MSK!<S2>>>    ;;ACCUMULATE MASK

;..STR5 - SAME AS ..STR4 EXCEPT GIVES EXTRA ARG IF MORE STUFF TO
;FOLLOW.

DEFINE ..STR5 (OP,AC,S1,L1,S2,L2)<
        IFDIF <L1><L2>,<        ;;IF THIS DIFFERENT FROM PREVIOUS,
          IFNB <L1>,<
            IFNB <L2>,<         ;;IF MORE TO COME,
                OP'1 (AC,L1,..MSK)> ;;DO VERSION 1
            IFB <L2>,<          ;;IF NO MORE,
                OP'2 (AC,L1,..MSK)>> ;;DO VERSION 2
          ..MSK==0>             ;;REINIT MASK
        IFNB <L2>,<
          ..MSK=..MSK!<S2>>>    ;;ACCUMULATE MASK
```

```
;'REMOTE' MACROS USED TO BUILD UP ARG LIST

;INITIALIZE CONS -- DEFINES CONS

DEFINE ..ICNS <
   DEFINE ..CNS (ARG)<
        ..CNS2 <ARG>,>

   DEFINE ..CNS2 (NEW,OLD)<
        DEFINE ..CNS (ARG)<
          ..CNS2 <ARG>,<OLD'NEW>>>
   >

;GET CONS -- EXECUTE STRING ACCUMULATED

DEFINE ..GCNS <
   DEFINE ..CNS2 (NEW,OLD)<
          OLD>                  ;;MAKE ..CNS2 DO THE STUFF
        ..CNS ()>               ;;GET ..CNS2 CALLED WITH THE STUFF
```

```
;SPECIFIC CASES

;LOAD, STORE
; AC - AC OPERAND
; STR - STRUCTURE NAME
; Y - (OPTIONAL) ADDITIONAL SPECIFICATION OF DATA LOCATION

DEFINE LOAD (AC,STR,Y)<
        ..STR0 (..LDB,AC,STR,Y)>

   DEFINE ..LDB (AC,LOC,MSK)<
        ..TSIZ (..PST,MSK)
        .CASE ..PST,<<
                MOVE AC,LOC>,<
                HRRZ AC,LOC>,<
                HLRZ AC,LOC>,<
                LDB AC,[POINTR (LOC,MSK)]>>>

DEFINE STOR (AC,STR,Y)<
        ..STR0 (..DPB,AC,STR,Y)>

   DEFINE ..DPB (AC,LOC,MSK)<
        ..TSIZ (..PST,MSK)
        .CASE ..PST,<<
                MOVEM AC,LOC>,<
                HRRM AC,LOC>,<
                HRLM AC,LOC>,<
                DPB AC,[POINTR (LOC,MSK)]>>>

;SET TO ZERO

DEFINE SETZRO (STR,Y)<
        ..STR1 (..TQZ,,<STR>,Y,..STR4)>

   DEFINE ..TQZ (AC,LOC,MSK)<
        ..TSIZ (..PST,MSK)      ;;SET ..PST TO CASE NUMBER
        .CASE ..PST,<<
                SETZM LOC>,<    ;;FULL WORD
                HLLZS LOC>,<    ;;RH
                HRRZS LOC>,<    ;;LH
          ..TSAC (..ACT,LOC)    ;;SEE IF LOC IS AC
          .IF0 ..ACT,<
                MOVX .SAC,MSK   ;;NOT AC
                ANDCAM .SAC,LOC>,<
                ..TX (Z,LOC,MSK)>>>>
```

```
;SET TO ONE

DEFINE SETONE (STR,Y)<
        ..STR1 (..TQO,,<STR>,Y...STR4)>

   DEFINE ..TQO (AC,LOC,MSK)<
        ..TSIZ (..PST,MSK)
        .CASE ..PST,<<
                SETOM LOC>,<
                HLLOS LOC>,<
                HRROS LOC>,<
          ..TSAC (..ACT,LOC)
          .IF0 ..ACT,<
                MOVX .SAC,MSK
                IORM .SAC,LOC>,<
                ..TX (O,LOC,MSK)>>>>

;SET TO COMPLEMENT

DEFINE SETCMP (STR,Y)<
        ..STR1 (..TQC,,<STR>,Y,..STR4)>

   DEFINE ..TQC (AC,LOC,MSK)<
        ..TSIZ (..PST,MSK)
        .IF0 ..PST,<              ;;IF FULL WORD,
                SETCMM LOC>,<     ;;CAN USE SETCMM
          ..TSAC (..ACT,LOC)      ;;OTHERWISE, CHECK FOR AC
          .IF0 ..ACT,<
                MOVX .SAC,MSK
                XORM .SAC,LOC>,<
                ..TX(C,LOC,MSK)>>>
```

```
;INCREMENT, DECREMENT FIELD

DEFINE INCR (STR,Y)<
        ..STR0 (.INCR0,,<STR>,Y)>

   DEFINE .INCR0 (AC,LOC,MSK)<
        ..PST==MSK&<-MSK>       ;;GET LOWEST BIT
        .IF0 ..PST-1,<
                AOS LOC>,<      ;;BIT 35, CAN USE AOS
                MOVX .SAC,..PST ;;LOAD A ONE IN THE APPROPRIATE POSITION
                ADDM .SAC,LOC>>

DEFINE DECR (STR,Y)<
        ..STR0 (.DECR0,,<STR>,Y)>

   DEFINE .DECR0 (AC,LOC,MSK)<
        ..PST==MSK&<-MSK>
        .IF0 ..PST-1,<
                SOS LOC>,<      ;;BIT 35, CAN USE SOS
                MOVX .SAC,-..PST ;;LOAD -1 IN APPROPRIATE POSITION
                ADDM .SAC,LOC>>

;GENERAL DEFAULT, TAKES OPCODE

DEFINE OPSTR (OP,STR,Y)<
        ..STR0 (.OPST1,<OP>,<STR>,Y)>

   DEFINE .OPST1 (OP,LOC,MSK)<
        ..TSIZ (..PST,MSK)
        .IF0 ..PST,<
                OP LOC>,<       ;;FULL WORD, USE GIVEN OP DIRECTLY
                ..LDB .SAC,LOC,MSK ;;OTHERWISE, GET SPECIFIED BYTE
                OP .SAC>>

DEFINE OPSTRM (OP,STR,Y)<
        ..STR0 (.OPST2,<OP>,<STR>,Y)>

   DEFINE .OPST2 (OP,LOC,MSK)<
        ..TSIZ (..PST,MSK)
        .IF0 ..PST,<
                OP LOC>,<       ;;FULL WORD, USE OP DIRECTLY
                ..LDB .SAC,LOC,MSK
                OP .SAC
                ..DPB .SAC,LOC,MSK>>
```

```
;JUMP IF ALL FIELDS ARE 0 (ONE REGISTER AT MOST)

DEFINE JE (STR,Y,BA)<
        ..STR1 (..JE,<BA>,<STR>,Y,..STR3)>

   DEFINE ..JE (BA,LOC,MSK)<
        ..TSAC (..ACT,LOC)      ;;SEE IF AC
        .IF0 ..ACT,<
          ..TSIZ (..PST,MSK)    ;;SEE WHICH CASE
          .CASE ..PST,<<
                SKIPN LOC       ;;FULL WORD, TEST IN MEMORY
                JRST BA>,<
                HRRZ .SAC,LOC   ;;RIGHT HALF, GET IT
                JUMPE .SAC,BA>,<
                HLRZ .SAC,LOC   ;;LEFT HALF, GET IT
                JUMPE .SAC,BA>,<
                MOVE .SAC,LOC   ;;NOTA, GET WORD
                JXE (.SAC,MSK,<BA>)>>>,<
          JXE (LOC,MSK,<BA>)>>

;JUMP IF NOT ALL FIELDS ARE 0 (ONE REGISTER AT MOST)

DEFINE JN (STR,Y,BA)<
        ..STR1 (..JN,<BA>,<STR>,Y,..STR3)>

   DEFINE ..JN (BA,LOC,MSK)<
        ..TSAC (..ACT,LOC)      ;;SEE IF AC
        .IF0 ..ACT,<
          ..TSIZ (..PST,MSK)
          .CASE ..PST,<<
                SKIPE LOC       ;;FULL WORD, TEST IN MEMORY
                JRST BA>,<
                HRRZ .SAC,LOC   ;;RIGHT HALF, GET IT
                JUMPN .SAC,BA>,<
                HLRZ .SAC,LOC   ;;LEFT HALF, GET IT
                JUMPN .SAC,BA>,<
                MOVE .SAC,LOC   ;;NOTA, GET WORD
                JXN (.SAC,MSK,<BA>)>>>,<
          JXN (LOC,MSK,<BA>)>>
```

```
;JOR - JUMP ON 'OR' OF ALL FIELDS

DEFINE JOR (STR,Y,BA)<
        ..STR1 (..JN,<BA>,<STR>,Y,..STR4)>

;JNAND - JUMP ON NOT 'AND' OF ALL FIELDS

DEFINE JNAND (STR,Y,BA)<
        ..STR1 (..JNA3,<BA>,<STR>,Y,..STR4)>

   DEFINE ..JNA3 (BA,LOC,MSK)<
        ..TSAC (..ACT,LOC)
        .IF0 ..ACT,<
          SETCM .SAC,LOC        ;;NOT AC, GET COMPLEMENT OF WORD
          JXN (.SAC,MSK,<BA>)>,<        ;;JUMP IF ANY BITS ORIGINALLY OFF
          JXF (LOC,MSK,<BA>)>>  ;;DO AC CASE
```

```
;JAND - JUMP ON 'AND' OF ALL FIELDS

DEFINE JAND (STR,Y,BA,%TG)<
        ..STR1 (..JAN,<%TG,<BA>>,<STR>,Y,..STR5)
%TG:>

   DEFINE ..JAN1 (BA1,BA2,LOC,MSK)<
        ..JNA3 (BA1,LOC,MSK)>    ;;DO JUMP NAND TO LOCAL TAG

   DEFINE ..JAN2 (BA1,BA2,LOC,MSK)<
        ..TSAC (..ACT,LOC)
        .IF0 ..ACT,<
          SETCM .SAC,LOC        ;;NOT AC, GET COMPLEMENT OF WORD
          JXE (.SAC,MSK,<BA2>)>,<       ;;JUMP IF ALL BITS ORIGINALLY ONES
          JXO (LOC,MSK,<BA2>)>> ;;DO AC CASE

;JNOR - JUMP ON NOT 'OR' OF ALL FIELDS

DEFINE JNOR (STR,Y,BA,%TG)<
        ..STR1 (..JNO,<%TG,<BA>>,<STR>,Y,..STR5)
%TG:>

   DEFINE ..JNO1 (BA1,BA2,LOC,MSK)<
        ..JN (BA1,LOC,MSK)>     ;;DO JUMP OR TO LOCAL TAG

   DEFINE ..JNO2 (BA1,BA2,LOC,MSK)<
        ..JE (<BA2>,LOC,MSK)>   ;;DO JUMP NOR TO GIVEN TAG

;TEST AND MODIFY GROUP USING DEFINED STRUCTURES.  TEST-ONLY AND
;MODIFY-ONLY PROVIDED FOR COMPLETENESS.
;GENERATES EXACTLY ONE INSTRUCTION

DEFINE ..DOTY (M,T)<            ;;MACRO TO DEFINE ALL CASES
        IRP M,<
        IRP T,<
          DEFINE TQ'M'T (STR,Y)<
            ..STR1 (..TY,M'T,<STR>,Y,..STR3)>>>>

        ..DOTY (<N,O,Z,C>,<,E,N,A>) ;DO 16 DEFINES
        PURGE ..DOTY
```

```
;SPECIAL DEFINE FOR THE TWO CASES WHICH CAN TAKE MEMORY ARG
;*NOTE* MAY GENERATE MORE THAN ONE INSTRUCTION - CANNOT BE SKIPPED

DEFINE TMNE (STR,Y)<
        ..STR1 (..TYNE,,<STR>,Y,..STR3)>

DEFINE ..TYNE (MT,LOC,MSK)<
        ..TSAC (..ACT,LOC)      ;;SEE IF LOC IS AC
        .IF0 ..ACT,<
          ..JX1==MSK
          .IF0 <..JX1-1B0>,<
                SKIPGE LOC>,<
            .IF0 <..JX1+1>,<
                SKIPE LOC>,<
                MOVE .SAC,LOC
                TXNE .SAC,MSK>>>,<
                TXNE LOC,MSK>>

DEFINE TMNN (STR,Y)<
        ..STR1 (..TYNN,,<STR>,Y,..STR3)>

DEFINE ..TYNN (MT,LOC,MSK)<
        ..TSAC (..ACT,LOC)      ;;SEE IF LOC IS AC
        .IF0 ..ACT,<
          ..JX1==MSK
          .IF0 <..JX1-1B0>,<
                SKIPL LOC>,<
            .IF0 <..JX1+1>,<
                SKIPN LOC>,<
                MOVE .SAC,LOC
                TXNN .SAC,MSK>>>,<
                TXNN LOC,MSK>>

;ALL TY MACROS CALL ..TY AFTER INITIAL STRUCTURE PROCESSING

DEFINE ..TY (MT,LOC,MSK)<
        ..TSAC (..ACT,LOC)      ;;SEE IF LOC IS AC
        .IF0 ..ACT,<
                PRINTX ?TQ'MT - LOC NOT IN AC>,<
                TX'MT LOC,MSK>>
```

```
        SUBTTL BLOCK MACROS

;MACROS TO PROVIDE SOME BLOCK HANDLING OF CODE

;DO. - LOOP STRUCTURE, DECLARES TOP OF LOOP
; LOOP. - JUMPS TO TOP OF LOOP
; EXIT. - EXITS LOOP
; TOP. - TAG AT TOP OF LOOP FOR JUMPS, E.G. SOJG T4,TOP.
; ENDLP. - TAG AT END OF LOOP FOR JUMPS, E.G. SOJL T4,ENDLP.

DEFINE DO. (%TGB,%TGE)<
        ..SVLD                          ;;SAVE CURRENT BLOCK
%TGB:!                                  ;;TOP OF LOOP
  DEFINE OD. <
%TGE:!                                  ;;END OF LOOP
        .POPX>                          ;;RESTORE DEFS
  DEFINE LOOP. <
        JRST %TGB>                      ;;LOOP TO TOP
  DEFINE TOP. <%TGB>                    ;;LABEL AT TOP FOR JUMPS
  DEFINE ENDLP. <%TGE>                  ;;LABEL AT END FOR JUMPS
  DEFINE EXIT. <
        JRST %TGE>>                     ;;EXIT LOOP

DEFINE ENDDO. <
        OD.>

DEFINE ..SVLD (%SY1,%SY2,%SY3,%SY4)<
        SYN OD.,%SY1
        SYN LOOP.,%SY2
        SYN TOP.,%SY3
        SYN EXIT.,%SY4
        .PSHX <
         SYN %SY1,OD.
         SYN %SY2,LOOP.
         SYN %SY3,TOP.
         SYN %SY4,EXIT.>>
```

```
;IFNSK., IFSKP. - "IF NO SKIP", "IF SKIP"

;These macros cause the following code to be conditionally executed
;depending on whether the preceding instruction(s) skipped or not.
;The following code is ended with ENDIF., with ELSE. optional
;within the range.

;Note: both of these result in the same or fewer instructions than
;the use of literals to handle the same cases.
;Also, since the code is not in literals, the binary appears in the
;listing, and the code is easier to follow with DDT.
;If the preceding skip can be written in either sense, it is better
;to use IFSKP. because one fewer instructions will be generated.

;IFSKP. and IFNSK. have an alternate form where the consequence code
;is given as a macro argument.  In the normal case, no macro argument is given.

;"IF NO SKIP" CONSEQUENCE-CODE ALTERNATIVE-CODE
;If the instruction(s) preceding the macro does not skip, the 'consequence
; code' will be executed; otherwise (i.e. if the instruction skips) the
; 'alternative code' will be executed.

DEFINE IFNSK. (NSCOD,SKCOD,%TG1,%TG2)<
   IFB <NSCOD'SKCOD>,<           ;;THE REGULAR FORM
        ..SVDF                   ;;SAVE DEFINITIONS OF OUTER BLOCK
        TRNA                     ;;SKIP
        JRST %TG1                ;;JUMP PAST CODE
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>          ;;SAVE THE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>          ;;SAVE THE END TAG
        >
   IFNB <NSCOD'SKCOD>,<          ;;THE ALTERNATE FORM
        JRST %TG1                ;;THE NOSKIP CASE
        SKCOD
        JRST %TG2
%TG1:!  NSCOD
%TG2:!>>
```

```
;If JSYS Error

DEFINE IFJER. (NSCOD,SKCOD,%TG1,%TG2,%TG3)<
   IFB <NSCOD'SKCOD>,<              ;;THE REGULAR FORM
        ..SVDF                      ;;SAVE DEFINITIONS OF OUTER BLOCK
        ERJMP %TG3                  ;;SKIP
        JRST %TG1                   ;;JUMP PAST CODE
%TG3:!
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>             ;;SAVE THE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>             ;;SAVE THE END TAG
        >
   IFNB <NSCOD'SKCOD>,<             ;;THE ALTERNATE FORM
        ERJMP %TG1                  ;;THE NOSKIP CASE
        SKCOD
        JRST %TG2
%TG1:!  NSCOD
%TG2:!>>

;OBSOLETE NAME

DEFINE IFNES. (ARG1,ARG2)<
        PRINTX % IFNES. should be changed to IFJER.
        IFJER. <ARG1>,<ARG2>>


;"IF SKIP" CONSEQUENCE-CODE
;If the instruction(s) preceding the macro skips, the 'consequence
; code' will be executed.

DEFINE IFSKP. (SKCOD,%TG,%TG2)<
   IFB <SKCOD>,<                    ;;REGULAR FORM
        ..SVDF                      ;;SAVE DEFINITIONS OF OUTER BLOCK
        JRST %TG
   DEFINE ..TAGF (INST,PCT)<
        INST %TG''PCT>              ;;SAVE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>             ;;SAVE END TAG
        >
   IFNB <SKCOD>,<
        JRST %TG
        SKCOD
%TG:!>>
```

```
;If No JSYS Error

DEFINE IFNJE. (SKCOD,%TG,%TG2)<
   IFB <SKCOD>,<                    ;;REGULAR FORM
        ..SVDF                      ;;SAVE DEFINITIONS OF OUTER BLOCK
        ERJMP %TG
   DEFINE ..TAGF (INST,PCT)<
        INST %TG''PCT>              ;;SAVE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>             ;;SAVE END TAG
        >
   IFNB <SKCOD>,<
        ERJMP %TG
        SKCOD
%TG:!>>

;OBSOLETE NAME

DEFINE IFESK. (ARG)<
        PRINTX % IFESK. should be changed to IFNJE.
        IFNJE. <ARG>>
```

```
;CONDITIONALS WHICH REPRESENT JUMP CASES - I.E. AC L, LE, G, ETC.
; IF CONDITION IS SATISFIED, DO BRACKETTED CODE

DEFINE IFE. (AC,%TG1,%TG2)<
        JUMPN AC,%TG1               ;;JUMP IF NOT CONDITION
        ..SVDF                      ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>             ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>             ;;DEFINE END TAG
        >

DEFINE IFN. (AC,%TG1,%TG2)<
        JUMPE AC,%TG1               ;;JUMP IF NOT CONDITION
        ..SVDF                      ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>             ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>             ;;DEFINE END TAG
        >

DEFINE IFG. (AC,%TG1,%TG2)<
        JUMPLE AC,%TG1              ;;JUMP IF NOT CONDITION
        ..SVDF                      ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>             ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>             ;;DEFINE END TAG
        >

DEFINE IFGE. (AC,%TG1,%TG2)<
        JUMPL AC,%TG1               ;;JUMP IF NOT CONDITION
        ..SVDF                      ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>             ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>             ;;DEFINE END TAG
        >

DEFINE IFLE. (AC,%TG1,%TG2)<
        JUMPG AC,%TG1               ;;JUMP IF NOT CONDITION
        ..SVDF                      ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>             ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>             ;;DEFINE END TAG
        >
```

```
DEFINE IFL. (AC,%TG1,%TG2)<
        JUMPGE AC,%TG1          ;;JUMP IF NOT CONDITION
        ..SVDF                  ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>         ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>         ;;DEFINE END TAG
        >

DEFINE IFXE. (AC,MASK,%TG1,%TG2)<
        JXN AC,MASK,%TG1        ;;JUMP IF NOT CONDITION
        ..SVDF                  ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>         ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>         ;;DEFINE END TAG
        >

DEFINE IFXN. (AC,MASK,%TG1,%TG2)<
        JXE AC,MASK,%TG1        ;;JUMP IF NOT CONDITION
        ..SVDF                  ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>         ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>         ;;DEFINE END TAG
        >

DEFINE IFQE. (STR,Y,%TG1,%TG2)<
        JN <STR>,<Y>,%TG1       ;;JUMP IF NOT CONDITION
        ..SVDF                  ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>         ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>         ;;DEFINE END TAG
        >

DEFINE IFQN. (STR,Y,%TG1,%TG2)<
        JE <STR>,<Y>,%TG1       ;;JUMP IF NOT CONDITION
        ..SVDF                  ;;SAVE OUTER BLOCK
   DEFINE ..TAGF (INST,PCT)<
        INST %TG1''PCT>         ;;DEFINE FALSE TAG
   DEFINE ..TAGE (INST,PCT)<
        INST %TG2''PCT>         ;;DEFINE END TAG
        >
```

```
;GENERAL CASES WITHIN CONDITIONALS

;"AND SKIP"

DEFINE ANSKP. <
        ..TAGF (JRST,)>         ;;JUMP TO 'FALSE'

DEFINE ANNSK. <
        TRNA
        ..TAGF (JRST,)>         ;;JUMP TO 'FALSE'

DEFINE ELSE. <....U>            ;;UNDEFINED UNTIL BLOCK ENTERED
DEFINE ENDIF. <....U>
DEFINE ..TAGF <....U>
DEFINE ..TAGE <....U>

;"AND E" ETC.

DEFINE ANDE. (AC)<
        ..TAGF (<JUMPN AC,>,)>  ;;JUMP IF NOT CONDITION

DEFINE ANDN. (AC)<
        ..TAGF (<JUMPE AC,>,)>  ;;JUMP IF NOT CONDITION

DEFINE ANDG. (AC)<
        ..TAGF (<JUMPLE AC,>,)> ;;JUMP IF NOT CONDITION

DEFINE ANDGE. (AC)<
        ..TAGF (<JUMPL AC,>,)>  ;;JUMP IF NOT CONDITION

DEFINE ANDLE. (AC)<
        ..TAGF (<JUMPG AC,>,)>  ;;JUMP IF NOT CONDITION

DEFINE ANDL. (AC)<
        ..TAGF (<JUMPGE AC,>,)> ;;JUMP IF NOT CONDITION

DEFINE ANDXE. (AC,MASK)<
        ..TAGF (<JXN AC,MASK,>,)> ;;JUMP IF NOT CONDITION

DEFINE ANDXN. (AC,MASK)<
        ..TAGF (<JXE AC,MASK,>,)> ;;JUMP IF NOT CONDITION

DEFINE ANDQE. (STR,Y)<
        ..TAGF (<JN <STR>,<Y>,>,)> ;;JUMP IF NOT CONDITION

DEFINE ANDQN. (STR,Y)<
        ..TAGF (<JE <STR>,<Y>,>,)> ;;JUMP IF NOT CONDITION
```

```
;LOCAL WORKER MACROS

;THIS INITS THE DEFINITIONS OF ELSE. AND ENDIF. WHEN ENTERING A
;NEW BLOCK.

