

Table of contents

2-	1	Branch vector for processing routines
4-	1	EMT entry processing
5-	1	Shared file control EMT's
6-	1	PLAS EMT's
7-	1	Message communication EMT's
8-	1	.WRITE
9-	1	.READ
10-	1	.SERR
10-	7	.HERR
11-	1	.SPFUN
12-	1	.SCCA
12-	11	.TRPSET
12-	20	.SFPA
12-	36	.QSET
12-	45	.SETTOP
13-	1	.GTJB
14-	1	.CHAIN
14-	17	.RCTRLO
15-	1	.HRESET
15-	9	.SRESET
16-	1	.SAVESTATUS
17-	1	.REOPEN
18-	1	.PURGE
19-	1	.WAIT
20-	1	.GTIM
20-	13	.DATE
20-	23	.SDTTM
21-	1	.TWAIT
22-	1	.MRKT
23-	1	.CMKT
25-	1	.CSTAT
25-	48	.CDFN
26-	1	.GVAL
28-	1	.EXIT
29-	1	TTEMT -- Terminal control EMT
31-	1	SNDMSG -- Send a message to a line
32-	1	GTMSBF -- Get free system message buffer
33-	1	QMSG -- Queue a message for a job
34-	1	ASTXIT -- Exit from completion routine
35-	1	.SPCPS -- Alter exit address from a completion routine
36-	1	EMT376 -- EMT 376 Processing
37-	1	JBINFO -- Get information about a specific job
39-	1	Get terminal type code
39-	39	Get Project-Programmer number
40-	1	MISC. TSX EMT'S
42-	1	Return system (swap) file specification
43-	1	Set transmit/receive speed for a line
45-	1	Access TSX system tables
46-	1	Get or set user name
47-	1	Request operator privilege
48-	1	Establish Break key sentinel control
49-	1	CHKTT -- CHECK I/O TO TT DEVICE
50-	1	CKIOST -- See if scheduler wants to stop I/O
51-	1	SETQ -- Do setup of I/O queue element
52-	1	SETCR -- Set completion routine address in queue element
53-	1	CVTUAD -- Convert user address to physical address
54-	1	LDIO -- Process I/O to logical disks

56-	1	INDIO -- Perform I/O for IND data segment
57-	1	EMTTRC -- Trace EMT execution
58-	1	PRTOCT -- Print an octal value

```

1
2          .TITLE  TSEM2  TSX-Plus EMT Overlay
3          .ENABL  LC
4          .ENABL  AMA
5          .DSABL  GBL
6
7          ; Copyright (C) 1976,1977,1978,1979,1980,1981,1982,1983,1984,1985.
8          ;
9          ; S&H Computer Systems, Inc.
10         ; Nashville, Tennessee
11         ;
12         ; This software is furnished under a license for use only
13         ; on a single computer system and may be copied only with
14         ; the inclusion of the above copyright notice.  This
15         ; software, or any other copies thereof, may not be provided
16         ; or otherwise made available to any other person except
17         ; for use on such system and to one who agrees to these
18         ; license terms.  Title to and ownership of the software
19         ; shall at all times remain with S&H Computer Systems, Inc.
20         ;
21 000000          .CSECT  TSEM2
22 000000  020550  TSEM2:  .RAD50  /EM2/          ;Overlay id
23          177776  PS      =      177776      ;Processor Status Word
24         ;-----
25         ; Macros to enable and disable interrupts.
26         ;
27         .MACRO  DISABL          ;DISABLE INTERRUPTS
28         BIS     #340,@#PS
29         .ENDM   DISABL
30         ;
31         .MACRO  ENABL          ;ENABLE INTERRUPTS
32         BIC     INTPRI,@#PS
33         .ENDM   ENABL
34         ;
35         ; Macro to print an error message when a system crash occurs.
36         ;
37         ; Arguments:
38         ; MSG = Name of error message to print.
39         ; ARG = (Optional) argument value to display with error message.
40         ;
41         .GLOBL  DIEMSG,DIEARG,SYSHLT
42         .MACRO  DIE      MSG,ARG
43         MOV     MSG,@#DIEMSG
44         .IF     NB,ARG
45         MOV     ARG,@#DIEARG
46         .ENDC
47         CALL    @#SYSHLT
48         .ENDM   DIE
49         ;
50         ; Macro definition for calling global routines residing in mapped
51         ; system regions.
52         ;
53         .MACRO  OCALL  ENTADD
54         .IF
55         .ERROR  ;OCALL SPECIFIED WITH NO ENTRY ADDRESS
56         .MEXIT
57         .ENDC

```

```

58          CALL    DVRHC          ;CALL THE OVERLAY HANDLER
59          .WORD    ENTADD         ;SPECIFY THE ENTRY POINT
60          .ENDM
61          ;
62          ; Macro calls
63          ;
64          .MCALL    .PURGE, .TTYOUT
65          ;
66          ; Global definitions
67          ;
68          .GLOBL    E375MX, EMT376, RT11EX
69          .GLOBL    TSEM2, READ, WRITE, DOEMT
70          .GLOBL    CINFLG, SDCLOS, UACHKW, GETQ
71          .GLOBL    QIO, GETUCH, CLASN
72          .GLOBL    QFREE, SPLEMT
73          .GLOBL    IOWAIT, CANMKT
74          .GLOBL    SETERR, EMTXIT
75          .GLOBL    BADEMT
76          .GLOBL    KMNEMT
77          .GLOBL    DETEMT, PMEMT
78          .GLOBL    SSEMT
79          .GLOBL    QMSG
80          ;
81          ; Global references
82          ;
83          .GLOBL    FREEXT, $CFOPN
84          .GLOBL    $VNOTT, SPPRED, $CTRLS, LDMENT, GETCXT, REDCXT, CXTBUF
85          .GLOBL    $DETECH, RUNDEV, Q. PA6, KPAR6, STTCPL, TSXVRS, VLSEMT
86          .GLOBL    DELETE, LOOKUP, ENTER, RENAME, RTDEV, SFDATE, SFPROT
87          .GLOBL    RTSPND, RTRSUM, ABTIO, XHIOUT, XHIIN, XTERCK, XRDTIM
88          .GLOBL    XHISSET, MOUNT, DISMNT, RTEMT, GFINFO, SFTIME, SETPRI
89          .GLOBL    ALCEMT, MONEMT, CPYEMT, TTYIN, TTYOUT, DSTAT, FETCH
90          .GLOBL    CSIGEN, CSISPC, PRINT, CHKUSP, EMTASP, EMTSP, EMTASP
91          .GLOBL    $EMTTR, PO$DBG, NLCHN, DOCOPN, DOOPAP, DOCULK, DORLK
92          .GLOBL    DOTLK, DOULK1, DOSFCK, VMXSF, P2$RLK, VMXWIN, EMTWIN, EMTPLS
93          .GLOBL    MSEND, MSGCK, MSGWT, P2$MSG
94          .GLOBL    PO$OPR, PO$SYS, P2$TRM, PO$NAM
95          .GLOBL    PRIVC2, PO$SND, PO$SPF, PO$BYP
96          .GLOBL    TSXSIT, DS$SFN, $VIRJB, VIMAGE, PO$SPV
97          .GLOBL    LDPDEV, LDSIZE, LDBASE, Q. DEVX, LDFLAG, LD$RON, LDVERS
98          .GLOBL    CQ$LNK, CQ$RTN, CQ$PA5, CQ$JOB, CQ$RO, VPLAS
99          .GLOBL    INTPRI, UPPN, VPAR6, NFRESB, LACTIV
100         .GLOBL    SYTIMH, SYTIML, SYSDAT, SETSPD, VMAXMC, MSGABT, MONABT
101         .GLOBL    PNAME, MAXQVL, NUCHN, ABRTCD, ABRTAD
102         .GLOBL    EMTBLK, LSW2, $CCLRN, EMTPS, PRIVCO
103         .GLOBL    LSW4, LSW5, $INKMN, VALADW, VALADB, $PRGLK
104         .GLOBL    LPROJ, LPROG, IOHALT, EMTADR, MAXLD, LDNAME
105         .GLOBL    SERFLG, UTRPAD, LUNAME, UCLBLK, LPARNT
106         .GLOBL    LSW, $CTRLD, STOP, EXCJOB, CUREMT, DH$$LB, MAPPLS
107         .GLOBL    RPAR, RPDR, ODTBAS, CLRDIR
108         .GLOBL    CQ$FLG, QF$IOT, DX$EBA, DVFLAG, WRITTT
109         .GLOBL    CS$RON, DS$RON, $INDRN, DS$DIR, CFLAG, SFWRIT
110         .GLOBL    SDMOVE, FAKCMP, TTREAD, DS$WON, DCRD1, INTERR, NUMDCD
111         .GLOBL    DCRD2, Q. CHAN, Q. CSW, Q. JOB, UACHKB, Q. BUFF, Q. JNUM
112         .GLOBL    Q. PAR, LDDEVX, Q. COMP, Q. PA5, CXTBAS, KMNBAS
113         .GLOBL    S$TWFN, S$OTLO, S$QMIO, S$QCCB, S$QSPD, S$TTSC
114         .GLOBL    S$TTFN, S$HICP, S$RT, S$WFM, S$LOW, LPARBS, VPAR5