DEFINE ..INDF <
 DEFINE ELSE. <
        ..TAGE (JRST,)          ;;JUMP TO END
        ..TAGF (,<:!>)          ;;DEFINE THE FALSE TAG
        SYN ..TAGE,..TAGF       ;;MAKE FALSE EQUIVALENT TO END
   DEFINE ELSE. <....U>>        ;;ELSE CAN APPEAR ONCE ONLY

 DEFINE ENDIF. <
        ..TAGF (,<:!>)          ;;DEFINE FALSE TAG
        ..RSDF>                 ;;RESTORE DEFINITIONS OF OUTER BLOCK
   >
```

```
;SAVE DEFINITIONS

DEFINE ..SVDF (%SY1,%SY2,%SY3,%SY4)<
        SYN ELSE.,%SY1
        SYN ENDIF.,%SY2
        SYN ..TAGF,%SY3
        SYN ..TAGE,%SY4
        .PSHX <
          SYN %SY1,ELSE.
          SYN %SY2,ENDIF.
          SYN %SY3,..TAGF
          SYN %SY4,..TAGE>
        ..INDF                  ;;REINIT DEFS
              >

DEFINE ..RSDF <
        .POPX>

;MACROS TO PUSH/POP STRINGS

DEFINE .PSHX (STUFF)<
        .PSHX1 (.PSHX2,<STUFF>)>

DEFINE .PSHX1 (WCH,STUFF)<
        WCH (<STUFF>)>

DEFINE .PSHX2 (OLD)<
   DEFINE .PSHX1 (WCH,STUFF)<
        WCH (<<STUFF>,<OLD>>)>>

DEFINE .POPX <
        .PSHX1 (.POPX2)>

DEFINE .POPX2 (STUFF)<
        .POPX4 STUFF>

DEFINE .POPX4 (JUNK,STUFF)<
        .POPX3 STUFF>

DEFINE .POPX3 (TOP,REST)<
        TOP
   DEFINE .PSHX1 (WCH,STUFF)<
        WCH (<<STUFF>,<REST>>)>>
```

```
        SUBTTL CALL, RET, JSERR

   IFE REL,<
        EXTERN JSERR0,JSMSG0,JSHLT0,R,RSKP>

;CALL AND RETURN

.AC1==1                         ;ACS FOR JSYS ARGS
.AC2==2
.AC3==3
.A16==16                        ;TEMP FOR STKVAR AND TRVAR
P=17                            ;STACK POINTER

OPDEF CALL [PUSHJ P,0]
OPDEF RET [POPJ P,0]

;ABBREVIATION FOR  CALL, RET, RETSKP

OPDEF CALLRET [JRST]
.NODDT CALLRET

DEFINE RETSKP <JRST RSKP>

;MACRO TO PRINT MESSAGE ON TERMINAL

DEFINE TMSG ($MSG)<
        HRROI .AC1,[ASCIZ \$MSG\]
        PSOUT>

;MACRO TO OUTPUT MESSAGE TO FILE
; ASSUMES JFN ALREADY IN .AC1

DEFINE FMSG ($MSG)<
        HRROI .AC2,[ASCIZ \$MSG\]
        MOVEI .AC3,0
        SOUT>

;MACRO TO PRINT MESSAGE FOR LAST ERROR, RETURNS +1

DEFINE PERSTR ($MSG)<
   IFNB <$MSG>,<
        TMSG <$MSG>>
        CALL JSMSG0>

;MACRO TO PRINT JSYS ERROR MESSAGE, RETURNS +1 ALWAYS

OPDEF JSERR[<CALL JSERR0>]
OPDEF EJSERR[<JUMP 17,JSERR0>]  ;Since MACRO couldn't handle OPDEF of an OPDEF
                                ;  (i.e. ERCAL) defined elsewhere, use JUMP 17
                                ;  instead

;MACRO FOR FATAL JSYS ERROR, PRINTS MSG THEN HALTS

OPDEF JSHLT[<CALL JSHLT0>]
OPDEF EJSHLT[<JUMP 17,JSHLT0>]  ;Since MACRO couldn't handle OPDEF of an OPDEF
                                ;  (i.e. ERCAL) defined elsewhere, use JUMP 17
                                ;  instead

;PRINT ERROR MESSAGE IF JSYS FAILS

DEFINE ERMSG(TEXT),<
        ERJMP   [TMSG <? TEXT>
```

```
                JSHLT]
>

;MAKE SYMBOLS EXTERN IF NOT ALREADY DEFINED

DEFINE EXT (SYM)<
   IF2,<
        IRP SYM,<
        IFNDEF SYM,<EXTERN SYM
        SUPPRE SYM>>>>
```

```
;MACRO TO ADD BREAK CHARACTER TO FOUR WORD BREAK MASK (W0., W1., W2., W3.)

DEFINE BRKCH. (%%V,V2)
<
%%FOO==%%V
        BRK0 (%%FOO,V2,0)
>

;MACRO TO REMOVE CHARACTER

DEFINE UNBRK. (%%V,V2)
<
%%FOO==%%V
        BRK0 (%%FOO,V2,1)
>

DEFINE BRK0 (%%11,V2,FLAVOR)
<       ..V22==%%11
        ..V1==%%11
        IFNB <V2>,<..V22==V2>
REPEAT ..V22-<%%11>+1,< ;;BRACKETS AROUND %%11 IN CASE ITS AN EXPRESSION
        %%W==..V1/^D32  ;;DECIDE WHICH WORD CHARACTER GOES IN
        %%X==..V1-%%W*^D32      ;;CALCULATE BIT POSITION WITHIN WORD
        IFE FLAVOR,BRKC1 \"<%%W+"0">    ;;MODIFY CORRECT MASK WORD
        IFN FLAVOR,BRKC2 \"<%%W+"0">
        ..V1==..V1+1
                >
>

DEFINE BRKC1 (ARG1)
<       W'ARG1'.==W'ARG1'.!<1B<%%X>>
>

DEFINE BRKC2 (ARG1)
<       W'ARG1'.==W'ARG1'.&<-1-1B<%%X>>
>

;MACRO TO INITIALIZE 4-WORD 12-BIT CHARACTER BREAK MASK

DEFINE BRINI.(A0<0>,A1<0>,A2<0>,A3<0>)
<
W0.==A0
W1.==A1                         ;INITIALIZE BREAK MASK
W2.==A2
W3.==A3
>

;MACRO TO DEFINE A BREAK SET

DEFINE BRMSK. (INI0,INI1,INI2,INI3,ALLOW,DISALW)
<       BRINI. INI0,INI1,INI2,INI3      ;;SET UP INITIAL MASK
        IRPC ALLOW,<    UNBRK. "ALLOW"> ;;DON'T BREAK ON CHARS TO BE ALLOWED IN FI
        IRPC DISALW,<   BRKCH. "DISALW">        ;;BREAK ON CHARACTERS NOT ALLOWED
        EXP W0.,W1.,W2.,W3.             ;;STORE RESULTANT MASK IN MEMORY
>
```

```
;COMND - MACRO FOR BUILDING FUNCTION DESCRIPTOR BLOCK
;THIS IS THE OLD ONE, BEFORE .CMBRK EXISTED.  USE FLDBK. FOR SPECIFYING
;BREAK SETS

DEFINE FLDDB. (TYP,FLGS,DATA,HLPM,DEFM,LST)<
        ..XX==<FLD(TYP,CM%FNC)>+FLGS+<0,,LST>
    IFNB <HLPM>,<..XX=CM%HPP!..XX>
    IFNB <DEFM>,<..XX=CM%DPP!..XX>
        ..XX
    IFNB <DATA>,<DATA>
    IFB <DATA>,<0>
    IFNB <HLPM>,<POINT 7,[ASCIZ ^VHLPM^V]>
    IFB <HLPM>,<IFNB <DEFM>,<0>>
    IFNB <DEFM>,<POINT 7,[ASCIZ \DEFM\]>>

;COMND - MACRO FOR BUILDING FUNCTION DESCRIPTOR BLOCK

DEFINE FLDBK. (TYP,FLGS,DATA,HLPM,DEFM,BRKADR,LST)<
        ..XX==<FLD(TYP,CM%FNC)>+FLGS+<Z LST>
    IFNB <HLPM>,<..XX=CM%HPP!..XX>
    IFNB <DEFM>,<..XX=CM%DPP!..XX>
    IFNB <BRKADR>,<..XX=CM%BRK!..XX>
        ..XX
    IFNB <DATA>,<DATA>
    IFB <DATA>,<0>
    IFNB <HLPM>,<POINT 7,[ASCIZ ^VHLPM^V]>
    IFB <HLPM>,<IFNB <DEFM'BRKADR>,<0>>
    IFB <DEFM>,<IFNB <BRKADR>,<0>>
    IFNB <DEFM>,<POINT 7,[ASCIZ \DEFM\]>
    IFNB <BRKADR>,<BRKADR>
    >
```

```
;USEFUL EXTENDED ADDRESSING DEFINITIONS

OPDEF   XMOVEI [SETMI]          ;EXTENDED MOVE IMMEDIATE
OPDEF   XHLLI [HLLI]            ;NOT YET IN MACRO

DEFINE XBLT. (A)<
        EXTEND A,[XBLT]>
```

```
        SUBTTL SUPPORT CODE FOR JSERR

   IFN REL,<

A=1
B=2
C=3
D=4

;JSYS ERROR HANDLER
;       CALL JSERR0
; RETURNS +1: ALWAYS, CAN BE USED IN +1 RETURN OF JSYS'S

JSERR0::MOVEI A,.PRIIN
        CFIBF                   ;CLEAR TYPAHEAD
        MOVEI A,.PRIOU
        DOBE                    ;WAIT FOR PREVIOUS OUTPUT TO FINISH
        TMSG <
? JSYS ERROR: >
JSMSG0::MOVEI A,.PRIOU
        HRLOI B,.FHSLF          ;SAY  THIS FORK ,, LAST ERROR
        SETZ C,
        ERSTR
         JFCL
         JFCL
        TMSG <
>
        RET

;FATAL JSYS ERROR - PRINT MESSAGE AND HALT
;       CALL JSHLT0
; RETURNS: NEVER

JSHLT0::CALL JSERR0             ;PRINT THE MSG
JSHLT1: HALTF
        TMSG <PROGRAM CANNOT CONTINUE
>
        JRST JSHLT1             ;HALT AGAIN IF CONTINUED
   >                            ;END OF IFN REL,
```

```
        SUBTTL STKVAR - STACK VARIABLE FACILITY

;MACRO FOR ALLOCATING VARIABLES ON THE STACK. ITS ARGUMENT IS
;A LIST OF ITEMS.  EACH ITEM MAY BE:
; 1. A SINGLE VARIABLE WHICH WILL BE ALLOCATED ONE WORD
; 2. A VARIABLE AND SIZE PARAMETER WRITTEN AS <VAR,SIZ>.  THE
;       VARIABLE WILL BE ALLOCATED THE SPECIFIED NUMBER OF WORDS.
;RETURN FROM A SUBROUTINE USING THIS FACILITY MUST BE VIA
;RET OR RETSKP.  A DUMMY RETURN WHICH FIXES UP THE STACK IS PUT ON
;THE STACK AT THE POINT THE STKVAR IS ENCOUNTERED.
;WITHIN THE RANGE OF A STKVAR, PUSH/POP CANNOT BE USED AS THEY WILL
;CAUSE THE VARIABLES (WHICH ARE DEFINED AS RELATIVE STACK LOCATIONS)
;TO REFERENCE THE WRONG PLACE.
;TYPICAL USE:   STKVAR <AA,BB,<QQ,5>,ZZ>
;               ENDSV.          ;END OF SCOPE OF NAMES

   IFE REL,<
        EXTERN .STKST,.STKRT>

DEFINE STKVAR (ARGS)<
        ..STKR==10              ;;REMEMBER RADIX
        RADIX 8
        ..STKN==0
        IRP ARGS,<
          .STKV1 (ARGS)>
        JSP .A16,.STKST
         ..STKN,,..STKN
        RADIX ..STKR
        DEFINE ENDSV.<.ENSV1 <ARGS>>
   >

;INTERMEDIATE MACRO TO PEAL OFF ANGLEBRACKETS IF ANY

DEFINE .STKV1 (ARG)<
        .STKV2 (ARG)>

;INTERMEDIATE MACRO TO CALCULATE OFFSET AND COUNT VARIABLES

DEFINE .STKV2 (VAR,SIZ)<
        IFB <SIZ>,<..STKN==..STKN+1>
        IFNB <SIZ>,<..STKN==..STKN+SIZ>
        ..STKQ==..STKN+1
        .STKV3 (VAR,\..STKQ)>

;INNERMOST MACRO TO DEFINE VARIABLE

DEFINE .STKV3 (VAR,LOC)<
   IFDEF VAR,<.IF VAR,SYMBOL,<PRINTX STKVAR VAR ALREADY DEFINED>>
        DEFINE VAR<-^O'LOC(P)>
        $'VAR==<Z VAR>>         ;SYMBOL FOR DDT

;CLEANUP NAMES

DEFINE .ENSV1 (ARGS)<
        IRP ARGS,<
          .ENSV2 (ARGS)>>

  DEFINE .ENSV2 (ARG)<
        .ENSV3 (ARG)>

  DEFINE .ENSV3 (ARG,SIZ)<
        DEFINE ARG<....U>>
```

```
   IFN REL,<

;COMMON ENTRY AND EXIT ROUTINE FOR STACK VARIABLE

        ENTRY .STKST

.STKST::ADD P,0(.A16)           ;BUMP STACK FOR VARIABLES USED
        JUMPGE P,STKSOV         ;TEST FOR STACK OVERFLOW
STKSE1: PUSH P,0(.A16)          ;SAVE BLOCK SIZE FOR RETURN
        PUSHJ P,1(.A16)         ;CONTINUE ROUTINE, EXIT TO .+1
.STKRT::JRST STKRT0             ;NON-SKIP RETURN COMES HERE
        POP P,.A16              ;SKIP RETURN COMES HERE-RECOVER COUNT
        SUB P,.A16              ;ADJUST STACK TO REMOVE BLOCK
        AOS 0(P)                ;NOW DO SKIP RETURN
        RET

STKRT0: POP P,.A16              ;RECOVER COUNT
        SUB P,.A16              ;ADJUST STACK TO REMOVE BLOCK
        RET                     ;DO NON-SKIP RETURN

STKSOV: SUB P,0(.A16)           ;STACK OVERFLOW- UNDO ADD
        HLL .A16,0(.A16)        ;SETUP TO DO MULTIPLE PUSH, GET COUNT
STKSO1: PUSH P,[0]              ;DO ONE PUSH AT A TIME, GET REGULAR
        SUB .A16,[1,,0]         ; ACTION ON OVERFLOW
        TLNE .A16,777777        ;COUNT DOWN TO 0?
        JRST STKSO1             ;NO, KEEP PUSHING
        JRST STKSE1
   >                            ;END OF IFN REL,
```

```
        SUBTTL TRVAR - TRANSIENT VARIABLE FACILITY

;TRANSIENT (STACK) VARIABLE FACILITY - EQUIVALENT TO STKVAR
;EXCEPT ALLOWS VARIABLES TO BE USED WITHIN LOWER LEVEL ROUTINES
;AND AFTER OTHER THINGS HAVE BEEN PUSHED ON STACK.
;N.B. USES .FP AS FRAME POINTER - MUST NOT BE CHANGED WHILE
;VARIABLES IN USE.

.FP==15                         ;DEFAULT FRAME POINTER

   IFE REL,<
        EXTERN .TRSET,.TRRET,.ASSET,.SASET,.ASRET>

DEFINE TRVAR (VARS)<
        ..TRR==10               ;;REMEMBER CURRENT RADIX
        RADIX 8
        ..NV==1                 ;;INIT COUNT OF STACK WORDS
        IRP VARS,<
          .TRV1 (VARS)>         ;;PROCESS LIST
        JSP .A16,.TRSET         ;;ALLOCATE STACK SPACE, SETUP .FP
         ..NV-1,,..NV-1
        RADIX ..TRR             ;;RESTORE RADIX
        DEFINE ENDTV.<.ENSV1 <VARS>>
        >

DEFINE .TRV1 (VAR)<
        .TRV2 (VAR)>            ;;PEEL OFF ANGLEBRACKETS IF ANY

DEFINE .TRV2 (NAM,SIZ)<
        .TRV3 (NAM,\..NV)       ;;DEFINE VARIABLE
        IFB <SIZ>,<..NV=..NV+1>
        IFNB <SIZ>,<..NV=..NV+SIZ>>

DEFINE .TRV3 (NAM,LOC)<
   IFDEF NAM,<.IF NAM,SYMBOL,<PRINTX TRVAR NAM ALREADY DEFINED>>
        DEFINE NAM<^O'LOC(.FP)>
        $'NAM==<Z NAM>>         ;;SYMBOL FOR DDT

;AC SUBROUTINE - ENTRY FOR SUBROUTINE CALLED WITH 1-4 ARGS IN ACS T1-T4.
;USES .FP AS FRAME PTR LIKE TRVAR

DEFINE ASUBR (ARGS)<
        ..TRR==10               ;;SAVE RADIX
        RADIX 8
        ..NV==1                 ;;INIT ARG COUNT
        IRP ARGS,<
          .TRV1 (ARGS)>         ;;DEFINE ARG SYMBOL
        IFG ..NV-5,<PRINTX ?TOO MANY ARGUMENTS: ARGS>
        JSP .A16,.ASSET         ;;SETUP STACK
        RADIX ..TRR             ;;RESTORE RADIX
        DEFINE ENDAS.<.ENSV1 <ARGS>>
        >
;SAME AS ABOVE EXCEPT ALSO RESTORES T1-T4 FROM STACK
```

```
DEFINE SASUBR (ARGS)<
        ..TRR==10                   ;;SAVE RADIX
        RADIX 8
        ..NV==1                     ;;INIT ARG COUNT
        IRP ARGS,<
          .TRV1 (ARGS)>             ;;DEFINE ARG SYMBOL
        IFG ..NV-5,<PRINTX ?TOO MANY ARGUMENTS: ARGS>
        JSP .A16,.SASET             ;;SETUP STACK
        RADIX ..TRR                 ;;RESTORE RADIX
        DEFINE ENDSA.<.ENSV1 <ARGS>>
        >
```

```
   IFN REL,<
;SUPPORT ROUTINE FOR TRVAR

.TRSET::PUSH P,.FP                 ;PRESERVE OLD .FP
        MOVE .FP,P                 ;SETUP FRAME PTR
        ADD P,0(.A16)              ;ALLOCATE SPACE
        JUMPGE P,TRSOV
TRSET1: PUSHJ P,1(.A16)            ;CONTINUE ROUTINE, EXIT VIA .+1
.TRRET::JRST [  MOVEM .FP,P        ;CLEAR STACK
                POP P,.FP          ;RESTORE OLD .FP
                POPJ P,]
        MOVEM .FP,P                ;HERE IF SKIP RETURN
        POP P,.FP
        AOS 0(P)                   ;PASS SKIP RETURN
        POPJ P,

TRSOV:  MOVE P,.FP                 ;STACK OVERFLOW, UNDO ADD
        PUSH P,.A16                ;SAVE LOCAL RETURN
        HRRZ .A16,0(.A16)          ;GET COUNT
        ADJSP P,-1(.A16)           ;ADJUST STACK, GET TRAP HERE OR ON PUSH
        MOVE .A16,1(.FP)           ;RESTORE LOCAL RETURN
        JRST TRSET1                ;NOW CHARGE AHEAD

;SUPPORT ROUTINE FOR ASUBR

.ASSET::PUSH P,.FP                 ;SAVE .FP
        MOVE .FP,P                 ;SETUP FRAME POINTER
        ADJSP P,4                  ;BUMP STACK
        DMOVEM A,1(.FP) ;SAVE ARGS
        DMOVEM C,3(.FP)
        PUSHJ P,0(.A16)            ;CONTINUE ROUTINE
.ASRET:: JRST [ MOVEM .FP,P        ;NO-SKIP RETURN, CLEAR STACK
                POP P,.FP
                POPJ P,]
        MOVEM .FP,P                ;SKIP RETURN, CLEAR STZCK
        POP P,.FP
        AOS 0(P)
        POPJ P,

;SUPPORT ROUTINE FOR SASUBR

.SASET::PUSH P,.FP                 ;SAVE .FP
        MOVE .FP,P                 ;SETUP FRAME POINTER
        ADJSP P,4                  ;BUMP STACK
        DMOVEM A,1(.FP) ;SAVE ARGS
        DMOVEM C,3(.FP)
        PUSHJ P,0(.A16)            ;CONTINUE ROUTINE
.SARET:: JRST [ DMOVE A,1(.FP)     ;RESTORE
                DMOVE C,3(.FP)
                MOVEM .FP,P        ;NO-SKIP RETURN, CLEAR STACK
                POP P,.FP
                POPJ P,]
        DMOVE A,1(.FP)             ;RESTORE
        DMOVE C,3(.FP)
        MOVEM .FP,P                ;SKIP RETURN, CLEAR STACK
        POP P,.FP
        AOS 0(P)
        POPJ P,
   >                               ;END OF IFN REL,
```

```
;AC VARIABLE FACILITY

   IFE REL,<
        EXTERN .SAV1,.SAV2,.SAV3,.SAV4,.SAV8>

.FPAC==5                        ;FIRST PRESERVED AC
.NPAC==10                       ;NUMBER OF PRESERVED ACS

DEFINE ACVAR (LIST)<
        ..NAC==0                ;;INIT NUMBER OF ACS USED
        IRP LIST,<
          .ACV1 (LIST)>         ;;PROCESS ITEMS
        .ACV3 (\..NAC)          ;;SAVE ACS USED
        DEFINE ENDAV.<.ENAV1 <LIST>>>

DEFINE .ACV1 (ITEM)<
        .ACV2 (ITEM)>           ;;PEEL OFF ANGLEBRACKETS IF ANY

DEFINE .ACV2 (NAM,SIZ)<
   IFDEF NAM,<.IF NAM,SYMBOL,<PRINTX ACVAR NAM ALREADY DEFINED>>
        NAM==.FPAC+..NAC        ;;DEFINE VARIABLE
        $'NAM==NAM              ;;FOR DDT
        IFB <SIZ>,<..NAC=..NAC+1>
        IFNB <SIZ>,<..NAC=..NAC+SIZ>>

DEFINE .ACV3 (N)<
        IFG N-.NPAC,<PRINTX ?TOO MANY ACS USED>
        IFLE N-4,<
          JSP .A16,.SAV'N>      ;;SAVE ACTUAL NUMBER USED
        IFG N-4,<
          JSP .A16,.SAV8>>      ;;SAVE ALL

DEFINE .ENAV1 (ARGS)<
        IRP ARGS,<
          .ENAV2 (ARGS)>>

  DEFINE .ENAV2 (ARG)<
        .ENAV3 (ARG)>

  DEFINE .ENAV3 (NAM,SIZ)<
        PURGE NAM,NAM
  >

   IFN REL,<
;SUPPORT ROUTINES FOR AC VARIABLE FACILITY

.SAV1:: PUSH P,.FPAC
        PUSHJ P,0(.A16)         ;CONTINUE PROGRAM
         SKIPA
        AOS -1(P)
        POP P,.FPAC
        POPJ P,

.SAV2:: PUSH P,.FPAC
        PUSH P,.FPAC+1
        PUSHJ P,0(.A16)
         SKIPA
        AOS -2(P)
        POP P,.FPAC+1
        POP P,.FPAC
        POPJ P,
```

```
.SAV3::
.SAV4:: PUSH P,.FPAC
        PUSH P,.FPAC+1
        PUSH P,.FPAC+2
        PUSH P,.FPAC+3
        PUSHJ P,0(.A16)
         SKIPA
        AOS -4(P)
        POP P,.FPAC+3
        POP P,.FPAC+2
        POP P,.FPAC+1
        POP P,.FPAC
        POPJ P,

.SAV8:: ADD P,[10,,10]
        JUMPGE P,[HALT .]
        DMOVEM .FPAC,-7(P)
        DMOVEM .FPAC+2,-5(P)
        DMOVEM .FPAC+4,-3(P)
        DMOVEM .FPAC+6,-1(P)
        PUSHJ P,0(.A16)
         SKIPA
        AOS -10(P)
        DMOVE .FPAC+6,-1(P)
        DMOVE .FPAC+4,-3(P)
        DMOVE .FPAC+2,-5(P)
        DMOVE .FPAC,-7(P)
        SUB P,[10,,10]
        POPJ P,
    >
```

```
;AC SAVE FACILITY - COMPILES OPEN PUSH'S
;       SAVEAC <LIST-OF-ACS>
;DUMMY ROUTINE PUT ON STACK TO CAUSE AUTOMATIC RESTORE. SUPPORTS
; +1 OR +2 RETURNS.

DEFINE SAVEAC (ACS)<
        .NAC==0
        IRP ACS,<
          PUSH P,ACS            ;;SAVE AN AC
          .NAC=.NAC+1>          ;;COUNT THEM
        .N1==.NAC
        SETMI .A16,[CAIA        ;;STACK DUMMY RETURN
                AOS -.N1(P)     ;;HANDLE SKIP RETURN
          IRP ACS,<
                .N1=.N1-1
                MOVE ACS,-.N1(P)>  ;;RESTORE AN AC
                SUB P,[.NAC,,.NAC]  ;;CLEAR STACK
                POPJ P,]        ;;FINAL RETURN
        PUSH P,.A16>

   IFN REL,<
;STANDARD RETURNS

RSKP::  AOS 0(P)
R::     RET
   >                            ;END OF IFN REL,
```

```
        SUBTTL BLSUBR - BLISS-STYLE SUBROUTINE MECHANISM

;MACROS FOR STACK-STYLE (BLISS) SUBROUTINE ENTRY
;BLSUBR DEFINE A SUBROUTINE ENTRY POINT.  IT TAKES THE LIST OF
;SYMBOLS WHICH WILL BE BOUND TO VALUES ON THE STACK AT ENTRY TO
;THE ROUTINE.  A STACK FRAME POINTER IS SETUP IN .FP AND MUST
;BE UNDISTURBED THROUGH THE ROUTINE.  OTHER MECHANISMS WHICH
;USE THE STACK (E.G. SAVEAC) CAN BE USED.
;AN OPTIONAL LIST OF VARIABLES IN THE SAME FORMAT AS FOR TRVAR CAN
;BE GIVEN TO ALLOCATE LOCAL DYNAMIC STORAGE.

;SUBROUTINES DEFINED HEREBY ARE CALLED WITH BLCALL.

   IFE REL,<
        EXTERN .ENTER>

DEFINE BLSUB. (ARGS,VARS)<          ;;ARGUMENTS, LOCAL VARIABLES
        ..TRR==10                   ;;REMEMBER CURRENT RADIX
        RADIX 8                     ;;SO BACKSLASH ARGS WILL WORK HEREIN
        ..NA==2                     ;;INIT ARG COUNT
        IRP ARGS,<
          ..NA=..NA+1>              ;;COUNT ARGS
        IRP ARGS,<
          .BLSU1(ARGS,\..NA)        ;;DEFINE AN ARG
          ..NA=..NA-1>
        ..NV==1                     ;;SETUP TO COUNT VARIABLE STORAGE
        IRP VARS,<
          .TRV1 (VARS)>             ;;COUNT WORDS AND DEFINE SYMBOLS
  DEFINE ENDBS. <.ENBS1 <ARGS>
              .ENSV1 <VARS>> ;;SAVE SYMBOLS
        JSP .A16,.ENTER
        ..NV-1,,..NV-1
        RADIX ..TRR>                ;;SETUP FRAME PTR

DEFINE .BLSU1 (ARG,LOC)<
        DEFINE ARG<-^O'LOC(.FP)>
        $'ARG==<Z ARG>>

DEFINE .ENBS1 (ARGS)<
        IRP ARGS,<
   DEFINE ARGS<....U>>>
```

```
;CALL STACK-STYLE (BLISS) SUBROUTINE
;THIS MACRO TAKES THE NAME OF THE SUBROUTINE AND A LIST OF ARGUMENTS.
;EACH ARGUMENT IN THE ARG LIST IS ONE OF THE FOLLOWING:
;  1. A NORMAL EFFECTIVE ADDRESS SPECIFICATION, E.G. FOO, @FIE(X)
;  2. AN IMMEDIATE ADDRESS WRITTEN AS <.,ADR> WHERE ADR IS AN EFFECTIVE
;       ADDRESS SPECIFICATION, E.G. FOO, @FIE(X).  NOTE THAT THIS
;       ADDRESS WILL BE COMPUTED BY AN XMOVEI AT THE TIME OF THE CALL
;       SO SECTION INFORMATION WILL BE BOUND AT THAT TIME.  NOTE ALSO
;       THAT THIS FORM SHOULD *NOT* BE USED FOR A LITERAL CONSTANT
;       WHERE YOU WOULD NOT WANT THE CURRENT SECTION PUT IN THE LEFT
;       HALF.  USE [CONST] INSTEAD.  YES, THE DOT HERE IS LIKE NO-DOT IN BLIS
;       AND VICE-VERSA.
;  3. A STRUCTURE REFERENCE SPECIFICATION, E.G. AAA, <BB,(X)>.  IF
;       THE LATTER FORM IS USED, THE BRACKETS ARE REQUIRED.

DEFINE BLCAL. (NAME,ARGS)<
        ..NA==0                 ;;INIT ARG COUNT
        IRP ARGS,<
          .BLCL2 ARGS>          ;;COMPILE PUSH
        PUSH P,[..NA+1,,..NA+1] ;;COUNT OF ARGS AND SELF
        PUSHJ P,NAME            ;;JUMP TO SUBR
   >

;SEPARATE PAIRED ARGS

  DEFINE .BLCL2 (ARGS)<
        .BLCL1 ARGS>

  DEFINE .BLCL1 (ARG1,ARG2)<
          IFIDN <ARG1><.>,<
                XMOVEI .A16,ARG2        ;;IMMEDIATE ARG
                PUSH P,.A16>
          IFDIF <ARG1><.>,<
          .IFATM <ARG1>,.BLF1   ;;SEE IF ARG IS ATOMIC
          IFN .BLF1,<
            .BLF1==0            ;;SET TO 1 IFF STRUCTURE REF
            .IF %'ARG1,MACRO,<  ;;CHECK RELATED STRUCTURE SYMBOL
                .BLF1==1>       ;;IS A STRUCTURE
            IFNB <ARG2>,<
                .BLF1==1>       ;;SECOND ARG IMPLIES STRUCTURE TOO
            IFN .BLF1,<         ;;'OR' OF ABOVE TWO CHECKS
                LOAD .A16,ARG1,ARG2
                PUSH P,.A16>>
          IFE .BLF1,<           ;IF WASN'T A STRUCTURE REF,
                PUSH P,ARG1>>   ;;PUSH ONE ARG
          ..NA=..NA+1>

;MACRO TO SEE IF STRING IS AN ATOM, I.E. CONTAINS ONLY LEGAL SYMBOL
;CONSTITUENTS A-Z, 0-9, %, $, .
;IT IS PAINFULLY SLOW, BUT MACRO PROVIDES NO OTHER WAY
;FLAG WILL BE SET TO 1 IF STRING IS ATOM, 0 OTHERWISE

DEFINE .IFATM (S,FLG)<
        IRPC S,<
        FLG==0
        IFGE "S"-"A",<IFLE "S"-"Z",<FLG=1>> ;;SET FLG IF LETTER OK
        IFGE "S"-"0",<IFLE "S"-"9",<FLG=1>>
        IFE "S"-"%",<FLG=1>
        IFE "S"-"$",<FLG=1>
        IFE "S"-".",<FLG=1>
        IFE FLG,<STOPI>>>
```

```
   IFN REL,<
;SUPPORT CODE FOR BLSUBR

.ENTER::PUSH P,.FP
        MOVE .FP,P
        ADD P,0(.A16)           ;ALLOCATE LOCAL STORAGE
        JUMPGE P,ENTOV          ;JUMP IF OVERFLOW
ENTOV1: PUSHJ P,1(.A16)
         JRST [ MOVE P,.FP      ;RESET STACK PTR
                JRST ENTX1]
        MOVE P,.FP
        AOS -1(P)               ;PROPAGATE SKIP
ENTX1:  POP P,.FP
        POP P,.A16
        SUB P,0(P)              ;REMOVE ARGS
        JRST 0(.A16)            ;RETURN

ENTOV:  MOVE P,.FP              ;STACK OVERFLOW, UNDO ADD
        PUSH P,.A16             ;SAVE LOCAL RETURN IN 1(.FP)
        HRRZ .A16,0(.A16)       ;GET COUNT
        ADJSP P,-1(.A16)        ;ALLOCATE SPACE, GET TRAP HERE OR ON PUSH
        MOVE .A16,1(.FP)        ;RESTORE LOCAL RETURN
        JRST ENTOV1             ;CHARGE AHEAD
   >                            ;END IFN REL
```

```
SUBTTL ERROR-MESSAGE SUPPORT FOR MACROS



;Macro to print current location, macro name, and text
DEFINE MPRNTX (MNAME,TEXT)
  <DEFINE ..MP. (LOCN,MTEXT,PTEXT)
    <PRINTX Location 'LOCN', Macro 'MTEXT': PTEXT
    >
  ..MP.(\.,MNAME,<TEXT>)
  PURGE ..MP.
  >

;Macro to print current location and text
DEFINE EPRNTX (TEXT)
  <DEFINE ..EP. (LOCN,PTEXT)
    <PRINTX Location 'LOCN': PTEXT
    >
  ..EP.(\.,<TEXT>)
  PURGE ..EP.
  >
```

```
        SUBTTL MACROS TO SUPPORT EXTENDED ADDRESSING

;       Local format indirect word
;       ===================================================================
;       !1!0!    Reserved    ! I !   X   !            ADDR                !
;       ===================================================================
;       !0!1!2             12! 13!14   17!18                            35!


        ;Macro to generate local-format (instruction-format) indirect words
        ;Args:
        ;       ADDR    18-bit in-section address (indexing or indirection
        ;               may be specified)

        ;Generates Q errors on the following:
        ;               Bits 0-12 non-zero in ADDR

        DEFINE LFIWM (ADDR)
        <..ERR.=0               ;;Reset error flag
        IFN <<ADDR>&<^O<777740,,0>>>,
          <MPRNTX(LFIWM,Bits 0 - 12 non-zero in address field: ADDR)
          ..ERR.=1
          >
        IFN ..ERR.,<-1,-1,-1>   ;;Generate Q error
        IFE ..ERR.,<1B0!<<^O<400037,,-1>>&<ADDR>>>    ;;Generate LFIW
        PURGE ..ERR.
        >
```

```
;       Global format indirect word
;       ================================================================
;       !0! I !   X   !        SEC         !              ADDR              !
;       ================================================================
;       !0! 1 !2     5!6                 17!                              35!


        ;Macro to generate global-format (extended-format) indirect words
        ;Args:
        ;       SEC     12-bit section number
        ;       ADDR    18-bit in-section address (indexing or indirection
        ;               may be specified)

        ;Generates Q errors on the following:
        ;               Bits 0-12 non-zero in ADDR
        ;               SEC greater than 12 bits

        DEFINE GFIWM (SEC,ADDR)
        <..ERR.=0               ;;Reset error flag
        IFN <<SEC>&<^O<-1,,770000>>>,
          <MPRNTX(GFIWM,Section greater than 12 bits: SEC)
          ..ERR.=1
          >
        IFN <<ADDR>&<^O<777740,,0>>>,
          <MPRNTX(GFIWM,Bits 0 - 12 non-zero in address field: ADDR)
          ..ERR.=1
          >
        IFN ..ERR.,<-1,-1,-1>   ;;Generate Q error
                                ;;Generate GFIW
        IFE ..ERR.,<
          <<<ADDR>_<^O14>>&<^O<370000,,0>>!<<ADDR>&<0,,-1>>!<<SEC>_<^O22>>>>
        PURGE ..ERR.
        >
```

```
;       The following macros generate all flavors of 1 and 2-word
;       global and local byte pointers.  They are similar to the
;       POINT pseudo-op, with the following exceptions:

;       1.      The basic argument triad of (bytesize,address,byte position)
;               is maintained.  However, some of the macros will prefix
;               and-or postfix the triad with additional argument(s).
;       2.      Numeric arguments are always interpreted in the current radix.
;               Assuming the current radix is octal, note the following
;               equivalences:
;                       a.  POINT 10,200,36
;                       b.  L1BPT(12,200,44)
;                       c.  L1BPT(^D10,200,^D36)
;       3.      Strict field-limits are enforced.  Any expression that
;               will not fit into its appropriate field will generate
;               an error message and cause a Q error.  Thus:
;               L1BPT (10,200,-1) will cause an error.  (The correct effect
;               is generated with:  L1BPT (10,200).)

;       Also, note that in those macros that generate global byte-pointers,
;       section values and address values must always be specified as distinct
;       arguments.  If address symbol FOO resolves to 377,,123456 , then it
;       would be specified in the macros as follows:
;               G2BPT(FOO_-^D18,7,FOO&777777,36)
;       Or (better):
;               FOOSEC=FOO_-^D18
;               FOOADR=FOO&777777
;               G2BPT(FOOSEC,7,FOOADR,36)

;       If runtime-generated values are needed, then any or all argument
;       fields may be assembled as zero and filled in at runtime using an
;       appropriate DPB instruction.  (G1BPT will not allow a zero bytesize
;       and will only allow a zero byte position if it is legal for that
;       particular bytesize.)
```

```
;       1-word local byte pointer
;       ================================================================
;       !  P  !  S  ! 0 ! I !  X  !             ADDR                   !
;       ================================================================
;       !0    5!6   11! 12! 13!14  17!18                             35!


        ;Macro to generate local, 1-word byte pointers
        ;Args:
        ;       BSIZ    Byte size
        ;       ADDR    18-bit address (indexing or indirection
        ;               may be specified)
        ;       BPOS    Optional byte position

        ;Generates Q errors on the following:
        ;               Bits 0-12 non-zero in ADDR
        ;               BSIZ or BPOS greater than 6 bits

        DEFINE L1BPT (BSIZ,ADDR,BPOS)
        <.BSIZ.=BSIZ            ;;Convert args to numeric
        .BPOS.=BPOS
        ..ERR.=0                ;;Reset error flag
        IFN <<ADDR>&<^O<777740,,0>>>,
          <MPRNTX(L1BPT,Bits 0 - 12 non-zero in address field: ADDR)
          ..ERR.=1
          >
        IFN <.BSIZ.&<^O<-1,,777700>>>,
          <MPRNTX(L1BPT,Bytesize greater than 6 bits: BSIZ)
          ..ERR.=1
          >
        IFN <.BPOS.&<^O<-1,,777700>>>,
          <MPRNTX(L1BPT,Byte offset greater than 6 bits: BPOS)
          ..ERR.=1
          >
        ;;Cause Q error
        IFN <..ERR.>,<-1,-1,-1>
        ;;Generate byte pointer
        IFE <..ERR.>,
          <IFIDN <BPOS><>,<POINT .BSIZ.,ADDR>
          IFDIF <BPOS><>,<POINT .BSIZ.,ADDR,.BPOS.>
          >
        PURGE ..ERR.,.BSIZ.,.BPOS.
        >
```

```
;       1-word global byte pointer
;       ================================================================
;       !     P,S     !        SEC         !              ADDR           !
;       ================================================================
;       !0          5!6                17!                            35!


        ;Macro to generate global, 1-word byte pointers
        ;Args:
        ;
        ;       SEC     12-bit section address
        ;       BSIZ    Byte size
        ;       ADDR    18-bit address (NO!! indexing or indirection
        ;               may be specified)
        ;       BPOS    Optional byte position

        ;Generates Q errors on following:
        ;               Illegal byte size or byte position
        ;               Indirection or indexing specified with ADDR
        ;               ADDR greater than 18 bits
        ;               SEC greater than 12 bits

        ;Legal sizes and positions are as follows:

        ;Size           Positions (Octal)
        ;6              44,36,30,22,14,6,0
        ;7              44,35,26,17,10,1
        ;8              44,34,24,14,4
        ;9              44,33,22,11,0
        ;18             44,22,0
```

```
DEFINE G1BPT (SEC,BSIZ,ADDR,BPOS<^O44>)
<..ERR.=0                       ;;Reset error flag
..ENC.=0                        ;;Define it as an arbitrary value
IFE BSIZ-7,                     ;;IF BYTESIZE=7
  <..ENC.=^O57                  ;;Get correct encode value
  IFN BPOS-^O44,<IFN BPOS-^O35,<IFN BPOS-^O26,<IFN BPOS-^O17,<IFN BPOS-^O10,
              <IFN BPOS-1<..ERR.=..ERR.!1>>>>>>
  >
IFN BSIZ-7,                     ;;ELSE IF BYTESIZE=6
  <IFE BSIZ-6,
    <..ENC.=^O45                        ;;Get correct encode value
    IFN BPOS-^O44,<IFN BPOS-^O36,<IFN BPOS-^O30,<IFN BPOS-^O22,<IFN BPOS-^O14,
              <IFN BPOS-6,<IFN BPOS,<..ERR.=.ERR.!1>>>>>>>
    >
  IFN BSIZ-6,                   ;;ELSE IF BYTESIZE=8
    <IFE BSIZ-^O10,
      <..ENC.=^O50              ;;Get correct encode value
      IFN BPOS-^O44,<IFN BPOS-^O34,<IFN BPOS-^O24,<IFN BPOS-^O14,<IFN BPOS-4,
              <..ERR.=.ERR.!1>>>>>
      >
    IFN BSIZ-^O10,              ;;ELSE IF BYTESIZE=9
      <IFE BSIZ-^O11,
        <..ENC.=^O62            ;;Get correct encode value
        IFN BPOS-^O44,<IFN BPOS-^O33,<IFN BPOS-^O22,<IFN BPOS-^O11,<IFN BPOS,
              <..ERR.=..ERR.!1>>>>>
        >
      IFN BSIZ-^O11,            ;;ELSE IF BYTESIZE=18
        <IFE BSIZ-^O22,
          <..ENC.=^O54          ;;Get correct encode value
          IFN BSIZ-^O44,<IFN BSIZ-^O22<IFN BSIZ,<..ERR.=..ERR.!1>>>
          >
        IFN BSIZ-^O22,<..ERR.=..ERR.!2> ;;ELSE: Illegal byte size
        >
      >
    >
  >
IFN <<ADDR>&<-1,,0>>,<..ERR.=..ERR.!4> ;;Address greater than 18 bits
                                        ;; or indexing or indirection specified
IFN <<SEC>&<^O<-1,,770000>>>,<..ERR.=..ERR.!<^O20>> ;;Section greater than 12 bits
IFN ..ERR.,
  <IFN ..ERR.&1,<MPRNTX (G1BPT,Illegal byte offset: BPOS)>
  IFN ..ERR.&2,<MPRNTX (G1BPT,Illegal byte size: BSIZ)>
  IFN ..ERR.&4,
    <MPRNTX (G1BPT,<Address indexed, indirect, or greater than 18 bits: ADDR>)>
  IFN ..ERR.&<^O20>,<MPRNTX (G1BPT,Section greater than 12 bits: SEC)>
  -1,-1,-1              ;;Cause a "Q" error
  >
;;Generate the byte pointer
IFE ..ERR.,
  <<<..ENC.+<<BSIZ>-<<BPOS>/<BSIZ>>>>_<^O36>>!<<ADDR>&<0,,-1>>!<<SEC>_^O<22>>>
PURGE ..ERR.,..ENC.
>
```

```
;       2-word local byte pointer

;       !0     5!6    11! 12! 13      17!18                          35!
;       =================================================================
;       !  P   !  S   ! 1 ! Reserved  !       Available to User        !
;       =================================================================
;       !1!0!    Reserved   ! I !   X   !            ADDR               !
;       =================================================================
;       !0!1!2           12! 13!14   17!18                          35!


        ;Macro to generate local, 2-word byte pointers
        ;Args:
        ;
        ;       BSIZ    Byte size
        ;       ADDR    18-bit address (Indexing or indirection
        ;               may be specified)
        ;       BPOS    Optional byte position
        ;       OPT     Optional user field available in word 1, right half

        ;Generates Q errors on the following:
        ;               Bits 0-12 non-zero in ADDR
        ;               Bits 0-17 non-zero in OPT
        ;               BSIZ or BPOS greater than 6 bits

        DEFINE L2BPT(BSIZ,ADDR,BPOS,OPT<0>)
        <..ERR.=0               ;;Reset error flag
          .BSIZ.=BSIZ           ;;Convert args to numeric
          .BPOS.=BPOS
        IFN <<ADDR>&<^O<777740,,0>>>,
          <MPRNTX(L2BPT,Bits 0 - 12 non-zero in address field: ADDR)
          ..ERR.=1
          >
        IFN <<OPT>&<-1,,0>>,
          <MPRNTX(L2BPT,Bits 0-17 non-zero in optional field: OPT)
          ..ERR.=1
          >
        IFN <.BSIZ.&<^O<-1,,777700>>>,
          <MPRNTX(L2BPT,Bytesize greater than 6 bits: BSIZ)
          ..ERR.=1
          >
        IFN <.BPOS.&<^O<-1,,777700>>>,
          <MPRNTX(L2BPT,Byte offset greater than 6 bits: BPOS)
          ..ERR.=1
          >
        IFN ..ERR.,<-1,-1,-1>           ;;Generate Q error
        ;;Generate the byte pointer
        IFE ..ERR.,
          <IFDIF <BPOS><>,<<<POINT .BSIZ.,OPT,.BPOS.>!1B12>&<^O<777740,,-1>>>
          IFIDN <BPOS><>,<<<POINT .BSIZ.,OPT>!1B12>&<^O<777740,,-1>>>
            <1B0!<<^O<400037,,-1>>&<ADDR>>>     ;;Generate LFIW
          >
        PURGE ..ERR.,.BSIZ.,.BPOS.
        >
```

```
;       2-word global byte pointer

;       !0      5!6     11! 12! 13      17!18                          35!
;       =================================================================
;       !   P   !   S   ! 1 ! Reserved  !       Available to User        !
;       =================================================================
;       !0! I !   X   !       SEC        !              ADDR              !
;       =================================================================
;       !0! 1 !2     5!6                17!                            35!


        ;Macro to generate global, 2-word byte pointers
        ;Args:
        ;       SEC     12-bit section address
        ;       BSIZ    Byte size
        ;       ADDR    18-bit address (Indexing or indirection
        ;               may be specified)
        ;       BPOS    Optional byte position
        ;       OPT     Optional user field available in word 1, right half

        ;Generates Q errors on the following:
        ;               SEC greater than 12 bits
        ;               Bits 0-12 non-zero in ADDR
        ;               Bits 0-17 non-zero in OPT
        ;               BSIZ or BPOS greater than 6 bits

        DEFINE G2BPT(SEC,BSIZ,ADDR,BPOS,OPT<0>)
        <..ERR.=0               ;;Reset error flag
          .BSIZ.=BSIZ           ;;Convert args to numeric
          .BPOS.=BPOS
        IFN <<SEC>&<^O<-1,,770000>>>,
          <MPRNTX(G2BPT,Section greater than 12 bits: SEC)
          ..ERR.=1
          >
        IFN <<ADDR>&<^O<777740,,0>>>,
          <MPRNTX(G2BPT,Bits 0 - 12 non-zero in address field: ADDR)
          ..ERR.=1
          >
        IFN <<OPT>&<-1,,0>>,
          <MPRNTX(G2BPT,Bits 0-17 non-zero in optional field: OPT)
          ..ERR.=1
          >
        IFN <.BSIZ.&<^O<-1,,777700>>>,
          <MPRNTX(G2BPT,Bytesize greater than 6 bits: BSIZ)
          ..ERR.=1
          >
        IFN <.BPOS.&<^O<-1,,777700>>>,
          <MPRNTX(G2BPT,Byte offset greater than 6 bits: BPOS)
          ..ERR.=1
          >
        IFN ..ERR.,<-1,-1,-1>           ;;Generate Q error
        ;;Generate the byte pointer
        IFE ..ERR.,
          <IFDIF <BPOS><>,<<<POINT .BSIZ.,OPT,.BPOS.>!1B12>&<^O<777740,,-1>>>
          IFIDN <BPOS><>,<<<POINT .BSIZ.,OPT>!1B12>&<^O<777740,,-1>>>
            ;;Generate GFIW
            <<<ADDR>_<^O14>>&<^O<370000,,0>>!<<ADDR>&<0,,-1>>!<<SEC>_<^O22>>>
          >
        PURGE ..ERR.,.BSIZ.,.BPOS.
        >
```

```
LIT                                  ;MAKE SURE LITERALS COME BEFORE END MARK
   IFN REL,<
.RLEND==:.-1                         ;MARK END OF CODE IN MACREL
   >
  IF2,<PURGE REL>                    ;FLUSH REL FROM UNIV FILE

        .XCMSY

        END             ;End of MACSYM
```

APPENDIX D

ACTSYM

```
;<5.UTILITIES>ACTSYM.MAC.2, 28-Oct-81 14:37:54, EDIT BY GRANT
;Change major version to 5


UNIVERSAL ACTSYM - DECsystem-10/20 symbol file for accounting V2(35)
SUBTTL B.A. HUIZENGA/BAH/TAH/DPM    22-Jul-81


;VERSION INFORMATION
        ACCVER==5                       ;MAJOR VERSION
        ACCEDT==35                      ;EDIT LEVEL
        ACCMIN==0                       ;MINOR VERSION
        ACCWHO==0                       ;LAST MODIFIER


;COPYRIGHT (C) 1979,1980,1981 BY
;DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
;
;THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
;ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH LICENSE AND WITH THE
;INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY  OTHER
;COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
;OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE  SOFTWARE  IS  HEREBY
;TRANSFERRED.
;
;THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT  NOTICE
;AND  SHOULD  NOT  BE  CONSTRUED  AS A COMMITMENT BY DIGITAL EQUIPMENT
;CORPORATION.
;
;DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY  OF  ITS
;SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.