```

```

115 . GLOBL LSTHL
116 . GLOBL SCHED
117 . GLOBL R$CHN, CHNSIZ, R$XCHN, Q. ICSW, Q. UCSW
118 . GLOBL $FORM, CQ$CP, CP$STD
119 . GLOBL RF$WRT, LSW6, $DBGMD
120 . GLOBL EMTMAP, CHNNUM, LJSW, LSTHL
121 . GLOBL CLOSE, INDDBL, INDDBS, R$INTC, $INDDW
122 . GLOBL LSW3, LSLEPH, LSLEPL
123 . GLOBL PASLIN, SCHAIN, LSPND
124 . GLOBL CHNADR, EMTCAD, EMTRAD, LSW3
125 . GLOBL CORUSR, CURCP, SPCPS, $CTRLD, EMTCAS
126 . GLOBL CXTRMN, UFPTRP, JCDB, $HARD, $DEAD
127 . GLOBL VTSLCH, LBRKCQ, LBRKCH, SYNAME, IDABFL
128 . GLOBL MXJPRI, LPRI, LBSPRI, CQ$RNS, CQ$PRI
129 . GLOBL ABORT, VSWPFL, VPRIDF, VPRILO, VPRIHI
130 . GLOBL $ODTMD, LSCCA, MXJADR, FPUUSE, CW$FPU
131 . GLOBL PO$MEM, LSW2S, LIOHLD, LIOCNT
132 . GLOBL MAXPRI, LITIME, $NOINT
133 . GLOBL URO, CHKART, NSPLDV
134 . GLOBL S$NEDQ, DVSTAT
135 . GLOBL JSWLOC, SUTOP, TTOPTS
136 . GLOBL SFSVST, $TTGAG, SB$TXT, SB$END
137 . GLOBL NMUMB, SNMSHD, QNSPND, SB$LNK, SB$PNT
138 . GLOBL LNPRIM, LMSQBF, TRNSTR, LNMAP, MAXSEC, LSECPT
139 . GLOBL EMterr, LSTATE, S$IOWT, UMODE
140 . GLOBL CS$ERR, CS$NMX, CS$ENT
141 . GLOBL CS$EOF, CS$OPN, CS$NMX, DS$NRD
142 . GLOBL C. CSW, C. USED, Q. WCNT, CINDAT, SFCLS
143 . GLOBL C. NUMQ, Q. BLKN, Q. FUNC
144 . GLOBL C. DEVQ, Q. UNIT
145 . GLOBL C. SBLK, C. LENG
146 . GLOBL UCHAN, CSIARE, OVLBIT
147 . GLOBL LBASE, CS$SPL, SFRSST
148 . GLOBL RMNBAS
149 . GLOBL S$WSMB
150 . GLOBL LSTPL, $DILUP
151 . GLOBL LCONTM, LSW9
152 . GLOBL S$IOfN, LSW7, S$OTFN
153 . GLOBL S$TMWT
154 . GLOBL QHDSFN, NUMDEV
155 . GLOBL MRKTHD, CQ$HOT, CQ$LOT
156 . GLOBL LSTD L
157 . GLOBL R$CHN, R$DATE
158 . GLOBL LEMTPC, LSW11, $V52EM
159 . GLOBL VT52, VT100, HAZEL, ADM3A, LA36, LA120
160 . GLOBL DIABLO, QUME, LTRMTP, VT2007, VT2008
161 . GLOBL $FORM
162 . GLOBL $IOMAP
163 . GLOBL CONFIG, LNSBLK
164 . GLOBL JIVLN, JIDLN, JIMLOK, JIPRIV
165 . GLOBL LSTSL, $KINIT, $MLOCK, LNBLKS, MINTIM
166 . GLOBL LPRG1, LPRG2, LCPUIH, LCPULO
167 . GLOBL S$IINWT, S$OTWT, S$SPND, S$SFWT, S$MSWT
168 . GLOBL S$QUSR, S$SFWT, S$SPDB, S$SPCB
169 . GLOBL S$CPU
170 . GLOBL SETC
171 . GLOBL MAXCC

```

```
172          .GLOBL  CSIARE
173          .GLOBL  RC$BAS
174          .GLOBL  SETRBF, CMDB, CMDC, GTSPAC, CMDE, CMDF, CMDG, CMDH
175          .GLOBL  CMDI, CMDJ, CMDK, CMDL, CMDM, CMDN, CMDO, RSSPAC, SFWAC
176          .GLOBL  CMDR, CMDs, CMDT, CMDU, SFWL, CMDW, CMDX, CMDY, CMDZ
177          .GLOBL  SWDBLK, SPLBLK, RSFBLK, UCLDAT, INDFIL
178          .GLOBL  GETDSS, SETDSS, MS$DTR
179          ; Globals used for system mapped regions
180          .GLOBL  OVRHC
```

Branch vector for processing routines

```

1          .SBTTL  Branch vector for processing routines
2          ;-----
3          ;
4          ;  DEFINE PROCESSING ROUTINES FOR EACH EMT CODE
5          ;
6          ;  EMT 375 FUNCTIONS
7          ;
8 000002 0000000 EMTLST: .WORD  DELETE      ; 0  DELETE
9 000004 0000000      .WORD  LOOKUP       ; 1  LOOKUP
10 000006 0000000      .WORD  ENTER       ; 2  ENTER
11 000010 002732'      .WORD  TRPSET      ; 3  SET TRAP
12 000012 0000000      .WORD  RENAME     ; 4  RENAME
13 000014 003470'      .WORD  SVSTAT     ; 5  SAVE STATUS
14 000016 003614'      .WORD  REOPEN     ; 6  REOPEN
15 000020 0000000      .WORD  CLOSE      ; 7  CLOSE
16 000022 001664'      .WORD  READ       ; 10 READ(C)(W)
17 000024 001076'      .WORD  WRITE     ; 11 WRITE(C)(W)
18 000026 004100'      .WORD  DOWAIT     ; 12 WAIT
19 000030 0000000      .WORD  BADEMT     ; 13 COPY CHANNEL
20 000032 0000000      .WORD  RTDEV      ; 14 DEVICE
21 000034 005154'      .WORD  CDFN       ; 15 DEFINE CHANNELS
22 000036 000544'      .WORD  EMT16      ; 16 GROUP 16 EMTS
23 000040 0000000      .WORD  BADEMT     ; 17
24 000042 003224'      .WORD  GTJB       ; 20 GTJB
25 000044 004160'      .WORD  GETTIM     ; 21 GET TIME OF DAY
26 000046 004404'      .WORD  MRKT       ; 22 MARK TIME
27 000050 004544'      .WORD  CMKT       ; 23 CANCEL MARK TIME
28 000052 004320'      .WORD  TIMWAT     ; 24 TIMED WAIT
29 000054 0000000      .WORD  BADEMT     ; 25 SEND DATA
30 000056 0000000      .WORD  BADEMT     ; 26 RECEIVE DATA
31 000060 005032'      .WORD  CHSTAT     ; 27 CHANNEL STATUS
32 000062 002752'      .WORD  SFPA       ; 30 SET FPP EXCEPTION
33 000064 0000000      .WORD  EMTXIT     ; 31 PROTECT VECTORS
34 000066 002422'      .WORD  SPFUN      ; 32 SPECIAL DEV FUNCTIONS
35 000070 0000000      .WORD  EMTXIT     ; 33 SET EXTRA SWAP ADDRESSES
36 000072 005166'      .WORD  GVAL       ; 34 .GVAL
37 000074 002706'      .WORD  SCCA       ; 35 .SCCA
38 000076 000740'      .WORD  EMTPLX     ; 36 .CRRG, .CRAW, .MAP, etc.
39 000100 0000000      .WORD  BADEMT     ; 37 Multi-terminal support
40 000102 004232'      .WORD  SDTTM      ; 40 .SDTTM
41 000104 007000'      .WORD  ESPCPS     ; 41 .SPCPS
42 000106 0000000      .WORD  SFDATE     ; 42 .SFDATE -- Set file date
43 000110 0000000      .WORD  SFPROT     ; 43 .FPROT -- Set file protection
44          000044
45          E375MX = <. -EMTLST>/2
46          ;
47          ;  EMT 374 FUNCTIONS
48 EMT374: .WORD  DOWAIT      ; 0  WAIT
49          .WORD  RTSPND     ; 1  SUSPEND THIS JOB
50          .WORD  RTRSUM     ; 2  RESUME THIS JOB
51          .WORD  PURGE      ; 3  PURGE A CHANNEL
52          .WORD  SERR       ; 4  SET SOFT ERRORS
53          .WORD  HERR       ; 5  SET HARD ERRORS
54          .WORD  CLOSE      ; 6  CLOSE
55          .WORD  EMTXIT     ; 7  TEST & LOCK USR
56          .WORD  DOCIN      ; 10 CHAIN
57          .WORD  BADEMT     ; 11 MESSAGE WAIT
          .WORD  DATE        ; 12 .DATE

```

58	000140	0000000	.WORD	ABTID	; 13 .ABTID
59		000014	E374MX =	<. -EMT374>/2	
60		000060	RT11EX =	<. -EMTLST>/2	
61					
62				EMT'S DEFINED BY TSX.	
63					
64	000142	000566'	TSXEMT: .WORD	CLKOPN	; 100 Open shared file channel
65	000144	000606'	.WORD	CUNLK	; 101 Unlock shared file channel
66	000146	000616'	.WORD	RLOCK	; 102 Lock a specific block
67	000150	000626'	.WORD	TRYLK	; 103 Try to lock a block
68	000152	000766'	.WORD	XMSEND	; 104 Send a message
69	000154	001006'	.WORD	XMSGCK	; 105 See if message is pending
70	000156	001026'	.WORD	XMSGWT	; 106 Wait for a message
71	000160	010150'	.WORD	XSPLSP	; 107 Get # of free spool blocks
72	000162	010032'	.WORD	XGTLN	; 110 Get TSX line #
73	000164	010162'	.WORD	XODTMD	; 111 Set/reset ODT mode
74	000166	010612'	.WORD	XGTTAB	; 112 Get table pointer
75	000170	000636'	.WORD	ULK1BK	; 113 Unlock a single shared file block
76	000172	0000000	.WORD	XHIOUT	; 114 High-efficiency tty output
77	000174	0000000	.WORD	XHIIN	; 115 High-efficiency tty input
78	000176	0000000	.WORD	XTERCK	; 116 Check for TTY input error
79	000200	0000000	.WORD	XRDTIM	; 117 Set TT read timeout
80	000202	0000000	.WORD	XHISCT	; 120 Turn high-efficiency mode on/off
81	000204	000646'	.WORD	CSFACT	; 121 Check if shared file has been modified
82	000206	000656'	.WORD	SFSPND	; 122 Suspend shared file channel
83	000210	010224'	.WORD	CKINAC	; 123 Check for input activation chars
84	000212	010242'	.WORD	GTLICN	; 124 Get TSX license number
85	000214	000576'	.WORD	CAOPPN	; 125 Open shared file with access protection
86	000216	0000000	.WORD	KMNEMT	; 126 KMON EMT's
87	000220	006262'	.WORD	SNDMSG	; 127 Send a message to a line
88	000222	010664'	.WORD	TBLEMT	; 130 Access internal tsx table
89	000224	011364'	.WORD	REQPRV	; 131 Request operator privilege
90	000226	0000000	.WORD	DETEMT	; 132 Detached job control emt
91	000230	011444'	.WORD	GETBRK	; 133 Establish break sentinel control
92	000232	0000000	.WORD	MOUNT	; 134 Mount a file structure
93	000234	0000000	.WORD	DISMNT	; 135 Dismount a file structure
94	000236	0000000	.WORD	PMEMT	; 136 Performance monitor EMT's
95	000240	007672'	.WORD	TRMEMT	; 137 Get terminal type
96	000242	0000000	.WORD	RTEMT	; 140 Real-time EMT's
97	000244	003102'	.WORD	SETSIK	; 141 TSX-Plus Settop
98	000246	010002'	.WORD	GETPPN	; 142 Determine Project-Programmer number
99	000250	0000000	.WORD	SSEMT	; 143 Control shared run-time systems
100	000252	007124'	.WORD	JBINFO	; 144 Get information about a specific line
101	000254	0000000	.WORD	GFINFO	; 145 Get information about a file
102	000256	0000000	.WORD	SFTIME	; 146 Set file creation time
103	000260	011172'	.WORD	EMUNAM	; 147 Get or set user name
104	000262	0000000	.WORD	SETPRI	; 150 Set job execution priority
105	000264	0000000	.WORD	SPLEMT	; 151 Spooling control for current channel
106	000266	006120'	.WORD	TTEMT	; 152 Terminal control
107	000270	010252'	.WORD	INTEMT	; 153 Set interactive/non-interactive mode
108	000272	010376'	.WORD	EMTSPD	; 154 Set terminal transmit/receive speed
109	000274	0000000	.WORD	CLASN	; 155 Assign a CL unit to a line
110	000276	0000000	.WORD	ALCEMT	; 156 Allocate or deallocate a device
111	000300	0000000	.WORD	MONEMT	; 157 Monitor status of a job
112	000302	0000000	.WORD	CPYEMT	; 160 Copy file/priv info from another job
113	000304	000720'	.WORD	CKWIN	; 161 Display window control
114	000306	0000000	.WORD	VLSEMT	; 162 Switch between processes

115	000310	0000000	.WORD	LDMEMT	; 163 Mount a logical disk structure
116	000312	010306'	.WORD	SYFEMT	; 164 Get system file dev: filspc. ext
117		000145	EMTMAX	=	<. -EMTLST>/2

Branch vector for processing routines

```
1  
2  
3  
4 000314 0000000  
5 000316 0000000  
6 000320 0000000  
7 000322 0000000  
8 000324 0000000  
9 000326 0000000  
10 000330 0000000  
11 000332 0000000  
12 000334 006034'  
13 000336 0000000  
14 000340 003370'  
15 000342 003016'  
16 000344 003030'  
17 000346 003340'  
18 000350 006746'  
19 000352 003362'
```

; BRANCH VECTOR FOR EMTS IN THE RANGE 340 TO 357
;
LST16: .WORD TTYIN ; 340 TTYIN
.WORD TTYOUT ; 341 TTYOUT
.WORD DSTAT ; 342 DSTATUS
.WORD FETCH ; 343 FETCH/REL
.WORD CSIGEN ; 344 CSIGEN
.WORD CSISPC ; 345 CSISPC
.WORD EMTXIT ; 346 LOCK USR
.WORD EMTXIT ; 347 RELEASE USR
.WORD EXIT ; 350 EXIT
.WORD PRINT ; 351 PRINT
.WORD SRESET ; 352 SRESET
.WORD QSET ; 353 QSET
.WORD SETTOP ; 354 SETTOP
.WORD RCTRLD ; 355 RCTRLD
.WORD ASTXIT ; 356 Completion routine exit
.WORD HRESET ; 357 HRESET

```

1          .SBTTL  EMT entry processing
2          -----
3          ; Complete EMT entry processing and branch off to routine to do the
4          ; actual processing of the EMT.
5          ;
6          ; At this point the EMT function code, channel number and arguments have
7          ; been moved to EMTBLK which can be directly addressed from kernel mode
8          ; within the EMT processing routines.
9          ;
10         ; If EMT tracing has been requested by SET EMT TRACE,
11         ; print info about this EMT.
12         ;
13 000354 113701 000000G DOEMT:  MOVB    CORUSR,R1      ;GET CURRENT JOB INDEX NUMBER
14 000360 032761 000000G 000000G      BIT    ##EMTTR,LSW6(R1);IS EMT TRACING WANTED?
15 000366 001406          BEQ     2$          ;BR IF NOT
16 000370 032737 000000G 000000G      BIT    #PO$DBG,PRIVCO ;Are we authorized to do EMT tracing?
17 000376 001402          BEQ     2$          ;Br if not
18 000400 004737 013456'          CALL   EMTTRC      ;TRACE THE EMT
19
20         ; Make sure the EMT function code is ok.
21         ;
22 000404 113705 000001G 2$:      MOVB    EMTBLK+1,R5      ;GET FUNCTION CODE
23 000410 020527 000145          CMP     R5,#EMTMAX      ;FUNCTION CODE OK?
24 000414 103402          BLO     5$          ;BR IF OK
25 000416 000137 000000G          JMP     BADEMT        ;INVALID EMT FUNCTION CODE
26
27         ; Make sure channel number is ok.
28         ;
29 000422 120527 000120 5$:      CMPB    R5,#<140+<RT11EX-100>> ;IS THIS TSX REAL-TIME EMT?
30 000426 001443          BEQ     4$          ;BR IF YES -- ALLOW ANY CHANNEL NUMBER
31 000430 005003          CLR     R3
32 000432 153703 000000G      BISB    EMTBLK,R3      ;GET CHANNEL # WITHOUT SIGN EXTENSION
33 000436 012700 000000G      MOV     #NLCHN,R0      ;MAX # CHANNELS FOR EACH JOB
34 000442 032737 000000G 000000G      BIT    #UMODE,EMTPS      ;WAS EMT DONE IN USER OR KERNEL MODE?
35 000450 001406          BEQ     3$          ;BR IF KERNEL MODE
36 000452 032761 000000G 000000G      BIT    ##INKMN,LSW4(R1);IS KMON RUNNING NOW?
37 000460 001002          BNE     3$          ;BR IF YES -- ALLOW ACCESS TO ALL CHANNELS
38 000462 012700 000000G      MOV     #NUCHN,R0      ;USER PROGRAM -- RESTRICT TO CHANS 0-15.
39 000466 020300 3$:      CMP     R3,R0      ;IS THIS CHANNEL # TOO BIG?
40 000470 103032          BHIS    CHNERR      ;BR IF YES
41
42         ; Save address of CSW for channel.
43 000472 010337 000000G      MOV     R3,CHNNUM      ;SAVE USER'S CHANNEL NUMBER
44 000476 013700 000000G      MOV     CXTRMN,R0      ;GET ADDRESS OF SIMULATED RMON
45 000502 062700 000000G      ADD     #R$CHN,R0      ;POINT TO 1ST CHANNEL AREA
46 000506 020327 000021          CMP     R3,#21      ;IS THIS CHANNEL IN EXTENDED CHANNEL AREA?
47 000512 103404          BLO     1$          ;BR IF NOT
48 000514 162703 000021          SUB     #21,R3      ;GET CHANNEL # IN EXTENDED AREA
49 000520 062700 000000G      ADD     #R$XCHN-R$CHN,R0;POINT TO EXTENDED CHANNEL AREA
50 000524 070327 000000G 1$:      MUL     #CHNSIZ,R3      ;MULTIPLY BY # BYTES PER CHANNEL
51 000530 060003          ADD     R0,R3      ;GET ABSOLUTE ADDRESS OF CHANNEL AREA
52 000532 010337 000000G      MOV     R3,CHNADR      ;SAVE CURRENT CHANNEL ADDRESS
53
54         ; Enter EMT processing routine.
55         ; R1 contains user index number.
56         ; URO contains contents of user's R0, put value here to return in R0.
57 000536 006305 4$:      ASL     R5      ;GET 2* FUNCTION CODE VALUE