IFNDEF  FTUUOS,<FTUUOS==0>              ;TOPS-10
IFNDEF  FTJSYS,<FTJSYS==-1>             ;TOPS-20

DEFINE  TOPS10,<IFN FTUUOS,>
DEFINE  TOPS20,<IFN FTJSYS,>

IF1,<
        TOPS10  <PRINTX [Assembling ACTSYM-10]>
        TOPS20  <PRINTX [Assembling ACTSYM-20]>
>
```

```
SUBTTL  PARAMETERS FOR USAGE ITEM DESCRIPTORS

;FIELDS IN DATA ITEM DESCRIPTOR

US%FLG==:77B5                   ;FLAGS
    US%IMM==:1B0                ; 1 - IMMEDIATE DATA ITEM
                                ; 0 - ADDRESS OF DATA ITEM
US%TYP==:77B11                  ;TYPE CODE
    .USASC==:0                  ;ASCII
    .USSIX==:1                  ;SIXBIT
    .USOCT==:2                  ;OCTAL
    .USDEC==:3                  ;DECIMAL
    .USDAT==:4                  ;DATE-TIME
    .USTAB==:5                  ;TABLE (SPECIAL FORM)
    .USVER==:6                  ;VERSION NUMBER
    .USSPC==:7                  ;SPACE FILL
    .USPDT==:10                 ;OLD STYLE TOPS-10 DATE/TIME

US%LEN==:777B20                 ;LENGTH
US%COD==:77777B35               ;ITEM CODE

;RECORD TYPE CODES

        RADIX 10                ;**** NOTE RADIX 10 ****

.UTRST==:1                      ;SYSTEM RESTART ENTRY
.UTSEN==:2                      ;SESSION ENTRY
.UTCKP==:3                      ;CHECKPOINT ENTRY (SYSTEM RESTART)
.UTUSB==:4                      ;FIRST ENTRY OF USAGE FILE (SAME AS .UTRST)
.UTTAD==:5                      ;DATE-TIME CHANGE
.UTBAT==:6                      ;BATCH PROCESSOR
.UTINP==:7                      ;INPUT SPOOLER ENTRY
.UTOUT==:8                      ;OUTPUT SPOOLER ENTRY
.UTFLU==:9                      ;FILE USAGE DIRECTORY ENTRY
.UTDSU==:10                     ;DISK SPINDLE USAGE ENTRY
.UTMNT==:11                     ;STRUCTURE MOUNT ENTRY
.UTMMT==:12                     ;TAPE MOUNT ENTRY
.UTDMT==:13                     ;DECtape MOUNT ENTRY
.UTFCM==:14                     ;FILE COMMAND ENTRY
.UTRET==:15                     ; File retrieved
.UTARC==:16                     ; File archived
.UTMIG==:17                     ; File migrated
.UTCOL==:18                     ; File collected

.UTUSR==:5000                   ;USER-DEFINED ENTRY TYPES ARE 5000-9999

        RADIX 8                 ;**** END OF RADIX 10 ****
```

COMMENT

The format of the data to be passed to the accounting system will consist of a list of items describing the entries in a single record.

The record descriptor list will have a header containing the record type code and the record version information.

Format of a record descriptor:

```
!=========================================================!
!  DEC ver.    !  CUST ver.  !          Entry Type         !
!---------------------------------------------------------!
! Flags   !  Type  !   Length     !        Item Code       !
!---------------------------------------------------------!
!            Data or Address (-1 for default)              !
!---------------------------------------------------------!
\                             .                            \
\                             .                            \
\                             .                            \
!---------------------------------------------------------!
!                   0 (Marks end of list)                  !
!=========================================================!
```

The generation of these tables will be controlled by the UITEM. macro. All known data items will have a name generated by the use of this macro. If any application dependent items are needed the UITEM. macro may be used to generate the new item. The USENT. macro may be used to generate the first word of the entry descriptor table.

All USAGE entry headers and the system-defined USAGE entry types use the specific item types and these items are ordered by the system. Installation-defined USAGE entries (with entry types above .UTUSR - 5000 to 9999) use the arbitrary data items (USUAS., USUSX., USUDC., USUOC., USUVR., USUDT., and USUSP.) in the order in which they are to be written into the USAGE entry record. Each arbitrary data record must be preceded by a USUAR. item.

Example of installation-defined USAGE entry:

```
;The following code writes a USAGE entry for a fictitious "file access count"
; in a user program.  This program must be running as an enabled OPERATOR or
; WHEEL.

;Here to write USAGE entry for file access count

        MOVEI T1,.USENT         ;USAGE function to write entry
        MOVEI T2,FILRDB         ;Address of Record Descriptor Block
        USAGE                   ;Write the entry
         ERJMP USGERR           ;Failed to write entry-- do something else
        JRST USGOK              ;Entry written-- go on

;Record descriptor block for file access count accounting

FILRDB:
        USENT. (.UTUSR+12,1,1)  ;Entry type 5012= file access count.
        USPVR. (<BYTE(3)VWHO(9)VMAJOR(6)VMINOR(18)VEDIT>,US%IMM) ;Version
                                ;  of this program (for header record).

        USUAR.                  ;Start of first arbitrary record.
        USUAS. ([ASCII \This appears in every entry\],,27) ;Text.
        USUSP. (,,5)            ;Space fill, 5 characters.
```

```
        USUSP. (,,5)            ;Space fill, 5 characters.
        USUDC. (FILCNT,,6)      ;Count of file accesses, 000000-999999.

        USUAR.                  ;Start of second arbitrary record.
        USUSX. (<SIXBIT \FILE: \>,US%IMM,6) ;SIXBIT text for filename.
        USUAS. (FILNAM,,200)    ;File name, 200 characters.

        EXP 0                   ;End of entry.

;Storage

FILCNT: BLOCK 1                 ;File access count
FILNAM: BLOCK ^D<200/5>         ;File name text

&       ;;; End of comment
```

```
SUBTTL UITEM. / USENT. / USAGE. DEFINITIONS

        SALL

DEFINE UITEM. (NAME,TYPE,LEN) <
    DEFINE US'NAME'. (DATA<-1>,IMMED<0>,ULEN<LEN>) <
        USAGE. (.US'NAME,ULEN,TYPE,IMMED,<DATA>)
    >
>

DEFINE USENT. (ETYPE,DVER,CVER) <
        BYTE (9) ^D<DVER>,^D<CVER> (18) ^D<ETYPE>
>

DEFINE USAGE. (CODE,LENGTH,TYPE,FLAGS,DATUM) <
        FLAGS+<TYPE>B11+<^D<LENGTH>>B20+CODE
    IFB  <DATUM>,<-1>
    IFNB <DATUM>,<DATUM>
>

DEFINE USDSK. (TABLE) <
        USAGE. (.USDST,0,.USTAB,US%IMM,<TABLE>)
>
```

```
SUBTTL USAGE. ITEM-CODE DEFINITIONS

DEFINE USLIST <

DEFUS (JNO,0,.USDEC,4)          ;JOB NUMBER
DEFUS (TAD,1,.USDAT,14)         ;CURRENT DATE/TIME
DEFUS (TRM,2,.USASC,1)          ;TERMINAL DESIGNATOR
DEFUS (LNO,3,.USOCT,4)          ;LINE NUMBER
DEFUS (PNM,4,.USSIX,6)          ;PROGRAM NAME (CALLER)
DEFUS (PVR,5,.USVER,15)         ;PROGRAM VERSION
DEFUS (AMV,6,.USVER,15)         ;ACCOUNTING MODULE VERSION
DEFUS (NOD,7,.USSIX,6)          ;CALLER'S LOCATION
DEFUS (PPN,10,.USOCT,12)        ;PROJECT / PROGRAMMER NUMBER (TOPS10 ONLY)
DEFUS (NM1,11,.USSIX,6)         ;NAME OF USER (TOPS10)
DEFUS (SNM,12,.USASC,39)        ;SYSTEM NAME
DEFUS (MVR,13,.USVER,15)        ;MONITOR VERSION NUMBER
DEFUS (MBD,14,.USDAT,14)        ;MONITOR BUILD DATE
DEFUS (MUP,15,.USDEC,18)        ;MONITOR UPTIME (IN SECONDS)
DEFUS (ACT,16,.USASC,39)        ;ACCOUNT STRING
DEFUS (LCK,17,.USDAT,14)        ;TIME OF LAST CHECKPOINT
DEFUS (RTM,20,.USDEC,9)         ;RUNTIME IN MS
DEFUS (CTI,21,.USDEC,11)        ;CORE-TIME INTEGRAL (TOPS10 ONLY)
DEFUS (SST,22,.USDAT,14)        ;SESSION START TIME
DEFUS (JTY,23,.USDEC,1)         ;JOB TYPE (BATCH / TIMESHARING)
DEFUS (BJN,24,.USSIX,6)         ;BATCH JOB NAME
DEFUS (BSN,25,.USDEC,6)         ;BATCH SEQUENCE NUMBER
DEFUS (COM,26,.USASC,39)        ;USER COMMENT
DEFUS (DKR,27,.USDEC,8)         ;DISK READS
DEFUS (DKW,30,.USDEC,8)         ;DISK WRITES
DEFUS (VTI,31,.USDEC,11)        ;VIRTUAL CORE-TIME INTEGRAL
DEFUS (EBX,32,.USDEC,9)         ;EBOX MEGACOUNTS (CYCLES * 10^6)
DEFUS (MBX,33,.USDEC,9)         ;MBOX MEGACOUNTS (CYCLES * 10^6)
DEFUS (MCL,34,.USDEC,6)         ;MONITOR CALLS
DEFUS (MCM,35,.USDEC,6)         ;MONITOR COMMANDS
DEFUS (SCL,36,.USDEC,3)         ;SCHEDULING CLASS
DEFUS (TYI,37,.USDEC,6)         ;TTY INPUT CHARACTERS
DEFUS (TYO,40,.USDEC,6)         ;TTY OUTPUT CHARACTERS
DEFUS (TYW,41,.USDEC,6)         ;TTY WAKEUPS
DEFUS (CPN,42,.USDEC,1)         ;NUMBER OF CPUS
DEFUS (CP0,43,.USDEC,4)         ;SERIAL NUMBER OF CPU0
DEFUS (CP1,44,.USDEC,4)         ;SERIAL NUMBER OF CPU1
DEFUS (CP2,45,.USDEC,4)         ;SERIAL NUMBER OF CPU2
DEFUS (CP3,46,.USDEC,4)         ;SERIAL NUMBER OF CPU3
DEFUS (CP4,47,.USDEC,4)         ;SERIAL NUMBER OF CPU4
DEFUS (CP5,50,.USDEC,4)         ;SERIAL NUMBER OF CPU5
DEFUS (RQQ,51,.USDEC,11)        ;RUN QUEUE QUOTIENT (TOPS10 ONLY)
DEFUS (NM2,52,.USASC,39)        ;NAME OF USER (TOPS20)
DEFUS (CCT,53,.USDEC,7)         ;CONSOLE CONNECT TIME (SECONDS)
DEFUS (DTL,54,.USDAT,14)        ;DATE/TIME BEFORE CHANGE (STAD)
```

```
;DISK UTILIZATION RECORD ENTRIES

DEFUS (NRF,55,.USDEC,3)         ;NUMBER OF RECORDS FOLLOWING
DEFUS (TAL,56,.USDEC,10)        ;TOTAL ALLOCATED STORAGE
DEFUS (TUS,57,.USDEC,10)        ;TOTAL STORAGE USED
DEFUS (TNF,60,.USDEC,5)         ;TOTAL NUMBER OF FILES
DEFUS (STR,61,.USASC,6)         ;STRUCTURE NAME
DEFUS (STP,62,.USDEC,1)         ;STRUCTURE TYPE CODE
DEFUS (KTP,63,.USDEC,3)         ;CONTROLLER TYPE
DEFUS (DTP,64,.USDEC,3)         ;DEVICE TYPE
DEFUS (LIQ,65,.USDEC,6)         ;LOGGED IN QUOTA
DEFUS (LOQ,66,.USDEC,6)         ;LOGGED OUT QUOTA
DEFUS (LLI,67,.USDAT,14)        ;LAST LOGGED IN DATE/TIME
DEFUS (LAT,70,.USDAT,14)        ;LAST DISK ACCOUNTING DATE/TIME
DEFUS (EXP,71,.USASC,1)         ;EXPIRED DIRECTORY (Y/N)
DEFUS (DIR,72,.USASC,39)        ;DIRECTORY NAME
DEFUS (ALC,73,.USDEC,10)        ;ALLOCATED STORAGE
DEFUS (USG,74,.USDEC,10)        ;STORAGE USED
DEFUS (FIL,75,.USDEC,5)         ;NUMBER OF FILES
DEFUS (FON,76,.USASC,1)         ;FILES ONLY INDICATOR (Y/N)

;SPOOLER INFORMATION RECORD ENTRIES

DEFUS (SRT,77,.USDEC,9)         ;SPOOLER RUNTIME
DEFUS (SCI,100,.USDEC,11)       ;CORE-TIME INTEGRAL
DEFUS (SDR,101,.USDEC,8)        ;SPOOLER DISK READS
DEFUS (SDW,102,.USDEC,8)        ;SPOOLER DISK WRITES
DEFUS (JNM,103,.USSIX,6)        ;JOB NAME
DEFUS (QNM,104,.USSIX,3)        ;QUEUE NAME
DEFUS (SDV,105,.USSIX,6)        ;PROCESSING DEVICE
DEFUS (SSN,106,.USDEC,6)        ;SEQUENCE NUMBER
DEFUS (SUN,107,.USDEC,6)        ;SPOOLER UNITS PROCESSED
DEFUS (CRT,110,.USDAT,14)       ;CREATION DATE/TIME OF REQUEST
DEFUS (DSP,111,.USSIX,6)        ;DISPOSITION
DEFUS (TXT,112,.USASC,39)       ;OPR OR SYSTEM TEXT
DEFUS (PRI,113,.USDEC,2)        ;PRIORITY
DEFUS (SNF,114,.USDEC,5)        ;NUMBER OF FILES PROCESSED
DEFUS (SCD,115,.USDAT,14)       ;SCHEDULED DATE/TIME
DEFUS (FRM,116,.USSIX,6)        ;FORMS TYPE

;DATE/TIME CHANGE RECORD ENTIRES

DEFUS (OFD,117,.USDEC,7)        ;OFFSET IN DAYS
DEFUS (OFS,120,.USDEC,7)        ;OFFSET IN SECONDS
DEFUS (ODT,121,.USDAT,14)       ;OLD DATE/TIME

;ARBITRARY RECORD ITEM TYPES

DEFUS (UAR,122,.USSPC,0)        ;USER-DEFINED ARBITRARY RECORD DELIMITER
DEFUS (UAS,123,.USASC,0)        ;USER-DEFINED ASCII STRING
DEFUS (USX,124,.USSIX,0)        ;USER-DEFINED SIXBIT STRING
DEFUS (UOC,125,.USOCT,0)        ;USER-DEFINED OCTAL NUMBER
DEFUS (UDC,126,.USDEC,0)        ;USER-DEFINED DECIMAL NUMBER
DEFUS (UDT,127,.USDAT,14)       ;USER-DEFINED DATE AND TIME
DEFUS (UVR,130,.USVER,15)       ;USER-DEFINED VERSION (STANDARD FORMAT)
DEFUS (USP,131,.USSPC,0)        ;USER-DEFINED SPACE FILL

;STRUCTURE MOUNT RECORD ENTRIES

DEFUS (SSI,132,.USSIX,6)        ;SIXBIT STRUCTURE ID
DEFUS (TNP,133,.USDEC,2)        ;TOTAL NUMBER OF PACKS
DEFUS (SRV,134,.USDAT,14)       ;SERVICED DATE/TIME OF REQUEST
```

```
DEFUS (MCT,135,.USDEC,3)        ;MOUNT COUNT BEFORE MOUNT
DEFUS (DCT,136,.USDEC,3)        ;MOUNT COUNT AFTER DISMOUNT
DEFUS (ATP,137,.USDEC,1)        ;ACCESS TYPE

;TAPE MOUNT RECORD ENTRIES

DEFUS (VID,140,.USSIX,6)        ;MAGTAPE VOLUME LABEL IN VOL1 LABEL
DEFUS (VSN,141,.USSIX,6)        ;VISUAL SERIAL NUMBER
DEFUS (MRF,142,.USDEC,8)        ;THOUSANDS OF FRAMES READ
DEFUS (MWF,143,.USDEC,8)        ;THOUSANDS OF FRAMES WRITTEN
DEFUS (MLT,144,.USDEC,2)        ;LABEL TYPE
DEFUS (MLS,145,.USDEC,1)        ;VOLUME LABEL STATE
DEFUS (MRD,146,.USDEC,8)        ;NUMBER OF PHYSICAL RECORDS READ
DEFUS (MWR,147,.USDEC,8)        ;NUMBER OF PHYSICAL RECORDS WRITTEN
DEFUS (FSI,150,.USSIX,6)        ;FILE SET IDENTIFIER
DEFUS (SRE,151,.USDEC,10)       ;NUMBER OF SOFT READ ERRORS
DEFUS (SWE,152,.USDEC,10)       ;NUMBER OF SOFT WRITE ERRORS
DEFUS (HRE,153,.USDEC,10)       ;NUMBER OF HARD READ ERRORS
DEFUS (HWE,154,.USDEC,10)       ;NUMBER OF HARD WRITE ERRORS
```

```
; Retrieve/archive/migration/collection entries

DEFUS (TP1,155,.USSIX,6)        ; Tape ID 1
DEFUS (TS1,156,.USDEC,4)        ; Tape saveset #
DEFUS (TF1,157,.USDEC,6)        ; Tape file #
DEFUS (TP2,160,.USSIX,6)        ; Tape 2 ID
DEFUS (TS2,161,.USDEC,4)        ; Tape saveset #
DEFUS (TF2,162,.USDEC,6)        ; Tape file #
DEFUS (RSN,163,.USOCT,1)        ; Reason offline code
DEFUS (EUT,164,.USDEC,7)        ; ELAPSED USE TIME (STRUCTURE AND TAPE)

;BATCH PROCESSOR RECORD ENTRIES

DEFUS (BAC,165,.USASC,39)       ;BATCH ACCOUNT STRING
DEFUS (BRN,166,.USDEC,9)        ;BATCH RUNTIME
DEFUS (BCT,167,.USDEC,11)       ;BATCH CORE-TIME INTEGRAL
DEFUS (BDR,170,.USDEC,8)        ;BATCH DISK READS
DEFUS (BDW,171,.USDEC,8)        ;BATCH DISK WRITES
DEFUS (BJB,172,.USSIX,6)        ;JOB NAME
DEFUS (BSQ,173,.USDEC,6)        ;SEQUENCE NUMBER
DEFUS (BDT,174,.USDAT,14)       ;CREATION DATE/TIME OF REQUEST
DEFUS (BET,175,.USDAT,14)       ;DATE/TIME JOB COULD BE SCHEDULED
DEFUS (BST,176,.USDAT,14)       ;DATE/TIME JOB STARTED RUNNING
DEFUS (BDS,177,.USSIX,6)        ;DISPOSTITION
DEFUS (BTX,200,.USASC,39)       ;TEXT
DEFUS (BPR,201,.USDEC,2)        ;PRIORITY OF REQUEST
DEFUS (URE,202,.USDEC,6)        ;USER'S RUNTIME ESTIMATE
DEFUS (UAC,203,.USDEC,6)        ;USER'S ACTUAL RUNTIME
DEFUS (UCE,204,.USDEC,4)        ;USER'S CORE ESTIMATE
DEFUS (UCH,205,.USDEC,4)        ;USER'S CORE HIGHWATER MARK
DEFUS (RIN,206,.USDEC,6)        ;REQUEST ID NUMBER

;INPUT SPOOLER RECORD ENTRIES

DEFUS (IAC,207,.USASC,39)       ;INPUT SPOOLER ACCOUNT STRING
DEFUS (IRN,210,.USDEC,9)        ;INPUT SPOOLER RUNTIME
DEFUS (ICT,211,.USDEC,11)       ;INPUT SPOOLER CORE-TIME INTERAL
DEFUS (IDR,212,.USDEC,8)        ;INPUT SPOOLER DISK READS
DEFUS (IDW,213,.USDEC,8)        ;INPUT SPOOLER DISK WRITES
DEFUS (IJN,214,.USSIX,6)        ;INPUT SPOOLER JOB NAME
DEFUS (IQN,215,.USSIX,3)        ;INPUT SPOOLER QUEUE NAME
DEFUS (IPD,216,.USSIX,6)        ;INPUT SPOOLER PROCESSING DEVICE
DEFUS (ISN,217,.USDEC,6)        ;INPUT SPOOLER SEQUENCE NUMBER
DEFUS (ICR,220,.USDEC,6)        ;INPUT SPOOLER NUMBER OF CARDS READ
DEFUS (ICD,221,.USDAT,14)       ;CREATION DATE/TIME OF REQUEST
DEFUS (IDS,222,.USSIX,6)        ;DISPOSITION
DEFUS (ITX,223,.USASC,39)       ;TEXT
DEFUS (IPR,224,.USDEC,2)        ;PRIORITY OF REQUEST
DEFUS (IRI,225,.USDEC,6)        ;REQUEST ID NUMBER
DEFUS (ICN,226,.USDEC,7)        ;CONNECT TIME

DEFUS (OAC,227,.USASC,39)       ;OUTPUT SPOOLER ACCOUNT STRING
DEFUS (ORI,230,.USDEC,6)        ;OUTPUT SPOOLER REQUEST ID NUMBER
DEFUS (OCN,231,.USDEC,7)        ;OUTPUT SPOOLER CONNECT TIME
DEFUS (DPN,232,.USASC,39)       ;DISK USAGE DIRECTORY PPN

;DISK USAGE ACCOUNT STRING RECORD - (DUA)

DEFUS (DAC,233,.USASC,39)       ;DUA - ACCOUNT STRING
DEFUS (DPP,232,.USASC,39)       ;DUA - PPN/DIRECTORY
DEFUS (DFN,233,.USDEC,5)        ;DUA - NUMBER OF FILES
DEFUS (DFS,234,.USSIX,6)        ;DUA - FILE STRUCTURE NAME
```

```
DEFUS (DFT,235,.USDEC,1)          ;DUA - FILE STRUCTURE TYPE
DEFUS (DKT,236,.USDEC,3)          ;DUA - CONTROLLER TYPE
DEFUS (DDT,237,.USDEC,3)          ;DUA - DEVICE TYPE

;DISK SPINDLE USAGE RECORD

DEFUS (SFS,240,.USSIX,6)          ;FILE STRUCTURE NAME
DEFUS (SFT,241,.USDEC,1)          ;TYPE OF FILE STRUCTURE
DEFUS (SCT,242,.USDEC,3)          ;CONTROLLER TYPE
DEFUS (SDT,243,.USDEC,3)          ;DEVICE TYPE
DEFUS (SPI,244,.USASC,12)         ;DISK PACK IDENTIFIER
DEFUS (SDU,245,.USSIX,4)          ;DISK UNIT NAME
DEFUS (SNP,246,.USDEC,2)          ;TOTAL NUMBER OF PACKS IN STRUCTURE
DEFUS (SMN,247,.USDEC,2)          ;M OF N PACK COUNT
DEFUS (DTF,250,.USDAT,14)         ;DATE/TIME OF FIRST MOUNT
DEFUS (DCC,251,.USDEC,7)          ;CONNECT TIME

;USER FILE STRUCTURE MOUNT RECORD (CONT.)