```

```
58 000540 000175 000002'          JMP      @EMTLST(R5)      ; JUMP TO EMT PROCESSING ROUTINE
59
60                                     ; -----
61                                     ; Sub-function jump routine for EMT's in the range 340-357 (group 16).
62                                     ;
63 000544 113705 0000000          EMT16:  MOVB     EMTBLK,R5      ; GET SUB-FUNCTION CODE
64 000550 006305                   ASL      R5                ; #2
65 000552 000175 000314'          JMP      @LST16(R5)         ; ENTER GROUP-16 PROCESSING ROUTINE
66
67                                     ;
68                                     ; Invalid channel number
69                                     ;
70 000556 012700 177770          CHNERR:  MOV     #-10,R0       ; ERROR CODE #
71 000562 000137 0000000          JMP      SETERR
```

```

1          .SBTTL  Shared file control EMT's
2          ;-----
3          ;  PROCESS THE .CLKOPN EMT WHICH IS USED TO OPEN A CHANNEL
4          ;  TO A SHARED FILE.
5          ;
6 000566 004737 000672' CLKOPN: CALL  CHKSF          ;See if shared file support is available
7 000572 000177 0000000  JMP      @DOCOPN        ;Do the open
8          ;
9          ;-----
10         ;  Open a shared file with access protection.
11         ;
12 000576 004737 000672' CAPOP: CALL  CHKSF          ;See if shared file support is available
13 000602 000177 0000000  JMP      @DOOPAP        ;Open the channel
14         ;
15         ;-----
16         ;  PROCESS THE .CUNLK EMT WHICH IS USED TO UNLOCK
17         ;  A SHARED FILE CHANNEL.
18         ;
19 000606 004737 000672' CUNLK: CALL  CHKSF          ;See if shared file support is available
20 000612 000177 0000000  JMP      @DOCULK        ;Unlock all blocks on channel
21         ;
22         ;-----
23         ;  PROCESS THE .RLOCK EMT WHICH IS USED TO LOCK A
24         ;  SPECIFIC BLOCK.
25         ;
26 000616 004737 000672' RLOCK: CALL  CHKSF          ;See if shared file support is available
27 000622 000177 0000000  JMP      @DORLK         ;Lock the block
28         ;
29         ;-----
30         ;  PROCESS THE .TRYLK EMT WHICH IS USED TO TRY TO LOCK
31         ;  A SPECIFIC BLOCK.
32         ;
33 000626 004737 000672' TRYLK: CALL  CHKSF          ;See if shared file support is available
34 000632 000177 0000000  JMP      @DOTLK         ;Try to lock the block
35         ;
36         ;-----
37         ;  UNLOCK A SPECIFIC SHARED FILE BLOCK.
38         ;
39 000636 004737 000672' ULK1BK: CALL  CHKSF          ;See if shared file support is available
40 000642 000177 0000000  JMP      @DOULK1        ;Unlock a block
41         ;
42         ;-----
43         ;  CHECK FOR WRITES TO A SHARED FILE BY OTHER USERS.
44         ;
45 000646 004737 000672' CSFACT: CALL  CHKSF          ;See if shared file support is available
46 000652 000177 0000000  JMP      @DOSFCK        ;Check for file modification
47         ;
48         ;-----
49         ;  SUSPEND A SHARED FILE CHANNEL
50         ;  THIS HAS THE SAME EFFECT ON THE SHARED FILE LOGIC AS DOING
51         ;  A .SAVESTATUS BUT THE CHANNEL IS UNAFFECTED.
52         ;
53 000656 004737 000672' SFSPND: CALL  CHKSF          ;See if shared file support is available
54 000662 004777 0000000  CALL      @SFSVST        ;TELL SHARED FILE LOGIC
55 000666 000137 0000000  JMP      EMTXIT
56         ;-----
57         ;

```

```

58          ; Determine if shared file support is genned into the system and the
59          ; user is authorized to use it.  If not, return error code 0 for the EMT.
60          ;
61 000672 005737 00000000 CHKSF: TST      VMXSF          ; Is shared file support genned in?
62 000676 001405          BEQ      10$              ; Br if not
63 000700 032737 00000000 00000000 BIT      #P2$RLK,PRIVC2 ; Are we privileged to access shared files?
64 000706 001401          BEQ      10$              ; Br if not
65 000710 000207          RETURN
66          ;
67          ; We cannot use shared file support
68          ;
69 000712 005000 10$:     CLR      R0                ; Return error code 0
70 000714 000137 00000000      JMP      SETERR
71
72          ; -----
73          ; Display window control EMT's
74          ;
75 000720 005737 00000000 CKWIN: TST      VMXWIN          ; Is window support genned in?
76 000724 001003          BNE      1$              ; Br if yes
77 000726 005000          CLR      R0                ; Return error code 0
78 000730 000137 00000000      JMP      SETERR
79 000734 000137 00000000 1$:     JMP      EMTWIN          ; Enter window control overlay

```

```
1  
2  
3  
4  
5  
6 000740 005737 0000000  
7 000744 001402  
8 000746 000137 0000000  
9  
10  
11  
12 000752 012700 000007  
13 000756 005037 0000000  
14 000762 000137 0000000
```

```
      .SBTTL  PLAS EMT's  
      ;-----  
      ; The following code is called when any PLAS related EMT is executed.  
      ; It checks to see if the PLAS support was generated into the system.  
      ;  
EMTPLX: TST      VPLAS      ; Was PLAS support generated into system?  
        BEQ      9$         ; Br if not  
        JMP      EMTPLS     ; Enter TSPLAS overlay  
      ;  
      ; Error: PLAS support not generated into system  
      ;  
9$:     MOV      #7,R0      ; Return error code 7  
        CLR      R0         ; Return 0 in R0  
        JMP      SETERR
```

```

1          .SBTTL  Message communication EMT's
2          ;-----
3          ; PROCESS THE .MSSEND EMT WHICH QUEUES A MESSAGE FOR
4          ; ANOTHER USER.
5          ;
6 000766 004737 001050' XMSSEND: CALL MSGPRV      ;Do we have access to message facility?
7 000772 012700 0000000 MOV      #EMTBLK,RO    ;Point to EMT arg block
8 000776          DCALL  MSEND      ;SEND THE MESSAGE
9 001004 000417          BR      CHKERR    ;CHECK FOR ERRORS
10         ;
11         ;-----
12         ; PROCESS THE .MSGCK EMT WHICH IS USED TO CHECK FOR
13         ; A PENDING MESSAGE.
14         ;
15 001006 004737 001050' XMSGCK: CALL MSGPRV      ;Do we have access to message facility?
16 001012 012700 0000000 MOV      #EMTBLK,RO    ;POINT TO EMT ARG BLOCK
17 001016          DCALL  MSGCK      ;CHECK FOR MESSAGE
18 001024 000407          BR      CHKERR    ;GO CHECK FOR ERRORS
19         ;
20         ;-----
21         ; PROCESS THE .MSGWT EMT WHICH IS USED TO WAIT FOR
22         ; A MESSAGE FROM ANOTHER USER.
23         ;
24 001026 004737 001050' XMSGWT: CALL MSGPRV      ;Do we have access to message facility?
25 001032 012700 0000000 MOV      #EMTBLK,RO    ;POINT TO EMT ARG BLOCK
26 001036          DCALL  MSGWT      ;WAIT FOR THE MESSAGE
27 001044 000137 0000000 CHKERR: JMP      EMTXIT
28         ;
29         ;-----
30         ; See if the message communication facility is genned into the system
31         ; and we are authorized to use it.  If not, return EMT error code 0.
32         ;
33 001050 005737 0000000 MSGPRV: TST      VMAXMC      ;Is message facility genned into system?
34 001054 001405          BEQ      10$          ;Br if not
35 001056 032737 0000000 0000000 BIT      #P2$MSG,PRIVC2 ;Are we authorized to use message facility?
36 001064 001401          BEQ      10$          ;Br if not
37 001066 000207          RETURN
38         ;
39         ; We cannot access message facility
40         ;
41 001070 005000 10$: CLR      RO      ;Return error code 0
42 001072 000137 0000000 JMP      SETERR

```

.WRITE

```

1          .SBTTL .WRITE
2          ;-----
3          ; Process a .WRITE EMT.
4          ;
5 001076 004737 011664' WRITE: CALL CKIOST ;SEE IF SCHEDULER WANTS TO STOP I/O
6 001102 013703 0000000 MOV CHNADR,R3 ;GET ADDRESS OF CHANNEL
7 001106 032763 0000000 0000000 BIT #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN?
8 001114 001004 0000002 BNE 6$ ;BR IF YES
9 001116 012700 0000002 MOV #2,R0 ;CHANNEL NOT OPEN
10 001122 000137 0000000 JMP SETERR
11          ;
12          ; Check to make sure the buffer is on a word boundary
13          ;
14 001126 032737 000001 0000040 6$: BIT #1,EMTBLK+4 ;Is buffer address even?
15 001134 001414 0000000 BEQ 15$ ;Br if yes
16 001136 016302 0000000 MOV C.CSW(R3),R2 ;Get CSW
17 001142 042702 177701 BIC #^C76,R2 ;Get device table index
18 001146 032762 0000000 0000000 BIT #DX$EBA,DVFLAG(R2); Does this device require even buffer addr?
19 001154 001404 0000000 BEQ 15$ ;Br if not
20 001156 012700 177766 MOV #-12,R0 ;Get abort code
21 001162 000137 0000000 JMP SETERR ;Abort the operation
22          ;
23          ; Make sure word count is ok
24          ;
25 001166 005737 0000060 15$: TST EMTBLK+6 ;Is word count negative (>32727.)?
26 001172 001405 0000000 BEQ 17$ ;Br if word count zero
27 001174 100016 0000000 BPL 18$ ;Branch if valid size (positive)
28 001176 012700 177766 MOV #-12,R0 ;Get abort code
29 001202 000137 0000000 JMP SETERR ;Abort the operation
30          ;
31          ; Word count is zero ==> Seek.
32          ; Treat seek like nop but call completion routine if requested.
33          ;
34 001206 005037 0000000 17$: CLR URO ;Say zero words read
35 001212 023727 0000100 000001 CMP EMTBLK+10,#1 ;Need to run completion routine?
36 001220 101402 0000000 BLOS 19$ ;Br if not
37 001222 004737 0000000 CALL FAKCMP ;Call completion routine
38 001226 000137 0000000 19$: JMP EMTXIT ;Finished with read
39          ;
40          ; See if we are writing to TT:
41          ;
42 001232 004737 011640' 18$: CALL CHKTT ;IS CHANNEL OPENED TO "TT:"?
43 001236 103402 0000000 BCS 10$ ;BR IF NOT WRITING TO TT
44 001240 000137 0000000 JMP WRITTT ;ENTER THE TT WRITE ROUTINE IN TSTTY
45          ;
46          ; See if we are writing to a spooled device.
47          ;
48 001244 032763 0000000 0000000 10$: BIT #CS$SPL,C.CSW(R3); IS THIS A SPOOLED DEVICE?
49 001252 001171 0000000 BNE WRTSPL ;BR IF IT IS A SPOOLED DEVICE
50          ;
51          ; See if we are attempting to write to a read-only device.
52          ;
53 001254 016302 0000000 MOV C.CSW(R3),R2 ;GET CSW
54 001260 032702 0000000 BIT #CS$RON,R2 ;IS USER AUTHORIZED FOR READ-ONLY ACCESS?
55 001264 001404 0000000 BEQ 9$ ;BR IF NOT READ-ONLY
56 001266 012700 177763 MOV #-15,R0 ;SET ABORT ERROR CODE
57 001272 000137 0000000 JMP SETERR ;ABORT EMT

```

.WRITE

```

58 001276 042702 177701 9$: BIC #^C76,R2 ;GET DEVICE TABLE INDEX
59 001302 032762 0000000 0000000 BIT #DS$RON,DVSTAT(R2); IS THIS A READ-ONLY DEVICE?
60 001310 001405 BEQ 5$ ;BR IF NOT READ-ONLY
61 001312 005037 0000000 8$: CLR URO ;SAY 0 WORDS WRITTEN
62 001316 005000 CLR RO ;RETURN ERROR CODE OF 0
63 001320 000137 0000000 JMP SETERR
64 ;
65 ; See if error status is pending in CSW for the channel
66 ;
67 001324 032763 0000000 0000000 5$: BIT #CS$ERR,C.CSW(R3);DID A WRITE ERROR OCCUR?
68 001332 001407 BEQ 20$ ;BR IF NOT
69 001334 042763 0000000 0000000 BIC #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
70 001342 012700 000001 MOV #1,RO ;RETURN ERROR CODE OF 1
71 001346 000137 0000000 JMP SETERR
72 ;
73 ; Check if this is an I/O operation by IND
74 ;
75 001352 032761 0000000 0000000 20$: BIT ##INDRN,LSW5(R1); IS IND RUNNING?
76 001360 001402 BEQ 11$ ;BR IF NOT
77 001362 004737 013350' CALL INDIO ;MAP I/O FOR IND
78 ;
79 ; If this is a write by a user-mode program which has opened a file
80 ; structured device in non-file-structured mode, clear the directory
81 ; cache for the device.
82 ;
83 001366 005763 0000000 11$: TST C.SBLK(R3) ;Is device opened non-file-structured mode?
84 001372 001017 BNE 16$ ;Br if not
85 001374 032762 0000000 0000000 BIT #DS$DIR,DVSTAT(R2); Is this a directory structured device?
86 001402 001413 BEQ 16$ ;Br if not
87 001404 023727 0000020 0000000 CMP EMTBLK+2,#DH$#LB; Could this block contain directory?
88 001412 101007 BHI 16$ ;Br if not
89 001414 032737 0000000 0000000 BIT #UMODE,EMTPS ;Was EMT executed in user mode?
90 001422 001403 BEQ 16$ ;Br if not -- Probably TSUSR doing dir I/O
91 001424 DCALL CLRDIR ;Clear out directory cache for device
92 ;
93 ; Build an I/O queue entry for the write request.
94 ;
95 001432 004737 0000000 16$: CALL GETQ ;GET AN I/O QUEUE ENTRY
96 ;
97 ; R1 now points to a free I/O queue entry.
98 ; Move arguments from EMTBLK to the queue entry.
99 ;
100 001436 004737 011772' CALL SETQ ;Do common queue setup
101 001442 103473 BCS 3$ ;Br if we must abort operation
102 001444 005761 0000000 TST Q.WCNT(R1) ;Is word count for transfer zero?
103 001450 001470 BEQ 3$ ;Br if yes -- Nothing to write
104 001452 020063 0000000 CMP RO,C.USED(R3) ;Are we writing beyond previous end point?
105 001456 101416 BLOS 2$ ;Br if not
106 001460 032713 0000000 BIT #CS$ENT,(R3) ;Was file opened with a .ENTER?
107 001464 001413 BEQ 2$ ;Br if not
108 001466 005763 0000000 TST C.SBLK(R3) ;Is this a directory operation?
109 001472 001004 BNE 14$ ;Br if not
110 001474 032762 0000000 0000000 BIT #DS$DIR,DVSTAT(R2) ;Is this a directory structured device?
111 001502 001004 BNE 2$ ;Br if yes
112 001504 010063 0000000 14$: MOV RO,C.USED(R3) ;Remember high-water-mark within file
113 001510 010061 0000000 MOV RO,Q.ICSW+C.USED(R1); Store into chan blk in Q element too
114 001514 005461 0000000 2$: NEG Q.WCNT(R1) ;Negative word count ==> write operation