DEFUS (FMA,252,.USASC,39)         ;ACCOUNT STRING
DEFUS (FST,254,.USDEC,1)          ;TYPE OF FILE STRUCTURE
DEFUS (FCT,255,.USDEC,3)          ;CONTROLLER TYPE
DEFUS (FDT,256,.USDEC,3)          ;DEVICE TYPE
DEFUS (FDS,257,.USSIX,6)          ;DISPOSITION
DEFUS (FOT,260,.USASC,39)         ;TEXT
DEFUS (FCD,261,.USDAT,14)         ;CREATION DATE/TIME OF REQUEST
DEFUS (FSD,262,.USDAT,14)         ;SCHEDULED DATE/TIME OF REQUEST
DEFUS (FCO,263,.USDEC,7)          ;CONNECT TIME

;USER MAGTAPE MOUNT RECORD

DEFUS (MAC,264,.USASC,39)         ;ACCOUNT STRING
DEFUS (MDS,265,.USSIX,6)          ;DISPOSITION
DEFUS (MTX,266,.USASC,39)         ;TEXT
DEFUS (MCD,267,.USDAT,14)         ;CREATION DATE/TIME OF REQUEST
DEFUS (MSD,270,.USDAT,14)         ;SCHEDULED DATE/TIME OF REQUEST
DEFUS (MVD,271,.USDAT,14)         ;SERVICED DATE/TIME OF REQUEST
DEFUS (MCO,272,.USDEC,3)          ;TYPE OF CONTROLLER
DEFUS (MCN,273,.USDEC,7)          ;CONNECT TIME

;USER DECTAPE MOUNT RECORD

DEFUS (DAN,274,.USASC,39)         ;ACCOUNT STRING
DEFUS (DVI,275,.USSIX,6)          ;DECTAPE VOLUME IDENTIFIER
DEFUS (DRI,276,.USSIX,6)          ;DECTAPE REEL IDENTIFIER
DEFUS (DTR,277,.USDEC,8)          ;DECTAPE READS
DEFUS (DTW,300,.USDEC,8)          ;DECTAPE WRITES
DEFUS (DDS,301,.USSIX,6)          ;DISPOSTITION
DEFUS (DTX,302,.USASC,39)         ;TEXT
DEFUS (DCE,303,.USDAT,14)         ;CREATION DATE/TIME OF REQUEST
DEFUS (DSQ,304,.USDAT,14)         ;SCHEDULED DATE/TIME OF REQUEST
DEFUS (DSS,305,.USDAT,14)         ;SERVICED DATE/TIME OF REQUEST
DEFUS (DCN,306,.USDEC,7)          ;CONNECT TIME

;USER DECTAPE FILE COMMAND RECORD

DEFUS (FAS,307,.USASC,39)         ;ACCOUNT STRING
DEFUS (FMR,310,.USDEC,9)          ;MOUNT RUNTIME TO PROCESS USER REQUEST
DEFUS (FCI,311,.USDEC,11)         ;MOUNT CORE-TIME INTEGRAL
DEFUS (FDR,312,.USDEC,8)          ;MOUNT DISK READS
DEFUS (FDW,313,.USDEC,8)          ;MOUNT DISK WRITES
DEFUS (FCM,314,.USASC,1)          ;TYPE OF FILE COMMAND
DEFUS (FNF,315,.USDEC,2)          ;NUMBER OF FILES TRANSFERRED
```

```
DEFUS (FDP,316,.USSIX,6)        ;DISPOSITION
DEFUS (FTX,317,.USASC,39)       ;TEXT
DEFUS (FCQ,320,.USDAT,14)       ;CREATION DATE/TIME OF REQUEST
DEFUS (FSH,321,.USDAT,14)       ;SCHEDULED DATE/TIME OF REQUEST
DEFUS (FVD,322,.USDAT,14)       ;SERVICED DATE/TIME OF REQUEST
DEFUS (FCE,323,.USDEC,7)        ;CONNECT TIME


;LATE COMERS

DEFUS (NM3,324,.USSIX,6)        ;TOPS-10 - 2ND HALF OF USER NAME
                                ; (#11 IS 1ST HALF)
DEFUS (B27,325,.USSPC,27)       ;SPACE FILL 27 CHARACTERS USED IN TOPS10 TO
                                ; USE INSTEAD OF PPN/DIRECTORY DEFUSES. THE
                                ; DEFUS PPN IS USED IN CONJUNCTION WITH
                                ; THIS ONE.
DEFUS (SID,326,.USSIX,6)        ;TOPS-10 - DISK PACK ID (USED INSTEAD OF #244)
DEFUS (B06,327,.USSPC,6)        ;SPACE FILL 6 CHARACTERS (USED WITH #326 ABOVE)
DEFUS (UPF,330,.USSIX,1)        ;TOPS-10 DISK USAGE - UFD WAS PROTECTED
DEFUS (FPF,331,.USSIX,1)        ;TOPS-10 DISK USAGE - SOME FILES WERE PROTECTEL
DEFUS (TMA,332,.USSIX,1)        ;TOPS-10 DISK USAGE - USER HAS TOO MANY AUNIQUE
                                ;ACCOUNT STRINGS IN DIRECTORY.  LIMIT IS DEFINE
                                ;IN IPCF MESSAGE FOR DISK USAGE FROM BACKUP
DEFUS (LLG,333,.USPDT,14)       ;TOPS-10 OLD FORMAT DATE/TIME OF LAST LOGIN
DEFUS (DVN,334,.USSIX,6)        ;TOPS-10 DEVICE NAME (MAG/DECTAPE MOUNTS)

>;;; END OF USLIST
```

```
SUBTTL  Macros to define all USAGE. item codes

TOPS10  <

DEFINE UITEM. (NAME,TYPE,LEN) <
        DEFINE US'NAME'. (DATA<-1>,IMMED<0>) <
                USAGE. (.US'NAME,TYPE,IMMED,<DATA>)
        >
>

DEFINE USAGE. (CODE,TYPE,FLAGS,DATUM) <
        IFN TYPE-.USASC,<LENGTH==1>
        IFE TYPE-.USASC,<LENGTH==10
        IFL FLAGS,<PRINTX ?ASCII DATA CANNOT BE IMMEDIATE>>
        IFB <DATUM>,<
        QA.IMM+<LENGTH>B17+CODE
        EXP     -1
        >
        IFNB <DATUM>,<
                IFL FLAGS,<
                QA.IMM+<LENGTH>B17+CODE
                DATUM>
                IFGE FLAGS,<IFIDN <DATUM><-1>,<
                IFE LENGTH-10,<PRINTX ?CANNOT BE DEFAULTED>
                QA.IMM+<LENGTH>B17+CODE
                DATUM>
                           IFDIF <DATUM><-1>,<
                <LENGTH>B17+CODE
                <DATUM>&<37,,777777>>
        >
>
>

DEFINE USENT. (ETYPE,DVER,CVER,LRESP,RESP) <
        FLAGS==0
        IFNB <LRESP>,<IFG LRESP,<FLAGS==QF.RSP>>
        FLAGS+.QUMAE
        0
        LRESP,,RESP
        QA.IMM+<1>B17+.QBAFN
        UGENT$
        QA.IMM+<1>B17+.QBAET
        ETYPE
>

> ;END OF TOPS-10 CONDITIONAL


DEFINE DEFUS (NAM,VAL,TYP,LEN) <
    IF1,<IFDEF .US'NAM,<
        PRINTX .US'NAM ALREADY DEFINED
    >>
        .US'NAM==:VAL
        UITEM. (NAM,TYP,LEN)
>
```

```
;EXPAND ALL DEFINITIONS

USLIST

;SPECIAL ITEM TYPE CODE DEFINITIONS

.USDSX==:7776           ;STRUCTURE/DIRECTORY INFO WORD (SPECIAL)
.USDST==:7777           ;DISK STATISTICS TABLE POINTER
```

```
SUBTTL  TOPS-10 IPCF message definitions and formats

TOPS10  < ;START OF A LONG CONDITIONAL

        FTCASECONVERT==:0          ;LOWER/UPPER CONVERSION SELECTION FEATURE TEST
                                   ; 0 = DON'T CONVERT
                                   ;-1 =CONVERT LOWER CASE LETTERS TO UPPER CASE

        PRJWPB==400                ;SET LOGICAL BLOCK SIZE IN PROJCT.SYS TO 2 DISK

        PRJWPB==<PRJWPB+177>&777600 ;ROUND UP TO NEXT FULL DISK BLOCK

;IPCF TYPES OF MESSAGES SENT TO THE ACCOUNTING DAEMON, ACTDAE.
;       THESE ARE THE ACCOUNTING SUBFUNCTION VALUES TO STORE IN .QBAFN
;       OF THE QUEUE. UUO.  SEE UUOSYM.MAC.

UGVAL$==:1                         ;VALIDATION MESSAGES
UGLGN$==:2                         ;LOGIN MESSAGES (USER IS LOGGING IN)
UGSES$==:3                         ;SESSION MESSAGES (USER TYPED A SESSION COMMAND)
UGATT$==:4                         ;ATTACH MESSAGES
UGSDT$==:5                         ;SET DATE/TIME EVENT FROM DAEMON
UGVAC$==:6                         ;RESPONSE TO ANY MESSAGE IF REQUESTED
                                   ; (??$ACK IS NON-ZERO)
UGENT$==:7                         ;MAKE AN ENTRY
UGEBC$==:10                        ;END A BILLING CLOSURE
UGUFC$==:11                        ;USAGE FILE CLOSURE
UGFDM$==:12                        ;USER FILE STRUCTURE MOUNT MESSAGE
UGFDD$==:13                        ;USER FILE STRUCTURE DISMOUNT MESSAGE
UGMGM$==:14                        ;USER MAGTAPE MOUNT MESSAGE
UGMGD$==:15                        ;USER MAGTAPE DISMOUNT MESSAGE
UGDTM$==:16                        ;USER DECTAPE MOUNT MESSAGE
UGDTD$==:17                        ;USER DECTAPE DISMOUNT MESSAGE
UGSPM$==:20                        ;DISK PACK SPINDLE SPIN-UP MESSAGE
UGSPD$==:21                        ;DISK PACK SPINDLE SPIN-DOWN MESSAGE
UGACK$==:22                        ;GENERAL ACK CODE
UGDUE$==:23                        ;DISK USAGE FROM BACKUP
UGACC$==:24                        ;ACCESS CONTROL
UGOUP$==:25                        ;OBTAIN USER PROFILE
                                   ;26 = UNUSED
UGLGO$==:27                        ;MONITOR LOGOUT MESSAGE (.IPCSL)

;SUCCESSFUL/UNSUCCESSFUL IPCF MESSAGE CODES

UGTRU$==:1                         ;SUCCESSFUL MESSAGE
UGFAL$==:2                         ;FAILURE MESSAGE
```

```
;FORMAT OF LOGIN MESSAGE (UGLGN$)

UL$TYP==:0                      ;TYPE OF MESSAGE
UL$ACK==:UL$TYP+1               ;UNIQUE MESSAGE IDENTIFIER
UL$LIN==:UL$ACK+1               ;LINE NUMBER
UL$PRG==:UL$LIN+1               ;PROGRAM NAME (ALWAYS LOGIN)
UL$VER==:UL$PRG+1               ;VERSION OF LOGIN
UL$NOD==:UL$VER+1               ;NODE NAME
UL$ACT==:UL$NOD+1               ;USER'S ACCOUNT STRING
UL$ACE==:UL$ACT+7               ;END OF ACCOUNT STRING
UL$BEG==:UL$ACE+1               ;SESSION START DATE/TIME
UL$JTY==:UL$BEG+1               ;JOB TYPE
        ULJTI$==:1              ;TIMESHARING
        ULJBA$==:2              ;BATCH
UL$BNM==:UL$JTY+1               ;BATCH JOB NAME
UL$BSQ==:UL$BNM+1               ;BATCH SEQUENCE NUMBER
UL$RMK==:UL$BSQ+1               ;SESSION REMARK
UL$RME==:UL$RMK+7               ;END OF SESSION REMARK
UL$CLS==:UL$RME+1               ;SCHEDULING CLASS
UL$PPN==:UL$CLS+1               ;PROJECT-PROGRAMMER NUMBER OF USER
UL$NM1==:UL$PPN+1               ;FIRST SIX LETTERS OF USER'S NAME
UL$NM2==:UL$NM1+1               ;LAST SIX LETTERS OF USER'S NAME
UL$BRI==:UL$NM2+1               ;BATCH REQUEST ID
UL$TDE==:UL$BRI+1               ;TERMINAL DESIGNATOR

;FORMAT OF SESSION MESSAGE (UGSES$)

US$TYP==:0                      ;TYPE OF MESSAGE
US$ACK==:US$TYP+1               ;UNIQUE MESSAGE IDENTIFIER
US$PRG==:US$ACK+1               ;PROGRAM NAME (ALWAYS LOGIN)
US$VER==:US$PRG+1               ;VERSION OF LOGIN
US$ACT==:US$VER+1               ;USER'S NEW ACCOUNT STRING
US$ACE==:US$ACT+7               ;END OF ACCOUNT STRING
US$BEG==:US$ACE+1               ;SESSION START DATE/TIME
US$RMK==:US$BEG+1               ;NEW SESSION REMARK
US$RME==:US$RMK+7               ;END OF SESSION REMARK


;FORMAT OF ATTACH MESSAGES (UGATT$)

UA$TYP==:0                      ;TYPE OF MESSAGE
UA$ACK==:UA$TYP+1               ;UNIQUE MESSAGE IDENTIFIER
UA$LIN==:UA$ACK+1               ;LINE NUMBER
UA$PRG==:UA$LIN+1               ;PROGRAM NAME (ALWAYS LOGIN)
UA$VER==:UA$PRG+1               ;VERSION LOGIN
UA$NOD==:UA$VER+1               ;NODE NAME
UA$TDE==:UA$NOD+1               ;TERMINAL DESIGNATOR
UA$TJN==:UA$TDE+1               ;TARGET JOB NUMBER
```

```
;FORMAT OF THE REQUEST FOR VALIDATION MESSAGE (UGVAL$)
;       THIS MESSAGE CAN BE EITHER IPCF OR QUEUE. UUO FORMAT.  THE SAMPLE
;       PROGRAM "VALID" SHOWS AN EXAMPLE FOR USING QUEUE. FOR VALIDATION.

UV$TYP==:0                      ;TYPE OF MESSAGE
UV$ACK==:1                      ;GIVEN TO US TO RETURN TO THE REQUESTOR OF
                                ; VALIDATION
UV$PPN==:2                      ;PPN TO VALIDATE
UV$ACT==:3                      ;BEGINNING OF ACCOUNT STRING TO VALIDATE
                                ; (MAX. OF 8 WORDS)
UV$ACE==:UV$ACT+7               ;LAST WORD OF THE ACCOUNT STRING

;FORMAT OF THE ANSWER TO A MESSAGE (UGVAC$)

UC$TYP==:0                      ;TYPE OF MESSAGE
UC$RES==:1                      ;RESPONSE -- EITHER UGTRU$ OR UGFAL$
UC$ACK==:2                      ;CODE TO RETURN TO THE REQUESTOR OF VALIDATION
UC$ERR==:3                      ;BEGINNING OF ASCIZ ERROR MESSAGE
UC$ACT==:UC$ERR                 ;ACCOUNT STRING RETURNED IF VALIDATION SUCCESS
UC$ACE==:UC$ACT+7               ;LAST WORD OF THE ACCOUNT STRING

;FORMAT OF THE SET DATE/TIME EVENT MESSAGE FROM DAEMON (UGSDT$)

UD$TYP==:0                      ;TYPE OF MESSAGE
UD$OFF==:1                      ;TIME OFFSET OF COMMAND
UD$ODT==:2                      ;OLD DATE/TIME
UD$PRG==:3                      ;NAME OF PROGRAM SENDING MESSAGE
                                ; (DAEMON)
UD$VER==:4                      ;VERSION OF PROGRAM SENDING MESSAGE
                                ; (DAEMON VERSION #)

;FORMAT OF "MAKE AN ENTRY" MESSAGE (UGENT$).

;       THIS MESSAGE IS ONLY GENERATED VIA THE QUEUE. UUO.  SEE SAMPLE PROGRAM
;       "USRENT" FOR AN EXAMPLE OF IT'S USE.
```

```
;FORMAT OF THE FIRST THREE WORDS OF ALL MOUNT AND DISMOUNT MESSAGES TO
;       CONFORM TO GALAXY-TYPE HEADER.

UX$TYP==0                       ;TYPE OF MESSAGE
UX$FLG==1                       ;FLAGS WORD
UX$COD==2                       ;ACK CODE

;FORMAT OF A USER FILE STRUCTURE MOUNT MESSAGE - UGFDM$

UF$DEV==:UX$COD+1               ;DEVICE NAME IN SIXBIT
UF$JOB==:UF$DEV+1               ;JOB NUMBER OF USER
UF$TRD==:UF$JOB+1               ;TERMINAL DESIGNATOR
UF$TNO==:UF$TRD+1               ;LINE NUMBER
UF$PNM==:UF$TNO+1               ;NAME OF PROGRAM (USUALLY PULSAR)
UF$PVR==:UF$PNM+1               ;VERSION OF PROGRAM (USUALLY PULSAR)
UF$NOD==:UF$PVR+1               ;NODE NAME OF USER'S LOCATION
UF$ACT==:UF$NOD+1               ;USER'S ACCOUNT STRING
UF$ACE==:UF$ACT+7               ;END OF ACCOUNT STRING
UF$PPN==:UF$ACE+1               ;PROJECT-PROGRAMMER NUMBER OF USER
UF$NM1==:UF$PPN+1               ;FIRST SIX LETTER OF USER'S NAME
UF$NM2==:UF$NM1+1               ;LAST SIX LETTERS OF USER'S NAME
UF$STY==:UF$NM2+1               ;TYPE OF FILE STRUCTURE
UF$PNO==:UF$STY+1               ;NUMBER OF PACKS IN FILE STRUCTURE
UF$CTY==:UF$PNO+1               ;CONTROLLER TYPE
UF$DTY==:UF$CTY+1               ;DEVICE TYPE
UF$DSP==:UF$DTY+1               ;DISPOSITION
UF$TXT==:UF$DSP+1               ;TEXT TO EXPLAIN DISPOSITION
UF$CDT==:UF$TXT+10              ;CREATION DATE/TIME OF MOUNT REQUEST
UF$SDT==:UF$CDT+1               ;SCHEDULED DATE/TIME OF MOUNT REQUEST
UF$VDT==:UF$SDT+1               ;SERVICED DATE/TIME OF MOUNT REQUEST
UF$CBR==:UF$VDT+1               ;MOUNT COUNT BEFORE REQUEST
UF$ACC==:UF$CBR+1               ;ACCESS TYPE

;FORMAT OF USER FILE STRUCTURE DISMOUNT MESSAGE (UGFDD$).  THE FOLLOWING
;       IS APPENDED TO THE MOUNT MESSAGE.

UF$SCT==:UF$ACC+1               ;MOUNT COUNT AFTER DISMOUNT
```

```
;FORMAT OF USER MAGTAPE MOUNT MESSAGE - UGMGM$

UM$DEV==:UX$COD+1              ;DEVICE NAME IN SIXBIT
UM$JOB==:UM$DEV+1              ;JOB NUMBER OF USER
UM$TRD==:UM$JOB+1              ;TERMINAL DESIGNATOR
UM$TNO==:UM$TRD+1              ;LINE NUMBER
UM$PNM==:UM$TNO+1              ;NAME OF PROGRAM (USUALLY PULSAR)
UM$PVR==:UM$PNM+1              ;VERSION OF PROGRAM (USUALLY PULSAR)
UM$NOD==:UM$PVR+1              ;NODE NAME OF USER'S LOCATION
UM$ACT==:UM$NOD+1              ;ACCOUNT STRING
UM$ACE==:UM$ACT+7              ;END OF ACCOUNT STRING
UM$PPN==:UM$ACE+1              ;PROJECT-PROGRAMMER NUMBER OF USER
UM$NM1==:UM$PPN+1              ;FIRST SIX LETTERS OF USER'S NAME
UM$NM2==:UM$NM1+1              ;LAST SIX LETTERS OF USER'S NAME
UM$CTY==:UM$NM2+1              ;CONTROLLER TYPE
UM$DSP==:UM$CTY+1              ;DISPOSITION
UM$TXT==:UM$DSP+1              ;TEXT TO EXPLAIN DISPOSITION
UM$CDT==:UM$TXT+10             ;CREATION DATE/TIME OF MOUNT REQUEST
UM$SDT==:UM$CDT+1              ;SCHEDULED DATE/TIME OF MOUNT REQUEST
UM$VDT==:UM$SDT+1              ;SERVICED DATE/TIME OF MOUNT REQUEST
UM$VID==:UM$VDT+1              ;VOLUME ID RECORDED IN VOL1 LABEL
UM$RID==:UM$VID+1              ;REEL ID VISUAL LABEL OF TAPE
UM$LTY==:UM$RID+1              ;LABEL TYPE
UM$LST==:UM$LTY+1              ;VOLUME LABEL STATE
UM$FSI==:UM$LST+1              ;FILE SET IDENTIFIER

;FORMAT OF A USER MAGTAPE DISMOUNT MESSAGE (UGMGD$).  THE FOLLOWING IS
;       APPENDED TO THE MOUNT MESSAGE.

UM$MRD==:UM$FSI+1              ;MAGTAPE READS - THOUSANDS OF CHARACTERS
UM$MWR==:UM$MRD+1              ;MAGTAPE WRITES - THOUSANDS OF CHARACTERS
UM$RRD==:UM$MWR+1              ;PHYSICAL RECORDS READ
UM$RWR==:UM$RRD+1              ;PHYSICAL RECORDS WRITTEN
UM$SRE==:UM$RWR+1              ;SOFT READ ERRORS
UM$SWE==:UM$SRE+1              ;SOFT WRITE ERRORS
UM$HRE==:UM$SWE+1              ;HARD READ ERRORS
UM$HWE==:UM$HRE+1              ;HARD WRITE ERRORS
```

```
;FORMAT OF A USER DECTAPE MOUNT MESSAGE - UGDTM$

UD$DEV==:UX$COD+1               ;DEVICE NAME IN SIXBIT
UD$JOB==:UD$DEV+1               ;JOB NUMBER OF USER
UD$TRD==:UD$JOB+1               ;TERMINAL DESIGNATOR
UD$TNO==:UD$TRD+1               ;LINE NUMBER
UD$PNM==:UD$TNO+1               ;NAME OF PROGRAM (USUALLY PULSAR)
UD$PVR==:UD$PNM+1               ;VERSION OF PROGRAM (USUALLY PULSAR)
UD$NOD==:UD$PVR+1               ;NODE NAME OF USER'S LOCATION
UD$ACT==:UD$NOD+1               ;ACCOUNT STRING
UD$ACE==:UD$ACT+7               ;END OF ACCOUNT STRING
UD$PPN==:UD$ACE+1               ;PROJECT-PROGRAMMER NUMBER OF USER
UD$NM1==:UD$PPN+1               ;FIRST SIX LETTERS OF USER'S NAME
UD$NM2==:UD$NM1+1               ;LAST SIX LETTERS OF USER'S NAME
UD$DSP==:UD$NM2+1               ;DISPOSITION
UD$TXT==:UD$DSP+1               ;TEXT TO EXPLAIN DISPOSITION
UD$CDT==:UD$TXT+10              ;CREATION DATE/TIME OF MOUNT REQUEST
UD$SDT==:UD$CDT+1               ;SCHEDULED DATE/TIME OF MOUNT REQUEST
UD$VDT==:UD$SDT+1               ;SERVICED DATE/TIME OF MOUNT REQUEST
UD$VID==:UD$VDT+1               ;VOLUME ID RECORDED ON DECTAPE
UD$RID==:UD$VID+1               ;REEL ID VISUAL LABEL OF DECTAPE

;FORMAT OF A USER DECTAPE DISMOUNT MESSAGE (UGDTD$).  THE FOLLOWING IS APPENDE
;       TO THE MOUNT MESSAGE.

UD$DRD==:UD$RID+1               ;DECTAPE READS - BLOCKS
UD$DWR==:UD$DRD+1               ;DECTAPE WRITES - BLOCKS
```

```
;FORMAT OF A DISK SPINDLE SPIN-UP MESSAGE - UGSPM$

US$DEV==:UX$COD+1               ;DISK UNIT NAME IN SIXBIT
US$JOB==:US$DEV+1               ;JOB NUMBER OF PULSAR
US$TRD==:US$JOB+1               ;TERMINAL DESIGNATOR
US$TNO==:US$TRD+1               ;LINE NUMBER
US$PNM==:US$TNO+1               ;NAME OF PROGRAM (USUALLY PULSAR)
US$PVR==:US$PNM+1               ;VERSION OF PROGRAM (USUALLY PULSAR)
US$NOD==:US$PVR+1               ;NODE NAME OF PULSAR'S LOCATION
US$FSN==:US$NOD+1               ;FILE STRUCTURE NAME
US$STY==:US$FSN+1               ;FILE STRUCTURE TYPE
US$CTY==:US$STY+1               ;CONTROLLER TYPE
US$DTY==:US$CTY+1               ;DEVICE TYPE
US$DPI==:US$DTY+1               ;DISK PACK IDENTIFIER
US$PNO==:US$DPI+1               ;NUMBER OF PACKS IN FILE STRUCTURE
US$MTH==:US$PNO+1               ;M OF N COUNT
US$DTM==:US$MTH+1               ;DATE/TIME PACK WAS SPUN UP

;FORMAT OF A DISK SPINDLE SPIN-DOWN MESSAGE (UGSPD$) IS THE SAME AS A MOUNT MESSAGE.
```

```
;FORMAT OF A DISK USAGE ENTRY FROM BACKUP - UGDUE$

UB$ACN==:UX$COD+1               ;NUMBER OF ACCOUNT STRINGS
UB$JOB==:UB$ACN+1               ;JOB NUMBER OF BACKUP
UB$TRD==:UB$JOB+1               ;TERMINAL DESIGNATOR
UB$TNO==:UB$TRD+1               ;TERMINAL NUMBER
UB$NOD==:UB$TNO+1               ;NODE NAME
UB$PNM==:UB$NOD+1               ;PROGRAM NAME (BACKUP)
UB$PVR==:UB$PNM+1               ;PROGRAM VERSION NUMBER
UB$TAU==:UB$PVR+1               ;TOTAL ALLOCATED DISK USAGE
UB$TWU==:UB$TAU+1               ;TOTAL WRITTEN DISK USAGE
UB$TNF==:UB$TWU+1               ;TOTAL NUMBER OF FILES
UB$FSN==:UB$TNF+1               ;FILE STRUCTURE NAME
UB$PPN==:UB$FSN+1               ;PPN
UB$FST==:UB$PPN+1               ;FILE STRUCTURE TYPE
UB$CNT==:UB$FST+1               ;CONTROLLER TYPE
UB$DVT==:UB$CNT+1               ;DEVICE TYPE
UB$QIN==:UB$DVT+1               ;LOGGED IN QUOTA
UB$QOU==:UB$QIN+1               ;LOGGED OUT QUOTA
UB$LLG==:UB$QOU+1               ;LAST LOGIN DATA/TIME (OLD FORMAT)
UB$LAT==:UB$LLG+1               ;LAST ACCOUNTING DATE/TIME
UB$EXP==:UB$LAT+1               ;EXPIRED DIRECTORY FLAG
UB$UPF==:UB$EXP+1               ;UFD WAS PROTECTED FLAG
UB$FPF==:UB$UPF+1               ;SOME FILES WERE PROTECTED FLAG
UB$ABO==:UB$FPF+1               ;ACCOUNT STRING BUFFER OVERFLOWED

;THE FOLLOWING 4 ITEMS ARE REPEATED FOR EACH ACCOUNT STRING IN THE UFD

UB$ACT==:UB$ABO+1               ;ACCOUNT STRING
UB$BAL==:UB$ACT+10              ;BLOCKS ALLOCATED TO THIS ACCOUNT STRING
UB$BWR==:UB$BAL+1               ;BLOCKS WRITTEN
UB$NFL==:UB$BWR+1               ;FILE WITH THIS ACCOUNT STRING

UB$END==:UB$NFL+1               ;LENGTH OF THE IPCF MESSAGE (NOT REALLY)

;COMPUTE THE MAXIMUM NUMBER OF ACCOUNTS STRINGS THAT CAN BE PASSED IN THIS
;       IPCF FORMAT. ONLY 1 IPCF MESSAGE CAN BE USED SO THIS IS IT.

UB$MAC==:<1000-UB$ACT>/<UB$END-UB$ACT>
```

```
;FORMAT OF A REQUEST FOR ACCESS CONTROL VALIDATION.