```

.WRITE

```

115 ;
116 ; If we shortened write request because we hit end of file, return error
117 ; code of 0 but go ahead and do the write.
118 ;
119 001520 023737 0000000 0000060      CMP      URO,EMTBLK+6      ;DID WE SHORTEN WRITE REQUEST?
120 001526 001403                      BEQ      7$              ;BR IF NOT
121 001530 052737 0000000 0000000      BIS      #CFLAG,EMTPS      ;SET C-FLAG IN PS
122 ;
123 ; Set address of write completion routine.
124 ;
125 001536 013700 0000100      7$:      MOV      EMTBLK+10,R0      ;GET ADDRESS OF COMPLETION ROUTINE
126 001542 004737 012472'      CALL     SETCR          ;STORE ADDR OF COMPL ROUTINE INTO QUEUE ELEM
127 001546 103431                      BCS      3$              ;BR IF FATAL ERROR
128 ;
129 ; We have built the I/O queue element.
130 ; Initiate the I/O operation.
131 ;
132 001550 004737 0000000      CALL     QIO              ;QUEUE AN I/O REQUEST
133 ;
134 ; See if we are writing to a shared file
135 ;
136 001554 005737 0000000      TST      JCDB              ;Does job have any shared files open?
137 001560 001402                      BEQ      12$            ;Br if not
138 001562 004777 0000000      CALL     @SFWRIT          ;Check for writes to shared files
139 ;
140 ; See if we should wait for the I/O operation to finish.
141 ;
142 001566 005737 0000100      12$:     TST      EMTBLK+10      ;IS THIS AN .WRITW?
143 001572 001015                      BNE      1$              ;BR IF NOT
144 001574 004737 0000000      CALL     IDWAIT          ;WAIT FOR I/O TO FINISH
145 ;
146 ; See if a hardware error occurred on the write
147 ;
148 001600 032763 0000000 0000000      BIT      #CS$ERR,C.CSW(R3);DID A WRITE ERROR OCCUR?
149 001606 001407                      BEQ      1$              ;BR IF NOT
150 001610 042763 0000000 0000000      BIC      #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
151 001616 012700 0000001      MOV      #1,R0              ;RETURN ERROR CODE OF 1
152 001622 000137 0000000      JMP      SETERR
153 ;
154 ; Finished the EMT.
155 ;
156 001626 000137 0000000      1$:      JMP      EMTXIT          ;EXIT FROM EMT
157 ;
158 ; Some fatal error occurred which causes us to have to abort the operation.
159 ;
160 001632 000137 002364'      3$:      JMP      IOABRT          ;ABORT THE I/O REQUEST
161 ;
162 ; Write to a spooled device.
163 ;
164 001636                      WRTSPL: OCALL     SDMOVE          ;WRITE USER'S DATA TO SPOOL FILE
165 ;
166 ; See if we need to queue a completion routine for this write request.
167 ;
168 001644 023727 0000100 0000001      WRTCPL: CMP      EMTBLK+10,#1      ;WAS A COMPLETION ROUTINE SPECIFIED?
169 001652 101402                      BLOS      1$              ;BR IF NOT
170 001654 004737 0000000      CALL     FAKCMP          ;QUEUE A COMPLETION ROUTINE REQUEST
171 ;

```

```
172      ; Finished spooled write  
173      ;  
174 001660 000137 0000000 1#: JMP EMTXIT
```

```

1          . SBTTL . READ
2
3          ; -----
4          ; Process a .READ EMT.
5          ;
6          READ: CALL CKIOST          ; SEE IF SCHEDULER WANTS TO STOP I/O
7                MOV CHNADR,R3       ; GET ADDRESS OF THE CHANNEL
8                BIT #CS$DPN,C.CSW(R3); IS THE CHANNEL OPEN?
9                BNE 2$              ; BR IF YES
10               MOV #2,R0            ; RETURN ERROR CODE
11               JMP SETERR
12
13          ; Check to make sure the buffer is on a word boundary
14          ;
15          2$: BIT #1,EMTBLK+4       ; Is buffer address even?
16               BEQ 15$              ; Br if yes
17               MOV C.CSW(R3),R2     ; Get CSW
18               BIC #^C76,R2        ; Get device table index
19               BIT #DX$EBA,DVFLAG(R2); Does this device require even buffer addr?
20               BEQ 15$              ; Br if not
21               MOV #-12,R0          ; Get abort code
22               JMP SETERR           ; Abort the operation
23
24          ; See if the read request is so large that
25          ; a handler might interpret it as a write
26          ;
27          15$: TST EMTBLK+6         ; Is word count negative (>32727.)?
28               BEQ 17$              ; Br if word count zero
29               BPL 14$              ; Branch if valid size (positive)
30               MOV #-12,R0          ; Get abort code
31               JMP SETERR           ; Abort the operation
32
33          ; Word count is zero ==> Seek.
34          ; Treat seek like nop but call completion routine if requested.
35          ;
36          17$: CLR URO              ; Say zero words read
37               CMP EMTBLK+10,#1     ; Need to run completion routine?
38               BLOS 19$             ; Br if not
39               CALL FAKCMP          ; Call completion routine
40               JMP EMTXIT           ; Finished with read
41
42          ; See if we are reading from TT:
43          ;
44          14$: CALL CHKTT           ; IS CHANNEL OPENED TO "TT:"?
45               BCS 10$              ; BR IF NOT TT READ
46               JMP TTREAD           ; TT READ FROM MAPPED TSTTY
47
48          ; We are reading from some device other than TT.
49          ; See if we are attempting to read from a write-only device.
50          ;
51          10$: MOV C.CSW(R3),R0     ; GET CSW
52               BIT #CS$SPL,R0       ; IS THIS A SPOOLED DEVICE?
53               BNE 4$               ; BR IF SPOOLED DEVICE
54               BIC #^C76,R0         ; GET DEVICE TABLE INDEX FOR DEVICE
55               BIT #DS$WON,DVSTAT(R0); IS THIS A WRITE-ONLY DEVICE?
56               BEQ 7$               ; BR IF NOT WRITE-ONLY
57               CLR URO              ; SAY 0 WORDS READ

```

```

58 002056 005000 CLR RO ;RETURN ERROR CODE 0
59 002060 000137 0000000 JMP SETERR
60 ;
61 ; See if error flag is pending in CSW for this channel
62 ;
63 002064 032763 0000000 0000000 7$: BIT #CS$ERR,C.CSW(R3);DID A HARDWARE ERROR OCCUR ON READ?
64 002072 001407 BEQ 16$ ;BR IF NOT
65 002074 042763 0000000 0000000 BIC #CS$ERR,C.CSW(R3);ACKNOWLEDGE ERROR STATUS
66 002102 012700 000001 MOV #1,RO ;RETURN ERROR CODE OF 1
67 002106 000137 0000000 JMP SETERR
68 ;
69 ; See if end-of-file status is pending in CSW for this channel
70 ;
71 002112 032763 0000000 0000000 16$: BIT #CS$EOF,C.CSW(R3);DID DEVICE REPORT END OF FILE?
72 002120 001410 BEQ 18$ ;BR IF NOT
73 002122 042763 0000000 0000000 BIC #CS$EOF,C.CSW(R3);ACKNOWLEDGE EOF STATUS
74 002130 005037 0000000 CLR URO ;SAY 0 WORDS READ
75 002134 005000 CLR RO ;RETURN ERROR CODE OF 0
76 002136 000137 0000000 JMP SETERR
77 ;
78 ; See if data being read is currently in the data cache
79 ;
80 002142 005737 0000000 18$: TST JCDB ;Any shared file channels open for job?
81 002146 001412 BEQ 8$ ;Br if not
82 002150 005737 0000000 TST NUMDCD ;Are we doing shared-file data caching?
83 002154 001407 BEQ 8$ ;Br if not
84 002156 004777 0000000 CALL @DCRD1 ;SEE IF DATA IS IN DATA CACHE
85 002162 103404 BCS 8$ ;BR IF DATA WAS NOT IN THE CACHE
86 002164 013737 0000006 0000000 MOV EMTBLK+6,URO ;RETURN WORD COUNT READ IN USER'S RO
87 002172 000624 BR WRTCPL ;FINISH UP AND EXIT
88 ;
89 ; Check for I/O operations by IND
90 ;
91 002174 032761 0000000 0000000 8$: BIT ##INDRN,LSW5(R1);IS IND RUNNING?
92 002202 001402 BEQ 11$ ;BR IF NOT
93 002204 004737 013350' CALL INDIO ;MAP I/O OPERATIONS FOR IND
94 ;
95 ; Build I/O queue element for I/O operation.
96 ;
97 002210 004737 0000000 11$: CALL GETQ ;GET A FREE I/O QUEUE ELEMENT
98 ; R1 now points to a free I/O queue element.
99 002214 004737 011772' CALL SETQ ;DO COMMON QUEUE SETUP
100 002220 103461 BCS IOABRT ;BR IF FATAL ERROR DETECTED
101 002222 013700 0000106 MOV EMTBLK+10,RO ;GET COMPLETION ROUTINE ADDRESS
102 002226 004737 012472' CALL SETCR ;STORE COMPL RTN ADDR IN I/O QUEUE
103 002232 103004 BCC 3$ ;BR IF ADDRESS OK
104 002234 112737 177766 0000000 MOVB #-12,INTERR ;SET ERROR CODE
105 002242 000450 BR IOABRT ;ABORT OPERATION
106 ;
107 ; We have build an I/O queue entry for read request.
108 ; Initiate the I/O operation.
109 ;
110 002244 004737 0000000 3$: CALL QIO ;QUEUE THE REQUEST
111 ;
112 ; See if we need to update data cache
113 ;
114 002250 005737 0000000 TST JCDB ;Any shared file channels for job?