;       THIS MESSAGE IS ONLY AVAILABLE VIA THE QUEUE. UUO.  THE SYMBOLS DEFINED
;       BELOW ARE USED AS THE DATA ITEM DESCRIPTORS FOR THE SUB-FUNCTION BLOCK.
;       SEE THE SAMPLE PROGRAM "ACCCHK" FOR AN EXAMPLE OF IT'S USE.

.UGTYP==:0                      ;TYPE OF ACCESS CHECK
        UG.VER==:0              ;VERIFY PPN, PASSWORD, AND ACCOUNT STRING
        UG.SPV==:1              ;SPRINT VERIFY PPN, PASSWORD, AND ACCT STRING
                                ;MORE FUNCTIONS LATER
.UGACT==:1                      ;ACCOUNT STRING BLOCK
.UGPPN==:2                      ;PPN BLOCK
.UGPSW==:3                      ;PASSWORD BLOCK
```

```
SUBTTL  TOPS-10 ACCT.SYS entry definitions

;*** Note *** Still under TOPS-10 conditional

.ACWRD==0       ;FIRST WORD OF THE FILE ACCT.SYS
        AC.VRS==777777B17       ;FORMAT VERSION NUMBER OF THE FILE
                .ACCVN==4       ;CURRENT VERSION NUMBER
        AC.LEN==777777B35       ;LENGTH OF EACH ENTRY
                .ACLEN==16      ;VERSION 4 IS 14(10) WORDS LONG

;ACCT.SYS ENTRY FORMAT

.ACPPN==0       ;PROJECT PROGRAMMER NUMBER
.ACPSW==1       ;PASSWORD IN SIXBIT
.ACPRV==2       ;PRIVILEGED BITS
        AC.IPC==1B0     ;JOB CAN DO IPCF PRIVILEGED FUNCTIONS
        AC.DSP==3B2     ;HIGHEST DISK PRIORITY
        AC.MET==1B3     ;JOB CAN DO METER UUO
        AC.POK==1B4     ;JOB CAN POKE THE MONITOR
        AC.CPU==1B5     ;JOB CAN CHANGE ITS CPU SPECIFICATION
        AC.HPQ==74B9    ;HIGHEST HPQ THAT JOB CAN SET
        AC.SPL==1B10    ;JOB CAN SET NO SPOOL
        AC.RTT==1B13    ;JOB CAN DO REAL TIME TRAPPING (RTTRP.) UUO
        AC.LOK==1B14    ;JOB CAN DO LOCK UUO
        AC.TRP==1B15    ;JOB CAN DO TRPSET UUO
        AC.SPC==1B16    ;JOB CAN PEEK/SPY AT ALL OF CORE
        AC.SPM==1B17    ;JOB CAN PEEK/SPY AT THE MONITOR
        AC.CUS==777777B35 ;RESERVED FOR CUSTOMER
.ACNM1==3       ;USER'S NAME IN SIXBIT - FIRST WORD
.ACNM2==4       ;USER'S NAME IN SIXBIT - SECOND WORD
.ACLIT==5       ;TIMES ALLOWED TO LOGIN
        AC.WDH==77777777B23     ;WEEKDAY HOURS, 0000-2359
        AC.WEH==7777B35         ;WEEKEND HOURS, 2-HOUR SHIFTS 0000-2200
.ACCIP==6       ;CORE AND IPCF PARAMETERS
        AC.NPP==777B8   ;MAXIMUM NUMBER OF PHYSICAL PAGES
        AC.NVP==777B17  ;MAXIMUM NUMBER OF VIRTUAL PAGES
        AC.SND==777B26  ;MAXIMUM NUMBER OF SENDS
        AC.RCV==777B35  ;MAXIMUM NUMBER OF RECEIVES
```

```
.ACPRO==7       ;PROFILE BITS
        AC.WDT==1B0     ;WATCH DAYTIME
        AC.WRT==1B1     ;WATCH RUNTIME
        AC.WWA==1B2     ;WATCH WAIT
        AC.RED==1B3     ;WATCH READ
        AC.WRI==1B4     ;WATCH WRITE
        AC.CDR==1B5     ;SPOOL CDR
        AC.CDP==1B6     ;SPOOL CDP
        AC.PTP==1B7     ;SPOOL PTP
        AC.PLT==1B8     ;SPOOL PLT
        AC.LPT==1B9     ;SPOOL LPT
        AC.WVR==1B10    ;WATCH VERSION
        AC.WMT==1B11    ;WATCH MTA
        AC.WFL==1B12    ;WATCH FILE
        AC.OPR==7B23    ;OPERATOR PRIVILEGE FIELD (SEE UUOSYM FOR SYMBOLS)
                ;.OBNOP==0      ;NO OPERATOR PRIVILEGES
                ;.OBSDP==1      ;SYSTEM OPERATOR PRIVILEGES
                ;.OBLOP==2      ;LOCAL OPERATOR PRIVILEGES
                ;.OBROP==3      ;REMOTE OPERATOR PRIVILEGES
        AC.RMK==1B24    ;REMARK IS REQUIRED
        AC.ACT==1B25    ;ACCOUNT IS REQUIRED
        AC.LOC==1B26    ;USER MAY LOGIN AT LOCAL TERMINAL
        AC.ROP==1B27    ;USER MAY LOGIN AT REMOTE OPR OR CTY
        AC.DST==1B28    ;USER MAY LOGIN AT DATA SET TERMINAL
        AC.RMT==1B29    ;USER MAY LOGIN AT REMOTE TERMINAL
        AC.SBJ==1B30    ;USER MAY LOGIN SUBJOB OF A BATCH JOB
        AC.BAT==1B31    ;USER MAY LOGIN UNDER BATCH
        AC.NRT==1B32    ;NAME REQUIRED UNDER TIMESHARING
        AC.NRB==1B33    ;NAME REQUIRED UNDER BATCH
        AC.PRT==1B34    ;PASSWORD REQUIRED FOR TIMESHARING
        AC.PRB==1B35    ;PASSWORD REQUIRED FOR BATCH
.ACCUS==10      ;CUSTOMER USER PROFILE BITS
.ACPGM==11      ;SIXBIT NAME OF PROGRAM TO RUN
.ACDEV==12      ;SIXBIT DEVICE FROM WHERE TO RUN PROGRAM
.ACDIR==13      ;DIRECTORY FORM WHERE TO RUN PROGRAM
.ACCNO==14      ;CHARGE NUMBER
.ACESE==15      ;EXPIRATION DATE,,SCHEDULER AND END/DEQ PARAMETERS
        AC.EXP==777777B17       ;EXPIRATION DATE
        AC.SCD==777B26          ;SCHEDULER TYPE
        AC.EDQ==777B35          ;ENQ/DEQ/QUOTA