```

.READ

```

115 002254 001405          BEQ      12$          ;Br if not
116 002256 005737 0000000  TST      NUMDCD      ;Are we doing shared-file data caching?
117 002262 001402          BEQ      12$          ;Br if not
118 002264 004777 0000000  CALL     @DCRD2      ;See if we need to update data cache
119
120          ; See if we should wait for the I/O operation to finish.
121
122 002270 005737 0000100  12$:  TST      EMTBLK+10      ;SHOULD WE WAIT?
123 002274 001031          BNE      1$          ;BR IF NOT
124 002276 004737 0000000  CALL     IOWAIT      ;WAIT FOR I/O OPERATION TO FINISH
125
126          ; See if a hardware error occurred on the read
127
128 002302 032763 0000000 0000000  BIT      #CS$ERR,C.CSW(R3);DID A HARDWARE ERROR OCCUR ON READ?
129 002310 001407          BEQ      6$          ;BR IF NOT
130 002312 042763 0000000 0000000  BIC      #CS$ERR,C.CSW(R3);ACKNOWLEDGE ERROR STATUS
131 002320 012700 000001    MOV      #1,R0          ;RETURN ERROR CODE OF 1
132 002324 000137 0000000  JMP      SETERR
133
134          ; See if the device reported end-of-file.
135
136 002330 032763 0000000 0000000  6$:  BIT      #CS$EOF,C.CSW(R3);DID DEVICE REPORT END OF FILE?
137 002336 001410          BEQ      1$          ;BR IF NOT
138 002340 042763 0000000 0000000  BIC      #CS$EOF,C.CSW(R3);ACKNOWLEDGE EOF STATUS
139 002346 005037 0000000  CLR      URO          ;SAY 0 WORDS READ
140 002352 005000  CLR      R0          ;RETURN ERROR CODE OF 0
141 002354 000137 0000000  JMP      SETERR
142
143          ; Finished operation.
144
145 002360 000137 0000000  1$:  JMP      EMTXIT
146
147          ; We detected a fatal error before queueing the request.
148          ; Return the I/O queue entry to the free list.
149
150 002364 004737 0000000  IOABRT: CALL     QFREE          ;RETURN I/O QUEUE ENTRY TO FREE LIST
151 002370 113700 0000000  MOVB     INTERR,R0      ;GET ERROR CODE
152 002374 000137 0000000  JMP      SETERR          ;ERROR RETURN

```

```
1          .SBTTL .SERR
2          ;-----
3          ; PROCESS THE .SERR EMT
4 002400 112737 000001 0000000 SERR:  MOVB  #1,SERFLG      ;REMEMBER USER ISSUED SERR
5 002406 000137 0000000      JMP    EMTXIT
6
7          .SBTTL .HERR
8          ;-----
9          ; PROCESS THE .HERR EMT
10 002412 105037 0000000 HERR:  CLRB  SERFLG      ;TURN OFF .SERR CONTROL
11 002416 000137 0000000      JMP    EMTXIT
```

.SPFUN

```

1          .SBTTL .SPFUN
2          ;-----
3          ; Perform special device functions.
4          ;
5 002422 004737 011664'  SPFUN: CALL CKIOST      ; See if scheduler wants to hold I/O starts
6 002426 013703 0000000  MOV      CHNADR,R3    ; GET ADDRESS OF CHANNEL BLOCK
7 002432 016302 0000000  MOV      C.CSW(R3),R2  ; GET CSW
8 002436 032702 0000000  BIT      #CS$OPN,R2    ; IS CHANNEL OPEN?
9 002442 001004          BNE      1$            ; BR IF YES
10 002444 012700 0000002  MOV      #2,R0        ; ERROR 2 IF CHANNEL NOT OPEN
11 002450 000137 0000000  JMP      SETERR
12          ;
13          ; See if device can accept special functions.
14          ;
15 002454 042702 177701 1$:      BIC      #177701,R2    ; EXTRACT DEVICE TABLE INDEX #
16 002460 032762 0000000 0000000  BIT      #DS$SFN,DVSTAT(R2); CAN DEVICE HANDLE SPECIAL FUNCTIONS?
17 002466 001002          BNE      2$            ; BR IF YES
18 002470 000137 0000000 4$:      JMP      EMTXIT      ; IGNORE .SPFUN REQUEST
19          ;
20          ; Check function code
21          ;
22 002474 105737 0000110 2$:      TSTB     EMTBLK+11    ; FUNCTION CODE MUST BE NEGATIVE
23 002500 002403          BLT      3$            ; BR IF OK
24 002502 005000          CLR      R0            ; ERROR CODE 0 IF NOT NEGATIVE FUNCTION
25 002504 000137 0000000  JMP      SETERR
26 002510 112737 000377 0000100 3$:      MOVB     #377,EMTBLK+10 ; MAKE SURE EVEN BYTE = 377
27          ;
28          ; Make sure user is privileged to do .SPFUN's
29          ;
30 002516 032762 0000000 0000000  BIT      #DS$DIR,DVSTAT(R2) ; Is this a directory-structured device?
31 002524 001413          BEQ      7$            ; Br if not
32 002526 032737 0000000 0000000  BIT      #P0$SPF,PRIVCO    ; Is user authorized to do .SPFUN?
33 002534 001007          BNE      7$            ; Br if yes
34 002536 120237 0000000  CMPB     R2,LDDEVX      ; Is this a .SPFUN to device LD?
35 002542 001404          BEQ      7$            ; Br if yes -- That is ok
36 002544 012700 177747  MOV      #-31,R0        ; Error -31 -- Not authorized for .SPFUN
37 002550 000137 0000000  JMP      SETERR
38          ;
39          ; Get a free I/O queue element and set it up.
40          ;
41 002554 004737 0000000 7$:      CALL     GETQ          ; GET A FREE I/O QUEUE ELEMENT
42          ;
43          ; R1 now points to I/O queue element.
44          ; Set special function code in queue element.
45          ;
46 002560 113761 0000110 0000000  MOVB     EMTBLK+11,Q.FUNC(R1); STORE SPECIAL-FUNCTION CODE VALUE
47          ;
48          ; Set up buffer address and other info in queue element
49          ;
50 002566 004737 011772'  CALL     SETQ          ; SET UP BUFFER ADDRESS AND WORD COUNT
51 002572 103443          BCS      9$            ; BR IF SOME ERROR DETECTED
52          ;
53          ; Set up address of completion routine.
54          ;
55 002574 013700 0000120  MOV      EMTBLK+12,R0    ; GET ADDRESS OF COMPLETION ROUTINE
56 002600 004737 012472'  CALL     SETCR      ; SET UP COMPL ADDR IN QUEUE ELEMENT
57 002604 103436          BCS      9$            ; BR IF INVALID COMPLETION QUEUE ADDRESS