> ;END OF TOPS-10 CONDITIONAL FROM WAY BACK

        END                     ;END OF ACTSYM.MAC
```

INDEX

+1 return, 1-10
+2 return, 1-10

18-bit address, 1-3
23-bit address, 1-2
30-bit address, 1-3

AC's, 1-1
AC's,
  Returning process, 3-338
  Setting process, 3-384
ACCES JSYS, 3-2
Access,
  Append, 2-8, 3-298
  Directory, 2-9, 3-2
  Execute, 2-8, 3-298
  Execute-only, 2-8
  File, 2-8, 3-298
  Gaining directory, 3-2
  Modes, 2-8
  Page, 3-355, 3-491
  Read, 2-8, 3-298
  Relinquishing directory, 3-2
  Thawed, 2-8, 3-298
  Write, 2-8, 3-298
Access control, 2-65, 3-157
Access modes, 2-8
Access-control,
  Functions, 2-65
Access-control functions, 2-65
Access-control JSYS's,
  GETOK%, 3-157
  GIVOK%, 3-167
  RCVOK%, 3-330
Access-control program, 3-167, 3-416
Accessibility,
  Setting page, 3-428
Account,
  Changing, 3-38
  Returning file's, 3-144
  Returning job's, 3-143
Accounting,
  Functions, 2-1
  System, 3-475
Accounting functions, 2-1
Accounting JSYS's,
  CACCT, 3-38
  GACCT, 3-143
  GACTF, 3-144
Accounting JSYS's (Cont.)
  SACTF, 3-369
  USAGE, 3-475
Accounts,
  Verifying, 3-482
Accumulators, 1-1
Acquiring physical memory, 3-309
Activation,
  Interrupt channel, 3-8
ADBRK JSYS, 3-5
Adding a table entry, 3-452
Address, 1-7
  18-bit, 1-3
  23-bit, 1-2
  30-bit, 1-3
  global, 1-3
  Section-relative, 1-3
Address breaks, 3-5
Address global, 1-3
Address section-relative, 1-3
Addresses,
  Assigning disk, 3-112
  Obtaining interrupt table, 3-490
  Setting interrupt table, 3-400, 3-494
Advising,
  Terminal, 2-53
AIC JSYS, 3-8
ALLOC JSYS, 3-9
Allocation,
  Device, 3-9
  Returning disk, 3-172
Analysis,
  System performance, 3-418
ANSI ASCII mode, 2-43
Append access, 2-8, 3-298
ARCF JSYS, 3-11
Archive-related JSYS's,
  ARCF, 3-11
  DELDF, 3-92
  DELF, 3-94
  DELNF, 3-96
  RFTAD, 3-349
Archive/virtual disk system, 2-81
Arguments,
  JSYS, 1-1, 1-2
Arguments JSYS, 1-1
ARPANET, 3-329
ARPANET host,
  Flushing an, 3-140
ARPANET host information,
  Returning, 3-176

ARPANET queue,
  Assigning, 3-16
ARPANET-related JSYS's,
  ASNSQ, 3-16
  ATNVT, 3-20
  CVHST, 3-89
  CVSKT, 3-90
  FLHST, 3-140
  GTHST%, 3-176
  RCVIM, 3-329
  RELSQ, 3-336
  SNDIM, 3-417
ASCII mode,
  ANSI, 2-43
ASCII strings, 1-7
ASND JSYS, 3-15
ASNSQ JSYS, 3-16
Assigning a device, 3-15
Assigning ARPANET queue, 3-16
Assigning devices, 3-335
Assigning disk addresses, 3-112
Assigning terminal interrupt, 3-19
Association,
  Physical/logical tape-drive, 3-247
ATACH JSYS, 3-17
ATI JSYS, 3-19
ATNVT JSYS, 3-20
Attaching a job, 3-17
Author,
  Returning file, 3-166
  Setting file, 3-394

Backing up pointer, 3-22
Base information,
  Monitor data, 3-415, 3-470
Bits,
  Magnetic tape status, 2-44
  MTA: status, 2-39
  PCDP: status, 2-34
  PCDR: status, 2-33
  PLPT: status, 2-37
  Terminal status, 2-45
  TTY: status, 2-45
Bits magnetic tape,
  Status, 2-44
Bits terminal,
  Status, 2-45
BKJFN JSYS, 3-22
Block,
  File descriptor, 2-10
  Job storage, 3-361
Block (Cont.)
  Returning file descriptor, 3-175
Blocking,
  Elapsed time process, 3-464
BOOT JSYS, 3-23
BOUT JSYS, 3-37
Breaks,
  Address, 3-5
Buffer,
  Clearing file input, 3-39
  Clearing file output, 3-40
  Rescan, 3-361
  Testing file input, 3-395
  Testing file output, 3-422, 3-423
Buffered I/O, 2-40
Byte count,
  File, 2-20
Byte input, 3-21, 3-303
Byte input,
  Random, 3-351
Byte output, 3-37, 3-304
Byte output,
  Random, 3-358
Byte pointer, 1-4, 1-6, 1-7
Byte pointer,
  local, 1-4
  one-word global, 1-4
  two-word global, 1-4
  two-word local, 1-4
Byte pointer local, 1-4
Byte pointer one-word global, 1-4
Byte size,
  Resetting file, 3-385
Byte-I/O JSYS's,
  BIN, 3-21
  BOUT, 3-37
  PBIN, 3-303
  PBOUT, 3-304
  RIN, 3-351, 3-358
Byte-size,
  Returning file, 3-339

CACCT JSYS, 3-38
Calls,
  Privileged monitor, 2-82
Capabilities, 2-64
  Enabling, 3-134
  Functions, 2-63
  Job, 2-64
  Process, 2-63, 2-64
Capabilities functions, 2-63

Carriage control tape, 2-36
CCOC word, 2-48, 2-49
CCOC words, 3-340, 3-386
CCOC words,
  Returning, 3-340
  Setting, 3-386
CDP:, 2-33, 2-35, 2-54
CDR:, 2-33, 2-34, 2-55
CFBIF JSYS, 3-39
CFBOF JSYS, 3-40
CFORK JSYS, 3-41
Changing account, 3-38
Channel,
  Reserving a, 3-102
  Software interrupt, 2-57
Channel activation,
  Interrupt, 3-8
Channels,
  Deactivating interrupt, 3-107
  Panic, 2-58, 2-60
Character editing, 2-2
Characteristics,
  Returning device, 3-121
  TTY:, 2-51, 2-52
Characters,
  PLPT: control, 2-37
  Wildcard, 2-3
CHFDB JSYS, 3-43
CHKAC JSYS, 3-45
CIS JSYS, 3-47
Class,
  Wakeup, 2-48, 2-49, 2-50
Clearing file input buffer, 3-39
Clearing file output buffer, 3-40
Clearing software interrupt system, 3-47
Clock,
  Returning high-precision, 3-202
Clock-related JSYS's,
  HPTIM, 3-202
  METER%, 3-220
CLOSF JSYS, 3-48
Closing a file, 3-48
Closing process files, 3-50
CLZFF JSYS, 3-50
Codes,
  JSYS error, 1-10
  Terminal interrupt, 2-60
Command parsing, 3-52
COMND JSYS, 3-52
Comparing strings, 3-437, 3-485
Comparison,
  Wild string, 3-485
Compatibility package, 2-70, 3-373
Compatibility package entry vector, 3-373
Compatible mode,
  Industry, 2-43
Condition,
  Setting error, 3-377
Connection,
  Creating NVT, 3-20
Control,
  Access, 2-65, 3-157
  I/O format, 2-76
  program, 1-10
  Scheduler, 3-404
Control characters,
  PLPT:, 2-37
Control program, 1-10
Control tape,
  Carriage, 2-36
Control word,
  Scheduler priority, 3-403
Controlling terminal,
  Redefining, 3-371
Conventions,
  Manual pointer, 1-7
Conversion,
  Date/time, 2-79
  I/O data, 2-75
Converting host number, 3-89
Converting internal date/time, 3-294
Converting socket number, 3-90
Converting to internal date/time, 3-204
Count,
  File byte, 2-20
CRDIR JSYS, 3-75
Creating a logical name, 3-87
Creating a new job, 3-81
Creating a section, 3-410
Creating an inferior process, 3-41
Creating NVT connection, 3-20
Creating sections, 3-363
CRJOB JSYS, 3-81
CRLNM JSYS, 3-87
Current section, 1-3
CVHST JSYS, 3-89
CVSKT JSYS, 3-90

Data base information,
  Monitor, 3-415, 3-470
Data conversion,
  I/O, 2-75
Data mode,
  Terminal, 2-46, 2-47
Data modes,
  Hardware, 2-42
  Software, 2-54, 2-56,
    2-57
Data vector,
  Program, 3-305
Data-conversion,
  Functions, 2-75
Data-conversion functions,
    2-75
Date,
  Setting system, 3-436
  System, 1-9
  system, 1-9
  System, 2-79
Date and time,
  Offline expiration, 3-349
  Online expiration, 3-349
Date system, 1-9
Date/time,
  Converting internal,
    3-294
  Converting to internal,
    3-204
  File, 3-349
  Format internal, 1-9
  Functions, 2-79
  Inputting, 3-205, 3-207
  Outputting, 3-295, 3-297
  Setting file, 3-392
  Standard, 1-9
Date/time conversion, 2-79
Date/time format, 2-79
  Internal, 1-9
Date/time functions, 2-79
Date/time JSYS's,
  FOO, 3-436
  GTAD, 3-171
  IDCNV, 3-204
  IDTIM, 3-205
  IDTNC, 3-207
  ODCNV, 3-294
  ODTIM, 3-295
  ODTNC, 3-297
  RFTAD, 3-349
  SFTAD, 3-392
Date/time standard, 1-9
DCN:, 2-33, 2-55
Deactivating interrupt
    channels, 3-107
Deassigning terminal
    interrupt, 3-116
DEBRK JSYS, 3-91
Debugging,
  Program, 3-5
Debugging JSYS's,
  ADBRK, 3-5
  MDDT%, 3-219
  PEEK, 3-308
  SNOOP, 3-418
  SWTRP%, 3-448
  UTEST, 3-479
DECnet logical link, 3-254
DECnet remote process,
    3-254
Default designator,
  Universal, 1-8
Deferred terminal interrupt,
    2-62
DELDF JSYS, 3-92
Deleting a table entry,
    3-453
Deleting files, 3-94, 3-96
DELF JSYS, 3-94
DELNF JSYS, 3-96
Density mode,
  High, 2-44
DEQ JSYS, 3-97
Descriptor block,
  File, 2-10
  Returning file, 3-175
Designator,
  Destination, 1-6
  Device, 1-6, 1-8
  File, 1-7
  Primary input, 3-332
  Primary output, 3-332
  Source, 1-6
  Source/destination, 1-6
  Terminal, 1-6, 1-8
  Universal default, 1-8
Designator destination, 1-6
Designator device, 1-6
Designator source, 1-6
Designator terminal, 1-6
Destination,
  Designator, 1-6
Destination designator, 1-6
Detaching a job, 3-115
Device,
  Assigning a, 3-15
  Designator, 1-6
  Functions, 2-5, 2-32
  Manipulating a spooled,
    3-431
  Opening a, 2-5
Device allocation, 3-9
Device characteristics,
  Returning, 3-121
Device designator, 1-6, 1-8
Device functions, 2-5, 2-32

Device functions,
  MT, 3-277
Device mode,
  Setting, 3-443
Device name string,
  Translating, 3-438
Device status,
  Returning, 3-147
Device string,
  Translating, 3-99
Device-control functions,
  3-248
Device-related JSYS's,
  ALLOC, 3-9
  ASND, 3-15
  CLOSF, 3-48
  CLZFF, 3-50
  DEVST, 3-99
  DIAG, 3-102
  DSKAS, 3-112
  DSKOP, 3-113
  DVCHR, 3-121
  GDSKC, 3-146
  GDSTS, 3-147
  GTDAL, 3-172
  LPINI, 3-218
  MTALN, 3-247
  MTOPR, 3-248
  MTU%, 3-277
  RELD, 3-335
  SDSTS, 3-375
  STPAR, 3-443
Devices, 2-32
  Assigning, 3-335
  Releasing, 3-335
DEVST JSYS, 3-99
DFIN JSYS, 3-100
DFOUT JSYS, 3-101
DIBE JSYS, 3-106
DIC JSYS, 3-107
Directory access, 2-9, 3-2
Directory access,
  Gaining, 3-2
  Relinquishing, 3-2
Directory information,
  Returning, 3-173
Directory name string,
  Translating, 3-444
Directory number,
  Translating, 3-109
Directory-related JSYS's,
  ACCES, 3-2
  CHKAC, 3-45
  CRDIR, 3-75
  DIRST, 3-109
  GJINF, 3-168
  GNJFN, 3-169
  GTDIR, 3-173
  RCDIR, 3-322
Directory-related JSYS's
  (Cont.)
  SPOOL, 3-431
  STDEV, 3-438
  STPPN, 3-444
DIRST JSYS, 3-109
Disabling interrupt system,
  3-108
Disk addresses,
  Assigning, 3-112
Disk allocation,
  Returning, 3-172
Disk system,
  Archive/virtual, 2-81
Disk usage,
  Returning, 3-146
Dismissing a process, 3-106,
  3-110, 3-111, 3-483
Dismissing an interrupt,
  2-62
Dismissing interrupt, 3-91
DISMS JSYS, 3-110
DOBE JSYS, 3-111
Double-precision input,
  3-100
Double-precision output,
  3-101
DSK:, 2-33, 2-55
DSKAS JSYS, 3-112
DSKOP JSYS, 3-113
DTACH JSYS, 3-115
DTI JSYS, 3-116
Dump input, 3-117
Dump mode, 2-43
Dump output, 3-119
Dump-I/O JSYS's,
  DUMPI, 3-117
  DUMPO, 3-119
DUMPI JSYS, 2-41, 3-117
DUMPO JSYS, 3-119
Duplex mode,
  Full, 2-46, 2-48
  Half, 2-46, 2-48
DVCHR JSYS, 3-121

EBOX/MBOX meter values,
  Returning, 3-220
Echo mode, 2-46, 2-47
Editing,
  Character, 2-2
EIR JSYS, 3-123
Elapsed system restart time,
  Returning, 3-465
Elapsed time process
  blocking, 3-464
Enabling capabilities,
  3-134

Enabling software interrupt
system, 3-123
End-of-file,
Testing for, 2-22
End-of-file limit, 2-21
ENQ JSYS, 3-124
ENQ/DEQ JSYS's,
ENQ, 3-124
ENQC, 3-130
FOO, 3-97
ENQC JSYS, 3-130
Entering MDDT, 3-219
Entry,
Adding a table, 3-452
Deleting a table, 3-453
Entry vector, 2-74
Compatibility package,
3-373
Returning PA1050, 3-145
Returning process, 3-162,
3-489
Returning RMS, 3-148
RMS, 3-376
Setting process, 3-383,
3-496
EOF limit, 2-21
EPCAP JSYS, 3-134
ERCAL, 1-10, 2-21
ERJMP, 1-10, 2-21
Error,
Return, 1-10
Returning most recent,
3-153
Error codes,
JSYS, 1-10
Error condition,
Setting, 3-377
Error file,
System, 3-450
Error messages, 2-24
Error return, 1-10
Error strings, 2-24
Outputting, 3-136
Translating, 3-135
Error-processing JSYS's,
ERSTR, 3-135
ESOUT, 3-136
GETER, 3-153
SETER, 3-377
SYERR, 3-450
Errors,
I/O, 2-21
JSYS, 1-10
ERSTR JSYS, 3-135
ESOUT JSYS, 3-136
Execute access, 2-8, 3-298
Execute-only, 2-9
Execute-only access, 2-8
Execute-only files, 2-68
Execute-only processes,
2-68
Execution,
Resuming process, 3-481
Expiration date and time,
Offline, 3-349
Online, 3-349
Expunging files, 3-92

FDB, 2-10
Modifying the, 2-10
Reading the, 2-10
FE:, 2-33
FFFFP JSYS, 3-137
FFORK JSYS, 3-138
FFUFP JSYS, 3-139
.FHINF, 1-9
.FHJOB, 1-9
.FHSAI, 1-9
.FHSLF, 1-9
.FHSUP, 1-9
.FHTOP, 1-9
File,
Closing a, 3-48
Functions, 2-1
Getting a save, 3-149
Nonsharable save, 2-70
Opening a, 2-5, 3-298
Primary input, 2-20
Primary output, 2-20
Recognition, 2-2
Renaming a, 3-356
Sharable save, 2-71
System error, 3-450
File access, 2-8, 3-298
File author,
Returning, 3-166
Setting, 3-394
File byte count, 2-20
File byte size,
Resetting, 3-385
File byte-size,
Returning, 3-339
File date/time, 3-349
Setting, 3-392
File descriptor block, 2-10
Returning, 3-175
File designator, 1-7
File functions, 2-1
File handle, 2-3
Indexable, 2-4
File handle indexable, 2-4
File input buffer,
Clearing, 3-39
Testing, 3-395
File number,
Indexable job, 1-5

File number (Cont.)
  job, 1-5
File number job, 1-5
File number job indexable,
    1-5
File opening a, 2-5
File output buffer,
  Clearing, 3-40
  Testing, 3-422, 3-423
File page,
  Finding first free, 3-137
  Finding first used, 3-139
File pages,
  Updating, 3-474
File pointer,
  Setting, 3-390
File recognition, 2-2
File specification, 2-1
  Parse-only, 2-4
  Returning, 3-211
File status,
  Returning, 3-198
  Setting, 3-445
File's account,
  Returning, 3-144
File-archival,
  Functions, 2-81
File-archival functions,
    2-81
File-related JSYS's,
  BKJFN, 3-22
  CFBIF, 3-39
  CFBOF, 3-40
  CHFDB, 3-43
  CHKAC, 3-45
  CLOSF, 3-48
  CLZFF, 3-50
  DELDF, 3-92
  DELF, 3-94
  DELNF, 3-96
  DIBE, 3-106
  DOBE, 3-111
  FFFFP, 3-137
  FFUFP, 3-139
  GACTF, 3-144
  GFUST, 3-166
  GNJFN, 3-169
  GTFDB, 3-175
  GTJFN(long), 3-187
  GTJFN(short), 3-179
  GTSTS, 3-198
  JFNS, 3-211
  OPENF, 3-298
  PMAP, 3-310
  RFBSZ, 3-339
  RFMOD, 3-341
  RFPTR, 3-344
  RFTAD, 3-349
  RLJFN, 3-354
File-related JSYS's (Cont.)
  RNAMF, 3-356
  SFBSZ, 3-385
  SFPTR, 3-390
  SFTAD, 3-392
  SFUST, 3-394
  SIBE, 3-395
  SIZEF, 3-402
  SOBE, 3-422
  SOBF, 3-423
  SPJFN, 3-429
  STSTS, 3-445
  SWJFN, 3-447
  UFPGS, 3-474
Files,
  Closing process, 3-50
  Deleting, 3-94, 3-96
  Execute-only, 2-68
  Expunging, 3-92
  Opening, 3-298
  Save, 2-70
Files opening, 3-298
Finding first free file
    page, 3-137
Finding first used file
    page, 3-139
First free file page,
  Finding, 3-137
First used file page,
  Finding, 3-139
Flags,
  Setting monitor, 3-415
  Testing monitor, 3-470
FLIN JSYS, 3-141
Floating-point input, 3-141
Floating-point output,
    3-142
FLOUT JSYS, 3-142
Flushing an ARPANET host,
    3-140
Fork handle,
  Getting a, 3-163
Form GTJFN,
  Long, 3-187
  Short, 3-179
Format,
  Date/time, 2-79
  Internal date/time, 1-9
Format control,
  I/O, 2-76
Format internal date/time,
    1-9
Format-controlling,
  Functions I/O, 2-76
Format-controlling
    functions,
  I/O, 2-76
Free file page,
  Finding first, 3-137

Freezing a process, 3-138
Frozen process, 3-342
Full duplex mode, 2-46, 2-48
Functions,
  Access-control, 2-65
  Accounting, 2-1
  Capabilities, 2-63
  Data-conversion, 2-75
  Date/time, 2-79
  Device, 2-5, 2-32
  Device-control, 3-248
  File, 2-1
  File-archival, 2-81
  I/O, 2-20
  I/O format-controlling, 2-76
  Information-obtaining, 2-24
  Line printer, 2-36
  Magnetic tape, 2-39
  Mountable-structure, 3-230
  MT device, 3-277
  Network, 3-286
  Privileged, 2-82
  Process-control, 2-63
  Process-controling, 2-67
  PSI, 2-57
  Software interrupt, 2-57
  Terminal, 2-45
Functions access-control, 2-65
Functions accounting, 2-1
Functions capabilities, 2-63
Functions data-conversion, 2-75
Functions date/time, 2-79
Functions device, 2-5, 2-32
Functions file, 2-1
Functions file-archival, 2-81
Functions I/O, 2-20
Functions I/O format-controlling, 2-76
Functions information-obtaining, 2-24
Functions line printer, 2-36
Functions magnetic tape, 2-39
Functions privileged, 2-82
Functions process-control, 2-63
Functions process-controling,
Functions (Cont.)
  2-67
Functions PSI, 2-57
Functions software interrupt, 2-57
Functions terminal, 2-45

GACCT JSYS, 3-143
GACTF JSYS, 3-144
Gaining directory access, 3-2
GCVEC JSYS, 3-145
GDSKC JSYS, 3-146
GDSTS JSYS, 3-147
GDVEC JSYS, 3-148
GET JSYS, 3-149
GETAB JSYS, 3-152
GETER JSYS, 3-153
GETJI JSYS, 3-154
GETNM JSYS, 3-156
GETOK JSYS, 3-416
GETOK% JSYS, 3-157
Getting a fork handle, 3-163
Getting a save file, 3-149
GEVEC JSYS, 3-162
GFRKH JSYS, 3-163
GFRKS JSYS, 3-164
GFUST JSYS, 3-166
GIVOK% JSYS, 3-167
GJINF JSYS, 3-168
Global,
  address, 1-3
  byte pointer one-word, 1-4
Global address, 1-3
Global byte pointer,
  one-word, 1-4
  two-word, 1-4
Global page numbers, 1-3
GNJFN JSYS, 3-169
GPJFN JSYS, 3-170
Greenwich Mean Time, 1-10
GTAD JSYS, 3-171
GTDAL JSYS, 3-172
GTDIR JSYS, 3-173
GTFDB JSYS, 3-175
GTHST% JSYS, 3-176
GTJFN,
  Long form, 3-187
  Short form, 3-179
GTJFN JSYS, 3-179, 3-187
GTRPI JSYS, 3-194
GTRPW JSYS, 3-197
GTSTS JSYS, 3-198
GTTYP JSYS, 3-199

Half duplex mode, 2-46,
    2-48
HALTF JSYS, 3-200
Halting a process, 3-200,
    3-201
Halting system, 3-203
Handle,
  File, 2-3
  Getting a fork, 3-163
  Indexable file, 2-4
  Page, 3-355
  Process, 1-9
  Process/file, 1-9
  Relative process, 1-9
  Releasing a process,
    3-345
  Section, 3-363
Handle indexable,
  File, 2-4
Handle page, 3-355
Handle process/file, 1-9
Handle relative,
  Process, 1-9
Handle section, 3-363
Hardware data modes, 2-42
HFORK JSYS, 3-201
High density mode, 2-44
High-precision clock,
  Returning, 3-202
Histogram,
  PC, 3-418
Host,
  Flushing an ARPANET,
    3-140
Host information,
  Returning ARPANET, 3-176
Host number,
  Converting, 3-89
HPTIM JSYS, 3-202
HSYS JSYS, 3-203

I/O,
  Buffered, 2-40
  Functions, 2-20
  Modes, 2-54
  Unbuffered, 2-41
I/O data conversion, 2-75
I/O errors, 2-21
I/O format control, 2-76
I/O format-controlling,
  Functions, 2-76
I/O format-controlling
    functions, 2-76
I/O functions, 2-20
I/O JSYS's,
  BIN, 3-21
  BOUT, 3-37
I/O JSYS's (Cont.)
  CFBIF, 3-39
  CFBOF, 3-40
  DFIN, 3-100
  DFOUT, 3-101
  DUMPI, 3-117
  DUMPO, 3-119
  FLIN, 3-141
  FLOUT, 3-142
  IDTIM, 3-205
  IDTNC, 3-207
  NIN, 3-285
  NOUT, 3-292
  PBIN, 3-303
  PBOUT, 3-304
  PSOUT, 3-321
  RDTTY, 3-332
  RIN, 3-351, 3-358
  SFPTR, 3-390
  SIBE, 3-395
  SIN, 3-396
  SINR, 3-398
  SOBE, 3-422
  SOBF, 3-423
  SOUT, 3-424
  SOUTR, 3-426
  TEXTI, 3-457
  TTMSG, 3-472
  USRIO, 3-478
I/O modes, 2-54
IDCNV JSYS, 3-204
IDTIM JSYS, 3-205
IDTNC JSYS, 3-207
IIC JSYS, 3-209
Immediate terminal
    interrupt, 2-62
Indexable,
  File handle, 2-4
  file number job, 1-5
Indexable file handle, 2-4
Indexable JFN, 1-5
Indexable job file number,
    1-5
Industry compatible mode,
    2-43
Inferior process,
  Creating an, 3-41
Info-returning JSYS's,
  CHKAC, 3-45
  DEVST, 3-99
  DIRST, 3-109
  DVCHR, 3-121
  GACCT, 3-143
  GACTF, 3-144
  GCVEC, 3-145
  GDSKC, 3-146
  GDSTS, 3-147
  GDVEC, 3-148
  GETAB, 3-152

Info-returning JSYS's
  (Cont.)
  GETER, 3-153
  GETJI, 3-154
  GETNM, 3-156
  GETOK%, 3-157
  GEVEC, 3-162
  GFRKH, 3-163
  GFUST, 3-166
  GJINF, 3-168
  GPJFN, 3-170
  GTAD, 3-171
  GTDAL, 3-172
  GTDIR, 3-173
  GTFDB, 3-175
  GTNCP, 3-195
  GTRPI, 3-194
  GTRPW, 3-197
  GTSTS, 3-198
  GTTYP, 3-199
  HPTIM, 3-202
  INLNM, 3-210
  JFNS, 3-211
  METER%, 3-220
  MSTR, 3-230
  PPNST, 3-318
  PRARG, 3-319
  RCDIR, 3-322
  RCM, 3-326
  RCUSR, 3-327
  RFACS, 3-338
  RFMOD, 3-341
  RFPOS, 3-343
  RFPTR, 3-344
  RFSTS, 3-346
  RFTAD, 3-349
  RIR, 3-352
  RIRCM, 3-353
  RPACS, 3-359
  RPCAP, 3-360
  RSMAP%, 3-363
  RTFRK, 3-364
  RTIW, 3-365
  RUNTM, 3-366
  RWM, 3-367
  SIZEF, 3-402
  STDEV, 3-438
  SYSGT, 3-451
  TMON, 3-470
  XGTPW%, 3-488
  XGVEC%, 3-489
Information,
  Monitor data base, 3-415,
    3-470
  Returning ARPANET host,
    3-176
  Returning directory,
    3-173
  Returning job, 3-154,
Information (Cont.)
    3-168
  Returning page trap,
    3-194
Information-obtaining,
  Functions, 2-24
Information-obtaining
    functions, 2-24
Initializing a process,
    3-337
Initiating software
    interrupts, 3-209
INLNM JSYS, 3-210
Input,
  Byte, 3-21, 3-303
  Double-precision, 3-100
  Dump, 3-117
  Floating-point, 3-141
  Random byte, 3-351
  Simulating terminal,
    3-439
  String, 3-396
Input buffer,
  Clearing file, 3-39
  Testing file, 3-395
Input designator,
  Primary, 3-332
Input file,
  Primary, 2-20
Inputting a number, 3-285
Inputting date/time, 3-205,
    3-207
Internal date/time,
  Converting, 3-294
  Converting to, 3-204
  Format, 1-9
Internal date/time format,
    1-9
Interrupt,
  Assigning terminal, 3-19
  Deassigning terminal,
    3-116
  Deferred terminal, 2-62
  Dismissing, 3-91
  Dismissing an, 2-62
  Functions software, 2-57
  Immediate terminal, 2-62
  Terminal, 2-60
Interrupt channel,
  Software, 2-57
Interrupt channel
    activation, 3-8
Interrupt channels,
  Deactivating, 3-107
Interrupt codes,
  Terminal, 2-60
Interrupt functions,
  Software, 2-57

Interrupt mask,
  Returning, 3-353, 3-367
  Setting, 3-401
Interrupt modes,
  Terminal, 2-62
Interrupt priority,
  Software, 2-58
Interrupt system,
  Clearing software, 3-47
  Disabling, 3-108
  Enabling software, 3-123
  Software, 2-57
Interrupt table,
  Returning, 3-352
  Software, 2-59
Interrupt table addresses,
  Obtaining, 3-490
  Setting, 3-400, 3-494
Interrupt word,
  Setting terminal, 3-440
  Terminal, 3-365
Interrupts,
  Initiating software,
    3-209
IPCF JSYS's,
  MRECV, 3-222
  MSEND, 3-224
  MUTIL, 3-279
IPCF logout message, 3-84
IPCF message,
  Retrieving an, 3-222
  Sending an, 3-224

JFN, 1-5, 3-169, 3-170,
    3-179, 3-187, 3-198,
    3-211
JFN,
  Indexable, 1-5
  Parse-only, 3-191, 3-212
  Releasing a, 3-354
  Restricted, 3-198, 3-445
  Setting primary, 3-429
  status word, 3-198
JFN mode word, 2-45, 3-341
JFN mode word,
  Returning, 3-341
JFN status word, 3-198
JFNs,
  Swapping, 3-447
JFNS JSYS, 3-211
Job,
  Attaching a, 3-17
  Creating a new, 3-81
  Detaching a, 3-115
  file number, 1-5
  Logging in a, 3-217
Job capabilities, 2-64
Job file number, 1-5
  Indexable, 1-5
Job indexable,
  file number, 1-5
Job information,
  Returning, 3-154, 3-168
Job parameters, 3-378
Job priority,
  Setting, 3-403
Job storage block, 3-361
Job's account,
  Returning, 3-143
Job-related JSYS's,
  ALLOC, 3-9
  ATACH, 3-17
  CACCT, 3-38
  CRJOB, 3-81
  DTACH, 3-115
  GACCT, 3-143
  GETJI, 3-154
  GETNM, 3-156
  GFRKS, 3-164
  GJINF, 3-168
  LGOUT, 3-215
  LOGIN, 3-217
  SETJB, 3-378
  SETNM, 3-381
  SETSN, 3-382
  SJPRI, 3-403
JSYS,
  ACCES, 3-2
  ADBRK, 3-5
  AIC, 3-8
  ALLOC, 3-9
  ARCF, 3-11
  Arguments, 1-1
  ASND, 3-15
  ASNSQ, 3-16
  ATACH, 3-17
  ATI, 3-19
  ATNVT, 3-20
  BKJFN, 3-22
  BOOT, 3-23
  BOUT, 3-37
  CACCT, 3-38
  CFBIF, 3-39
  CFBOF, 3-40
  CFORK, 3-41
  CHFDB, 3-43
  CHKAC, 3-45
  CIS, 3-47
  CLOSF, 3-48
  CLZFF, 3-50
  COMND, 3-52
  CRDIR, 3-75
  CRJOB, 3-81
  CRLNM, 3-87
  CVHST, 3-89
  CVSKT, 3-90

JSYS (Cont.)
DEBRK, 3-91
DELDF, 3-92
DELF, 3-94
DELNF, 3-96
DEQ, 3-97
DEVST, 3-99
DFIN, 3-100
DFOUT, 3-101
DIBE, 3-106
DIC, 3-107
DIRST, 3-109
DISMS, 3-110
DOBE, 3-111
DSKAS, 3-112
DSKOP, 3-113
DTACH, 3-115
DTI, 3-116
DUMPI, 2-41, 3-117
DUMPO, 3-119
DVCHR, 3-121
EIR, 3-123
ENQ, 3-124
ENQC, 3-130
EPCAP, 3-134
ERSTR, 3-135
ESOUT, 3-136
FFFFP, 3-137
FFORK, 3-138
FFUFP, 3-139
FLIN, 3-141
FLOUT, 3-142
GACCT, 3-143
GACTF, 3-144
GCVEC, 3-145
GDSKC, 3-146
GDSTS, 3-147
GDVEC, 3-148
GET, 3-149
GETAB, 3-152
GETER, 3-153
GETJI, 3-154
GETNM, 3-156
GETOK, 3-416
GETOK%, 3-157
GEVEC, 3-162
GFRKH, 3-163
GFRKS, 3-164
GFUST, 3-166
GIVOK%, 3-167
GJINF, 3-168
GNJFN, 3-169
GPJFN, 3-170
GTAD, 3-171
GTDAL, 3-172
GTDIR, 3-173
GTFDB, 3-175
GTHST%, 3-176
GTJFN, 3-179, 3-187

JSYS (Cont.)
GTRPI, 3-194
GTRPW, 3-197
GTSTS, 3-198
GTTYP, 3-199
HALTF, 3-200
HFORK, 3-201
HPTIM, 3-202
HSYS, 3-203
IDCNV, 3-204
IDTIM, 3-205
IDTNC, 3-207
IIC, 3-209
INLNM, 3-210
JFNS, 3-211
KFORK, 3-214
LGOUT, 3-215
LNMST, 3-216
LOGIN, 3-217
LPINI, 3-218
MDDT%, 3-219
METER%, 3-220
MRECV, 3-222
MSEND, 3-224
MSFRK, 3-229
MSTR, 3-230
MTALN, 3-247
MTOPR, 3-248
MTU%, 3-277
MUTIL, 3-279
NIN, 3-285
NODE, 3-286
NOUT, 3-292.1
NTMAN%, 3-292.2
ODCNV, 3-294
ODTIM, 3-295
ODTNC, 3-297
OPENF, 3-298
PBIN, 3-303
PBOUT, 3-304
PEEK, 3-308
PLOCK, 3-309
PMAP, 3-310
PMCTL, 3-315
PPNST, 3-318
PRARG, 3-319
PSOUT, 3-321
RCDIR, 3-322
RCM, 3-326
RCUSR, 3-327
RCVIM, 3-329
RCVOK%, 3-330
RDTTY, 3-332
RELD, 3-335
RELSQ, 3-336
RESET, 3-337
RFACS, 3-338
RFBSZ, 3-339
RFCOC, 3-340
RFMOD, 3-341

JSYS (Cont.)
RFORK, 3-342
RFPOS, 3-343
RFPTR, 3-344
RFRKH, 3-345
RFSTS, 3-346
RFTAD, 3-349
RIN, 3-351
RIR, 3-352
RIRCM, 3-353
RLJFN, 3-354
RMAP, 3-355
RNAMF, 3-356
ROUT, 3-358
RPACS, 3-359
RPCAP, 3-360
RSCAN, 3-361
RSMAP%, 3-363
RTFRK, 3-364
RTIW, 3-365
RUNTM, 3-366
RWM, 3-367
RWSET, 3-368
SACTF, 3-369
SAVE, 3-370
SCTTY, 3-371
SCVEC, 3-373
SDSTS, 3-375
SDVEC, 3-376
SETER, 3-377
SETJB, 3-378
SETNM, 3-381
SETSN, 3-382
SEVEC, 3-383
SFACS, 3-384
SFBSZ, 3-385
SFCOC, 3-386
SFMOD, 3-387
SFORK, 3-388
SFPOS, 3-389
SFPTR, 3-390
SFRKV, 3-391
SFTAD, 3-392
SFUST, 3-394
SIBE, 3-395
SIN, 3-396
SINR, 3-398
SIR, 3-400
SIRCM, 3-401
SIZEF, 3-402
SJPRI, 3-403
SKED%, 3-404
SKPIR, 3-409
SMAP%, 3-410
SMON, 3-415
SNDIM, 3-417
SNOOP, 3-418
SOBE, 3-422
SOBF, 3-423

JSYS (Cont.)
SOUT, 3-424
SOUTR, 3-426
SPACS, 3-428
SPJFN, 3-429
SPLFK, 3-430
SPOOL, 3-431
SPRIW, 3-433
SSAVE, 3-434
STAD, 3-436
STCMP, 3-437
STDEV, 1-8, 3-438
STI, 3-439
STIW, 3-440
STO, 3-442
STPAR, 3-443
STPPN, 3-444
STSTS, 3-445
STTYP, 3-446
SWJFN, 3-447
SWTRP%, 3-448
SYERR, 3-450
SYSGT, 3-451
TBADD, 3-452
TBDEL, 3-453
TBLUK, 3-454
TEXTI, 3-457
TFORK, 3-461
THIBR, 3-464
TIME, 3-465
TIMER, 3-466
TLINK, 3-468
TMON, 3-470
TTMSG, 3-472
TWAKE, 3-473
UFPGS, 3-474
USAGE, 3-475
USRIO, 3-478
UTEST, 3-479
UTFRK, 3-481
VACCT, 3-482
WAIT, 3-483
WFORK, 3-484
WILD%, 3-485
XGTPW%, 3-488
XGVEC%, 3-489
XRIR%, 3-490
XRMAP%, 3-491
XSFRK%, 3-493
XSIR%, 3-494
XSVEC%, 3-496
JSYS arguments, 1-1, 1-2
JSYS error codes, 1-10
JSYS errors, 1-10
JSYS return, 1-1, 1-10

KFORK JSYS, 3-214
Killing a process, 3-214

Length,
  Terminal, 2-46, 2-47
LGOUT JSYS, 3-215
Limit,
  End-of-file, 2-21
  EOF, 2-21
Line printer functions, 2-36
Link,
  DECnet logical, 3-254
Linking,
  Terminal, 2-53, 3-468
LNMST JSYS, 3-216
Loading VFU, 3-218
Local,
  byte pointer, 1-4
Local byte pointer, 1-4
  two-word, 1-4
Local time, 1-10
Logging in a job, 3-217
Logical,
  Magnetic tape, 2-45
  Name, 3-87
Logical link,
  DECnet, 3-254
Logical magnetic tape, 2-45
Logical name, 3-87
  Creating a, 3-87
  Translating a, 3-216
Logical names, 2-2
Logical-name JSYS's,
  CRLNM, 3-87
  FOO, 3-216
  INLNM, 3-210
LOGIN JSYS, 3-217
Logout message,
  IPCF, 3-84
Long form GTJFN, 3-187
LPINI JSYS, 3-218
LPT:, 2-33, 2-38, 2-55, 2-56

Magnetic tape,
  Functions, 2-39
  Logical, 2-45
  Physical, 2-45
  Status bits, 2-44
Magnetic tape functions, 2-39
Magnetic tape logical, 2-45
Magnetic tape physical, 2-45
Magnetic tape status bits, 2-44
Manipulating a spooled device, 3-431
Manual pointer conventions, 1-7
Manual references, 1
Mapping,
  Page, 3-310, 3-311, 3-312, 3-313
  Section, 3-363
Mapping a section, 3-410
Mapping memory, 3-363, 3-410
Mask,
  Returning interrupt, 3-353, 3-367
  Setting interrupt, 3-401
MDDT,
  Entering, 3-219
MDDT% JSYS, 3-219
Memory, 3-410
  Acquiring physical, 3-309
  Mapping, 3-363, 3-410
Message,
  IPCF logout, 3-84
  Retrieving an IPCF, 3-222
  Sending an IPCF, 3-224
Messages,
  Error, 2-24
Meter values,
  Returning EBOX/MBOX, 3-220
METER% JSYS, 3-220
Mode,
  ANSI ASCII, 2-43
  Dump, 2-43
  Echo, 2-46, 2-47
  Full duplex, 2-46, 2-48
  Half duplex, 2-46, 2-48
  High density, 2-44
  Industry compatible, 2-43
  Setting device, 3-443
  SIXBIT, 2-44
  Terminal data, 2-46, 2-47
  User-I/O, 3-478
Mode word,
  JFN, 2-45, 3-341
  Returning JFN, 3-341
Modes,
  Access, 2-8
  Hardware data, 2-42
  I/O, 2-54
  Setting terminal, 3-387
  Software data, 2-54, 2-56, 2-57
  Terminal interrupt, 2-62
Modes access, 2-8
Modes I/O, 2-54

Modifying the FDB, 2-10
Monitor calls,
  Privileged, 2-82
Monitor data base
    information, 3-415,
    3-470
Monitor flags,
  Setting, 3-415
  Testing, 3-470
Most recent error,
  Returning, 3-153
Mountable-structure
    functions, 3-230
MRECV JSYS, 3-222
MSEND JSYS, 3-224
MSFRK JSYS, 3-229
.MSSSS, 3-230
MSTR JSYS, 3-230
MT device functions, 3-277
MT:, 2-33, 2-45, 2-56
MTA:, 2-33, 2-39, 2-56
MTA: status bits, 2-39
MTALN JSYS, 3-247
MTOPR JSYS, 3-248
MTU% JSYS, 3-277
MUTIL JSYS, 3-279

Name,
  Creating a logical, 3-87
  Logical, 3-87
  Private program, 3-381,
    3-382
  Returning program, 3-156
  Setting program, 3-381,
    3-382
  System program, 3-382
  Translating a logical,
    3-216
Name logical, 3-87
Name string,
  Translating device, 3-438
  Translating directory,
    3-444
Names,
  Logical, 2-2
NET:, 2-33, 2-56
Network functions, 3-286, 3-292.2
Network terminal, 3-254
New job,
  Creating a, 3-81
NIN JSYS, 3-285
NODE JSYS, 3-286
Nonsharable save file, 2-70
Nonshareable save, 3-370
NOUT JSYS, 3-292
NUL:, 2-33, 2-56
Number,
  Converting host, 3-89
  Converting socket, 3-90
  Indexable job file, 1-5
  Inputting a, 3-285
  job file, 1-5
  Outputting a, 3-292
  Section-relative page,
    1-3
  Setting terminal, 3-446
  Translating directory,
    3-109
Number job,
  file, 1-5
Number job indexable,
  file, 1-5
Numbers,
  Global page, 1-3
Numeric-I/O JSYS's,
  COMND, 3-52
  DFIN, 3-100
  DFOUT, 3-101
  FLIN, 3-141
  FLOUT, 3-142
  NIN, 3-285
  NOUT, 3-292
NVT connection,
  Creating, 3-20

Obtaining interrupt table
    addresses, 3-490
ODCNV JSYS, 3-294
ODTIM JSYS, 3-295
ODTNC JSYS, 3-297
Offline expiration date and
    time, 3-349
Offset,
  Timezone, 1-10
One-word global,
  byte pointer, 1-4
One-word global byte
    pointer, 1-4
Online expiration date and
    time, 3-349
OPENF JSYS, 3-298
Opening,
  Files, 3-298
Opening a,
  File, 2-5
Opening a device, 2-5
Opening a file, 2-5, 3-298
Opening files, 3-298
Operations,
  Process, 2-67, 3-430,
    3-473, 3-484

Output,
  Byte, 3-37, 3-304
  Double-precision, 3-101
  Dump, 3-119
  Floating-point, 3-142
  Random byte, 3-358
  Simulating terminal,
    3-442
  String, 3-321, 3-424
Output buffer,
  Clearing file, 3-40
  Testing file, 3-422,
    3-423
Output designator,
  Primary, 3-332
Output file,
  Primary, 2-20
Outputting a number, 3-292
Outputting date/time, 3-295,
    3-297
Outputting error strings,
    3-136
Overflow trapping, 3-448

PA1050, 2-70
PA1050 entry vector,
  Returning, 3-145
Package,
  Compatibility, 2-70,
    3-373
Page,
  Finding first free file,
    3-137
  Finding first used file,
    3-139
  Handle, 3-355
Page access, 3-355, 3-491
Page accessibility,
  Setting, 3-428
Page handle, 3-355
Page mapping, 3-310, 3-311,
    3-312, 3-313
Page number,
  Section-relative, 1-3
Page numbers,
  Global, 1-3
Page trap information,
  Returning, 3-194
Page-related JSYS's,
  FFFFP, 3-137
  FFUFP, 3-139
  FOO, 3-428
  GET, 3-149
  GTRPI, 3-194, 3-309
  PMAP, 3-310
  PMCTL, 3-315
  RMAP, 3-355
Page-related JSYS's (Cont.)
  RPACS, 3-359
  RWSET, 3-368
  SAVE, 3-370
  UFPGS, 3-474
  XRMAP%, 3-491
Pages,
  Updating file, 3-474
Panic channels, 2-58, 2-60
Parameter-reading JSYS's,
  FOO, 3-198, 3-199
  PRARG, 3-319
  TMON, 3-470
Parameter-setting JSYS's,
  CACCT, 3-38
  CHFDB, 3-43
  EPCAP, 3-134
  PRARG, 3-319
  SETJB, 3-378
  SMON, 3-415
Parameters,
  Job, 3-378
Parse-only file
    specification, 2-4
Parse-only JFN, 3-191,
    3-212
Parsing,
  Command, 3-52
PBIN JSYS, 3-303
PBOUT JSYS, 3-304
PC histogram, 3-418
PCDP, 2-33
PCDP:, 2-33, 2-34, 2-35,
    2-54
PCDP: status bits, 2-34
PCDR:, 2-33, 2-55
PCDR: status bits, 2-33
PEEK JSYS, 3-308
Performance analysis,
  System, 3-418
Physical,
  Magnetic tape, 2-45
Physical magnetic tape,
    2-45
Physical memory,
  Acquiring, 3-309
Physical/logical tape-drive
    association, 3-247
PLOCK JSYS, 3-309
PLPT:, 2-33, 2-36, 2-37,
    2-55
PLPT: control characters,
    2-37
PLPT: status bits, 2-37
PMAP JSYS, 3-310
PMCTL JSYS, 3-315
Pointer,
  Backing up, 3-22
  byte, 1-4, 1-6

Pointer (Cont.)
  Byte, 1-7
  local byte, 1-4
  one-word global byte, 1-4
  Setting file, 3-390
  Setting terminal, 3-389
Pointer conventions,
  Manual, 1-7
Pointer local,
  byte, 1-4
Pointer one-word global,
  byte, 1-4
PPN, 3-318, 3-444
PPNST JSYS, 3-318
PRARG JSYS, 3-319
Primary input designator,
    3-332
Primary input file, 2-20
Primary JFN,
  Setting, 3-429
Primary output designator,
    3-332
Primary output file, 2-20
Printer,
  Functions line, 2-36
Priority,
  Setting job, 3-403
  Setting process, 3-433
  Software interrupt, 2-58
Priority control word,
  Scheduler, 3-403
Private program name, 3-381,
    3-382
Privileged,
  Functions, 2-82
Privileged functions, 2-82
Privileged monitor calls,
    2-82
Process,
  Creating an inferior,
    3-41
  DECnet remote, 3-254
  Dismissing a, 3-106,
    3-110, 3-111, 3-483
  Freezing a, 3-138
  Frozen, 3-342
  Halting a, 3-200, 3-201
  Initializing a, 3-337
  Killing a, 3-214
  Resetting a, 3-337
  Resuming a, 3-342
  Splicing a, 3-430
  Starting a, 3-229, 3-388,
    3-391, 3-493
  Waking a, 3-473
Process AC's,
  Returning, 3-338
  Setting, 3-384
Process blocking,
  Elapsed time, 3-464
Process capabilities, 2-63,
    2-64
Process entry vector,
  Returning, 3-162, 3-489
  Setting, 3-383, 3-496
Process execution,
  Resuming, 3-481
Process files,
  Closing, 3-50
Process handle, 1-9
  Relative, 1-9
  Releasing a, 3-345
Process handle relative,
    1-9
Process operations, 2-67,
    3-430, 3-473, 3-484
Process priority,
  Setting, 3-433
Process status, 3-347
  Returning, 3-346
Process termination,
  Waiting for, 3-484
Process timing, 3-466
Process-control,
  Functions, 2-63
Process-control functions,
    2-63
Process-controling,
  Functions, 2-67
Process-controling
    functions, 2-67
Process-related JSYS's,
  CFORK, 3-41
  CIS, 3-47
  CLZFF, 3-50
  DIBE, 3-106
  DISMS, 3-110
  DOBE, 3-111
  EIR, 3-123
  EPCAP, 3-134
  FFORK, 3-138
  GET, 3-149
  GEVEC, 3-162
  GFRKH, 3-163
  GFRKS, 3-164
  GPJFN, 3-170
  GTRPI, 3-194, 3-309
  HALTF, 3-200
  HFORK, 3-201
  IIC, 3-209
  KFORK, 3-214
  MRECV, 3-222
  MSEND, 3-224
  MSFRK, 3-229
  PMAP, 3-310
  PRARG, 3-319

Process-related JSYS's
  (Cont.)
  RESET, 3-337
  RFACS, 3-338
  RFORK, 3-342
  RFRKH, 3-345
  RFSTS, 3-346
  RMAP, 3-355
  RSMAP%, 3-363
  RTFRK, 3-364
  RWM, 3-367
  SCTTY, 3-371
  SETER, 3-377
  SEVEC, 3-383
  SFACS, 3-384
  SFORK, 3-388
  SFRKV, 3-391
  SIR, 3-400
  SIRCM, 3-401
  SKPIR, 3-409
  SPJFN, 3-429
  SPLFK, 3-430
  SPRIW, 3-433
  SSAVE, 3-434
  TFORK, 3-461
  THIBR, 3-464
  TIMER, 3-466
  TWAKE, 3-473
  UTFRK, 3-481
  WAIT, 3-483
  WFORK, 3-484
  XGVEC%, 3-489
  XRIR%, 3-490
  XRMAP%, 3-491
  XSFRK%, 3-493
  XSIR%, 3-494
  XSVEC%, 3-496
Process/file,
  handle, 1-9
Process/file handle, 1-9
Processes,
  Execute-only, 2-68
Program,
  Access-control, 3-167,
    3-416
  control, 1-10
  Sample, 2-5
Program control, 1-10
Program data vector, 3-305
Program debugging, 3-5
Program name,
  Private, 3-381, 3-382
  Returning, 3-156
  Setting, 3-381, 3-382
  System, 3-382
Project-programmer number
    (PPN), 3-318, 3-444
PSI,
  Functions, 2-57
PSI functions, 2-57
PSOUT JSYS, 3-321
PTY:, 2-33

Queue,
  Assigning ARPANET, 3-16

Random byte input, 3-351
Random byte output, 3-358
Random-I/O JSYS's,
  RIN, 3-351, 3-358
RCDIR JSYS, 3-322
RCM JSYS, 3-326
RCUSR JSYS, 3-327
RCVIM JSYS, 3-329
RCVOK% JSYS, 3-330
RDTTY JSYS, 3-332
Read access, 2-8, 3-298
Reading the FDB, 2-10
Recent error,
  Returning most, 3-153
Recognition,
  File, 2-2
Recognition file, 2-2
Record-I/O JSYS's,
  SINR, 3-398
  SOUTR, 3-426
Redefining controlling
    terminal, 3-371
References,
  Manual, 1
Regulated structure, 2-5,
    3-2, 3-230, 3-246,
    3-475
Relative,
  Process handle, 1-9
Relative process handle,
    1-9
RELD JSYS, 3-335
Releasing a JFN, 3-354
Releasing a process handle,
    3-345
Releasing devices, 3-335
Releasing working set,
    3-368
Relinquishing directory
    access, 3-2
RELSQ JSYS, 3-336
Remote process,
  DECnet, 3-254
Renaming a file, 3-356
Rescan buffer, 3-361
Reserving a channel, 3-102
RESET JSYS, 3-337
Resetting a process, 3-337

Resetting file byte size,
    3-385
Restart time,
  Returning elapsed system,
    3-465
Restricted JFN, 3-198,
    3-445
Resuming a process, 3-342
Resuming process execution,
    3-481
Retrieving an IPCF message,
    3-222
Return,
  +1, 1-10
  +2, 1-10
  Error, 1-10
  JSYS, 1-1, 1-10
Return error, 1-10
Returning ARPANET host
    information, 3-176
Returning CCOC words, 3-340
Returning device
    characteristics, 3-121
Returning device status,
    3-147
Returning directory
    information, 3-173
Returning disk allocation,
    3-172
Returning disk usage, 3-146
Returning EBOX/MBOX meter
    values, 3-220
Returning elapsed system
    restart time, 3-465
Returning file author,
    3-166
Returning file byte-size,
    3-339
Returning file descriptor
    block, 3-175
Returning file
    specification, 3-211
Returning file status,
    3-198
Returning file's account,
    3-144
Returning high-precision
    clock, 3-202
Returning interrupt mask,
    3-353, 3-367
Returning interrupt table,
    3-352
Returning JFN mode word,
    3-341
Returning job information,
    3-154, 3-168
Returning job's account,
    3-143
Returning most recent error,
    3-153
Returning PA1050 entry
    vector, 3-145
Returning page trap
    information, 3-194
Returning process AC's,
    3-338
Returning process entry
    vector, 3-162, 3-489
Returning process status,
    3-346
Returning program name,
    3-156
Returning RMS entry vector,
    3-148
Returning system table,
    3-152, 3-451
Returning terminal type,
    3-199
Returning trap words, 3-197,
    3-488
RFACS JSYS, 3-338
RFBSZ JSYS, 3-339
RFCOC JSYS, 3-340
RFMOD JSYS, 3-341
RFORK JSYS, 3-342
RFPOS JSYS, 3-343
RFPTR JSYS, 3-344
RFRKH JSYS, 3-345
RFSTS JSYS, 3-346
RFTAD JSYS, 3-349
RIN JSYS, 3-351
RIR JSYS, 3-352
RIRCM JSYS, 3-353
RLJFN JSYS, 3-354
RMAP JSYS, 3-355
RMS entry vector, 3-376
  Returning, 3-148
RNAMF JSYS, 3-356
ROUT JSYS, 3-358
RPACS JSYS, 3-359
RPCAP JSYS, 3-360
RSCAN JSYS, 3-361
RSMAP% JSYS, 3-363
RTFRK JSYS, 3-364
RTIW JSYS, 3-365
Run time, 3-366
RUNTM JSYS, 3-366
RWM JSYS, 3-367
RWSET JSYS, 3-368

SACTF JSYS, 3-369
Sample program, 2-5
Save,
  Nonshareable, 3-370

Save (Cont.)
  Sharable, 3-434
Save file,
  Getting a, 3-149
  Nonsharable, 2-70
  Sharable, 2-71
Save files, 2-70
SAVE JSYS, 3-370
Scheduler control, 3-404
Scheduler priority control word, 3-403
SCTTY JSYS, 3-371
SCVEC JSYS, 3-373
SDSTS JSYS, 3-375
SDVEC JSYS, 3-376
Searching,
  Table, 3-454
Section,
  Creating a, 3-410
  current, 1-3
  Handle, 3-363
  Mapping a, 3-410
Section handle, 3-363
Section mapping, 3-363
Section-relative,
  address, 1-3
Section-relative address, 1-3
Section-relative page number, 1-3
Sections,
  Creating, 3-363
Sending an IPCF message, 3-224
Set,
  Releasing working, 3-368
SETER JSYS, 3-377
SETJB JSYS, 3-378
SETNM JSYS, 3-381
SETSN JSYS, 3-382
Setting CCOC words, 3-386
Setting device mode, 3-443
Setting error condition, 3-377
Setting file author, 3-394
Setting file date/time, 3-392
Setting file pointer, 3-390
Setting file status, 3-445
Setting interrupt mask, 3-401
Setting interrupt table addresses, 3-400, 3-494
Setting job priority, 3-403
Setting monitor flags, 3-415
Setting page accessibility, 3-428
Setting primary JFN, 3-429
Setting process AC's, 3-384
Setting process entry vector, 3-383, 3-496
Setting process priority, 3-433
Setting program name, 3-381, 3-382
Setting system date, 3-436
Setting terminal interrupt word, 3-440
Setting terminal modes, 3-387
Setting terminal number, 3-446
Setting terminal pointer, 3-389
SEVEC JSYS, 3-383
SFACS JSYS, 3-384
SFBSZ JSYS, 3-385
SFCOC JSYS, 3-386
SFMOD JSYS, 3-387
SFORK JSYS, 3-388
SFPOS JSYS, 3-389
SFPTR JSYS, 3-390
SFRKV JSYS, 3-391
SFTAD JSYS, 3-392
SFUST JSYS, 3-394
Sharable save, 3-434
Sharable save file, 2-71
Short form GTJFN, 3-179
SIBE JSYS, 3-395
Simulating terminal input, 3-439
Simulating terminal output, 3-442
SIN JSYS, 3-396
SINR JSYS, 3-398
SIR JSYS, 3-400
SIRCM JSYS, 3-401
SIXBIT mode, 2-44
Size,
  Resetting file byte, 3-385
SIZEF JSYS, 3-402
SJPRI JSYS, 3-403
SKED% JSYS, 3-404
SKPIR JSYS, 3-409
SMAP% JSYS, 3-410
SMON JSYS, 3-415
SNDIM JSYS, 3-417
SNOOP JSYS, 3-418
SOBE JSYS, 3-422
SOBF JSYS, 3-423
Socket number,
  Converting, 3-90
Software data modes, 2-54, 2-56, 2-57
Software interrupt,
  Functions, 2-57

Software interrupt channel,
    2-57
Software interrupt
    functions, 2-57
Software interrupt priority,
    2-58
Software interrupt system,
    2-57
  Clearing, 3-47
  Enabling, 3-123
Software interrupt table,
    2-59
Software interrupts,
  Initiating, 3-209
Software-interrupt JSYS's,
  AIC, 3-8
  ATI, 3-19
  CIS, 3-47
  DEBRK, 3-91
  DIBE, 3-106
  DIC, 3-107
  DIR, 3-108
  DOBE, 3-111
  DTI, 3-116
  EIR, 3-123
  IIC, 3-209
  RCM, 3-326
  RIR, 3-352
  RIRCM, 3-353
  RWM, 3-367
  SIR, 3-400
  SIRCM, 3-401
  SKPIR, 3-409
  STIW, 3-440
  XRIR%, 3-490
  XSIR%, 3-494
Source,
  Designator, 1-6
Source designator, 1-6
Source/destination
    designator, 1-6
SOUT JSYS, 3-424
SOUTR JSYS, 3-426
SPACS JSYS, 3-428
Specification,
  File, 2-1
  Parse-only file, 2-4
  Returning file, 3-211
SPJFN JSYS, 3-429
SPLFK JSYS, 3-430
Splicing a process, 3-430
SPOOL JSYS, 3-431
Spooled device,
  Manipulating a, 3-431
SPRIW JSYS, 3-433
SRV:, 2-57
SSAVE JSYS, 3-434
STAD JSYS, 3-436
Standard,
  Date/time, 1-9
Standard date/time, 1-9
Starting a process, 3-229,
    3-388, 3-391, 3-493
Status,
  Process, 3-347
  Returning device, 3-147
  Returning file, 3-198
  Returning process, 3-346
  Setting file, 3-445
Status bits,
  Magnetic tape, 2-44
  MTA:, 2-39
  PCDP:, 2-34
  PCDR:, 2-33
  PLPT:, 2-37
  Terminal, 2-45
  TTY:, 2-45
Status bits magnetic tape,
    2-44
Status bits terminal, 2-45
Status word,
  JFN, 3-198
Status word JFN, 3-198
STCMP JSYS, 3-437
STDEV JSYS, 1-8, 3-438
STI JSYS, 3-439
STIW JSYS, 3-440
STO JSYS, 3-442
Storage block,
  Job, 3-361
STPAR JSYS, 3-443
STPPN JSYS, 3-444
String,
  Translating device, 3-99
  Translating device name,
    3-438
  Translating directory
    name, 3-444
String comparison,
  Wild, 3-485
String input, 3-396
String output, 3-321, 3-424
String-compare JSYS's,
  STCMP, 3-437
  WILD%, 3-485
String-I/O JSYS's,
  PSOUT, 3-321
  SIN, 3-396
  SOUT, 3-424
Strings, 1-7
  ASCII, 1-7
  Comparing, 3-437, 3-485
  Error, 2-24
  Outputting error, 3-136
  Translating error, 3-135

Structure,
  Regulated, 2-5, 3-2,
    3-230, 3-246, 3-475
Structure-related JSYS's,
  ACCES, 3-2
  GNJFN, 3-169
  MSTR, 3-230
STSTS JSYS, 3-445
STTYP JSYS, 3-446
Swapping JFNs, 3-447
SWJFN JSYS, 3-447
SWTRP% JSYS, 3-448
SYERR JSYS, 3-450
SYSGT JSYS, 3-451
SYSTAT table, 2-29
System,
  Archive/virtual disk,
    2-81
  Clearing software
    interrupt, 3-47
  date, 1-9
  Disabling interrupt,
    3-108
  Enabling software
    interrupt, 3-123
  Halting, 3-203
  Software interrupt, 2-57
System accounting, 3-475
System date, 1-9, 2-79
System date,
  Setting, 3-436
System error file, 3-450
System performance analysis,
    3-418
System program name, 3-382
System restart time,
  Returning elapsed, 3-465
System table,
  Returning, 3-152, 3-451
System tables, 2-24

Table,
  Returning interrupt,
    3-352
  Returning system, 3-152,
    3-451
  Software interrupt, 2-59
  SYSTAT, 2-29
Table addresses,
  Obtaining interrupt,
    3-490
  Setting interrupt, 3-400,
    3-494
Table entry,
  Adding a, 3-452
  Deleting a, 3-453
Table searching, 3-454
Table-lookup JSYS's,
  TBADD, 3-452
  TBDEL, 3-453
  TBLUK, 3-454
Tables,
  System, 2-24
Tape,
  Carriage control, 2-36
  Functions magnetic, 2-39
  Logical magnetic, 2-45
  Physical magnetic, 2-45
  Status bits magnetic,
    2-44
Tape functions,
  Magnetic, 2-39
Tape logical,
  Magnetic, 2-45
Tape physical,
  Magnetic, 2-45
Tape status bits,
  Magnetic, 2-44
Tape-drive association,
  Physical/logical, 3-247
TBADD JSYS, 3-452
TBDEL JSYS, 3-453
TBLUK JSYS, 3-454
Terminal,
  Designator, 1-6
  Functions, 2-45
  Network, 3-254
  Redefining controlling,
    3-371
  Status bits, 2-45
Terminal advising, 2-53
Terminal data mode, 2-46,
    2-47
Terminal designator, 1-6,
    1-8
Terminal functions, 2-45
Terminal input,
  Simulating, 3-439
Terminal interrupt, 2-60
  Assigning, 3-19
  Deassigning, 3-116
  Deferred, 2-62
  Immediate, 2-62
Terminal interrupt codes,
    2-60
Terminal interrupt modes,
    2-62
Terminal interrupt word,
    3-365
  Setting, 3-440
Terminal length, 2-46, 2-47
Terminal linking, 2-53,
    3-468
Terminal modes,
  Setting, 3-387

Terminal number,
  Setting, 3-446
Terminal output,
  Simulating, 3-442
Terminal pointer,
  Setting, 3-389
Terminal status bits, 2-45
Terminal type,
  Returning, 3-199
Terminal width, 2-46, 2-47
Terminal-related JSYS's,
  ATACH, 3-17
  ATI, 3-19
  BKJFN, 3-22
  CFBIF, 3-39
  CFBOF, 3-40
  DTACH, 3-115
  DTI, 3-116
  GJINF, 3-168
  GTTYP, 3-199
  RDTTY, 3-332
  RFCOC, 3-340
  RFPOS, 3-343
  RSCAN, 3-361
  RTIW, 3-365
  SCTTY, 3-371
  SFCOC, 3-386
  SFMOD, 3-387
  SFPOS, 3-389
  STI, 3-439
  STIW, 3-440
  STO, 3-442
  STTYP, 3-446
  TEXTI, 3-457
  TLINK, 3-468
  TTMSG, 3-472
Termination,
  Waiting for process,
    3-484
Testing file input buffer,
    3-395
Testing file output buffer,
    3-422, 3-423
Testing for end-of-file,
    2-22
Testing monitor flags,
    3-470
TEXTI JSYS, 3-457
TFORK JSYS, 3-461
Thawed access, 2-8, 3-298
THIBR JSYS, 3-464
Time,
  Greenwich Mean, 1-10
  Local, 1-10
  Offline expiration date
    and, 3-349
  Online expiration date
    and, 3-349
  Returning elapsed system
Time (Cont.)
    restart, 3-465
  Run, 3-366
TIME JSYS, 3-465
Time process blocking,
  Elapsed, 3-464
Time zone, 1-10
TIMER JSYS, 3-466
Timezone offset, 1-10
Timing,
  Process, 3-466
TLINK JSYS, 3-468
TMON JSYS, 3-470
TOPS-10 monitor calls, 1-1
Translating a logical name,
    3-216
Translating device name
    string, 3-438
Translating device string,
    3-99
Translating directory name
    string, 3-444
Translating directory
    number, 3-109
Translating error strings,
    3-135
Trap information,
  Returning page, 3-194
Trap words,
  Returning, 3-197, 3-488
Trap-related JSYS's,
  GTRPI, 3-194
  GTRPW, 3-197
  XGTPW%, 3-488
Trapping,
  Overflow, 3-448
  Underflow, 3-448
TTMSG JSYS, 3-472
TTY-I/O JSYS's,
  BIN, 3-21
  BOUT, 3-37
  CFBIF, 3-39
  CFBOF, 3-40
  COMND, 3-52
  DFIN, 3-100
  DFOUT, 3-101
  FLIN, 3-141
  FLOUT, 3-142
  IDTIM, 3-176, 3-205
  IDTNC, 3-207
  NIN, 3-285
  NOUT, 3-292
  PBIN, 3-303
  PBOUT, 3-304
  RDTTY, 3-332
  RFCOC, 3-340
  RFPOS, 3-343
  SFCOC, 3-386
  SFMOD, 3-387

TTY-I/O JSYS's (Cont.)
  SFPOS, 3-389
  SIN, 3-396
  SINR, 3-398
  SOUT, 3-424
  STI, 3-439
  STO, 3-442
  TEXTI, 3-457
  TTMSG, 3-472
TTY:, 2-33, 2-45, 2-57
TTY: characteristics, 2-51, 2-52
TTY: status bits, 2-45
TWAKE JSYS, 3-473
Two-word global byte pointer, 1-4
Two-word local byte pointer, 1-4
Type,
  Returning terminal, 3-199

UFPGS JSYS, 3-474
Unbuffered I/O, 2-41
Underflow trapping, 3-448
Universal default designator, 1-8
Up pointer,
  Backing, 3-22
Updating file pages, 3-474
Usage,
  Returning disk, 3-146
USAGE JSYS, 3-475
Used file page,
  Finding first, 3-139
User-I/O mode, 3-478
USRIO JSYS, 3-478
UTEST JSYS, 3-479
UTFRK JSYS, 3-481
UUO's, 1-1

VACCT JSYS, 3-482
Values,
  Returning EBOX/MBOX meter, 3-220
Vector,
  Compatibility package entry, 3-373
  Entry, 2-74
  Program data, 3-305
  Returning PA1050 entry, 3-145
  Returning process entry, 3-162, 3-489
  Returning RMS entry, 3-148
  RMS entry, 3-376
  Setting process entry, 3-383, 3-496
Verifying accounts, 3-482
VFU, 2-36
  Loading, 3-218

WAIT JSYS, 3-483
Waiting for process termination, 3-484
Wakeup class, 2-48, 2-49, 2-50
Waking a process, 3-473
WFORK JSYS, 3-484
Width,
  Terminal, 2-46, 2-47
Wild string comparison, 3-485
WILD% JSYS, 3-485
Wildcard characters, 2-3
Word,
  CCOC, 2-48, 2-49
  JFN mode, 2-45, 3-341
  JFN status, 3-198
  Returning JFN mode, 3-341
  Scheduler priority control, 3-403
  Setting terminal interrupt, 3-440
  Terminal interrupt, 3-365
Word JFN,
  status, 3-198
Words,
  CCOC, 3-340, 3-386
  Returning CCOC, 3-340
  Returning trap, 3-197, 3-488
  Setting CCOC, 3-386
Working set,
  Releasing, 3-368
Write access, 2-8, 3-298

XGTPW% JSYS, 3-488
XGVEC% JSYS, 3-489
XRIR% JSYS, 3-490
XRMAP% JSYS, 3-491
XSFRK% JSYS, 3-493
XSIR% JSYS, 3-494
XSVEC% JSYS, 3-496

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ____________________

Name ____________________ Date ____________________

Organization ____________________ Telephone ____________________

Street ____________________

City ____________________ State ____________ Zip Code or Country ____________

--- Do Not Tear – Fold Here and Tape ---

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MRO1-2/L12
MARLBOROUGH, MA 01752

-- Do Not Tear – Fold Here and Tape ---