```

```

58      ;
59      ; Initiate the I/O operation.
60      ;
61 002606 004737 0000000  CALL    QIO          ; QUEUE THE I/O REQUEST
62      ;
63      ; See if we should wait for the I/O operation to finish.
64      ;
65 002612 005737 0000120  TST     EMTBLK+12      ; IS WAIT REQUESTED?
66 002616 001002          BNE     13$          ; BR IF NOT
67      ;
68      ; Wait for I/O to finish
69      ;
70 002620 004737 0000000  CALL    IDWAIT        ; WAIT FOR I/O TO FINISH
71      ;
72      ; See if a hardware error or end-of-file occurred.
73      ;
74 002624 032763 0000000 0000000 13$:  BIT     #CS$ERR,C.CSW(R3); DID A HARDWARE ERROR OCCUR?
75 002632 001407          BEQ     6$          ; BR IF NOT
76 002634 042763 0000000 0000000  BIC     #CS$ERR,C.CSW(R3); CLEAR ERROR FLAG IN CSW
77 002642 012700 0000001          MOV     #1,R0          ; REPORT ERROR
78 002646 000137 0000000          JMP     SETERR
79 002652 032763 0000000 0000000 6$:  BIT     #CS$EOF,C.CSW(R3); DID WE HIT END-OF-FILE?
80 002660 001406          BEQ     5$          ; BR IF NOT
81 002662 042763 0000000 0000000  BIC     #CS$EOF,C.CSW(R3); CLEAR END-OF-FILE FLAG IN CSW
82 002670 005000          CLR     R0          ; RETURN ERROR CODE OF 0
83 002672 000137 0000000          JMP     SETERR
84      ;
85      ; Normal completion of .SPFUN
86      ;
87 002676 000137 0000000 5$:  JMP     EMTXIT        ; FINISHED EMT
88      ;
89      ; Some fatal error occurred. Abort request.
90      ;
91 002702 000137 002364' 9$:  JMP     IOABRT        ; ABORT I/O REQUEST
    
```

.SCCA

```

1          . SBTTL . SCCA
2          ; -----
3          ; PROCESS THE . SCCA EMT
4          ;
5 002706 016137 000000G 000000G SCCA:  MOV     LSCCA(R1),URO  ; RETURN OLD SCCA RTN ADDRESS TO USER IN R0
6 002714 013700 000002G          MOV     EMTBLK+2,R0      ; GET ADDRESS OF SCCA ROUTINE
7 002720 004737 000000G          CALL    VALADW          ; VALIDATE THE ADDRESS
8 002724 010061 000000G          MOV     R0,LSCCA(R1)      ; SAVE ADDRESS OF SCCA ROUTINE FOR JOB
9 002730 000406          BR         JNOP
10
11         . SBTTL . TRPSET
12         ; -----
13         ; PROCESS THE . TRPSET EMT
14         ;
15 002732 013700 000002G TRPSET: MOV     EMTBLK+2,R0      ; GET ADDRESS OF USER'S TRAP ROUTINE
16 002736 004737 000000G          CALL    VALADW          ; VALIDATE THE ADDRESS
17 002742 010037 000000G          MOV     R0,UTRPAD        ; SET TRAP ROUTINE ADDRESS FOR JOB
18 002746 000137 000000G JNOP:   JMP      EMTXIT
19
20         . SBTTL . SFPA
21         ; -----
22         ; PROCESS THE . SFPA EMT
23         ;
24 002752 013700 000002G SFPA:   MOV     EMTBLK+2,R0      ; GET ADDRESS OF TRAP ROUTINE
25 002756 004737 000000G          CALL    VALADB          ; VALIDATE IT (NOTE: ADDRESS MAY = 1)
26 002762 010037 000000G          MOV     R0,UFPTRP        ; SET ADDRESS OF TRAP ROUTINE
27 002766 005700          TST      R0                    ; IS USER GOING TO USE FPU UNIT?
28 002770 001407          BEQ      2$                    ; BR IF NOT
29 002772 032737 000000G 000000G          BIT      #CW$FPU,CONFIG ; DOES SYSTEM HAVE A FPU UNIT?
30 003000 001403          BEQ      2$                    ; BR IF NOT
31 003002 110137 000000G          MOVB    R1,FPUUSE        ; REMEMBER USER IS ACCESSING FPU
32 003006 000402          BR        3$
33 003010 105037 000000G 2$:     CLRB    FPUUSE          ; SAY FPU NOT IN USE
34 003014 000754          3$:     BR      JNOP
35
36         . SBTTL . QSET
37         ; -----
38         ; Process the . QSET EMT.
39         ; This is essentially a NOP under TSX-Plus.
40         ; The address beyond the end of the queue buffer is returned in R0.
41         ;
42 003016 013737 000004G 000000G QSET:  MOV     EMTBLK+4,URO      ; Return address of Q buffer in R0
43 003024 000137 000000G          JMP      EMTXIT          ; Finished with EMT
44
45         . SBTTL . SETTOP
46         ; -----
47         ; PROCESS THE . SETTOP EMT
48         ;
49 003030 013700 000000G SETTOP: MOV     URO,R0          ; GET USER'S REQUESTED TOP ADDRESS
50 003034 005200          SETTP1: INC     R0              ; BOUND ADDRESS UP TO WORD IF IT'S ODD
51 003036 001001          BNE      2$                    ; BR IF DID NOT OVERFLOW
52 003040 005300          DEC     R0                    ; IF OVERFLOWED, FORCE BACK TO 177777
53 003042 042700 000001 2$:     BIC     #1,R0            ; FORCE ADDRESS TO BE EVEN
54 003046 020037 000000G          CMP     R0,ODTBAS        ; DOES REQUEST EXCEED JOB'S LIMIT?
55 003052 103404          BLO      1$                    ; BR IF NOT
56 003054 013700 000000G          MOV     ODTBAS,R0        ; GET MAX AMT JOB IS ALLOWED TO USE
57 003060 162700 000002          SUB     #2,R0            ; POINT TO WORD BELOW TOP LIMIT

```

```

58 003064 010037 0000000 1$: MOV R0, URO ; RETURN TOP ADDRESS IN USER'S R0
59 003070 010046 MOV R0, -(SP) ; AND IN LOCATION 50
60 003072 106637 000050 MTPD @#50
61 003076 000137 0000000 JMP EMTXIT ; FINISHED
62
63 ; -----
64 ; The TSX-Plus SETSIZ emt is used to change the memory space allocated
65 ; to a job.
66 ;
67 ; The form of the EMT argument block is:
68 ;
69 ; . BYTE 0, 141
70 ; . WORD . top-addr ; Requested top address for job
71 ;
72 003102 013700 0000020 SETSIZ: MOV EMTBLK+2, R0 ; GET REQUESTED TOP OF JOB ADDRESS
73 003106 105737 0000000 TSTB VSWPFL ; IS THIS A NON-SWAPPING SYSTEM?
74 003112 001750 BEQ SETTP1 ; IF NON-SWAPPING THEN CAN'T CHANGE SIZE
75 003114 120137 0000000 CMPB R1, EXCJOB ; DOES JOB HAVE EXCLUSIVE ACCESS TO SYSTEM?
76 003120 001745 BEQ SETTP1 ; IF YES THEN CAN'T CHANGE SIZE
77 003122 013702 0000000 MOV MXJADR, R2 ; GET ADDRESS ABOVE TOP OF AVAILABLE SPACE
78 003126 162702 0000002 SUB #2, R2 ; GET ADDRESS OF TOP AVAILABLE WORD
79 003132 020002 CMP R0, R2 ; IS REQUEST ABOVE TOP ALLOWED?
80 003134 101401 BLOS 1$ ; BR IF NOT
81 003136 010200 MOV R2, R0 ; CONSTRAIN TO TOP ALLOWED ADDRESS
82 003140 010037 0000000 1$: MOV R0, URO ; RETURN TOP ADDRESS TO USER IN R0
83 003144 010046 MOV R0, -(SP) ; AND PUT IN USER'S LOCATION 50
84 003146 106637 000050 MTPD @#50
85 003152 020027 160000 CMP R0, #160000 ; IS JOB NOW USING MORE THAN 56KB?
86 003156 103412 BLO 2$ ; BR IF NOT
87 003160 052761 0000000 0000000 BIS #VIMAGE, LJSW(R1) ; SET VIRTUAL-JOB FLAG
88 003166 016146 0000000 MOV LJSW(R1), -(SP)
89 003172 106637 0000000 MTPD @#JSWLOC
90 003176 052761 0000000 0000000 BIS ##VIRJB, LSW9(R1) ; ALSO SET IN INTERNAL SYSTEM TABLE
91 003204 062700 0000002 2$: ADD #2, R0 ; GET ADDRESS ABOVE TOP WORD WANTED
92 003210 010037 0000000 MOV R0, ODTBAS ; CONSTRAIN .SETTOP TO THIS
93 003214 004737 0000000 CALL SUTOP ; DO MEMORY ALLOCATION FOR JOB
94 003220 000137 0000000 JMP EMTXIT ; FINISHED

```

1				.SBTTL .GTJB
2				-----
3				; PROCESS THE .GTJB EMT
4				;
5	003224	013700	0000020	GTJB: MOV EMTBLK+2,R0 ; GET ADDRESS OF USER'S BUFFER
6	003230	004737	0000000	CALL VALADW ; VALIDATE BUFFER ADDRESS
7	003234	010146		MOV R1,-(SP) ; RETURN JOB #
8	003236	006216		ASR (SP)
9	003240	106620		MTPD (R0)+ ; [WORD 1]
10	003242	013746	0000000	MOV ODTBAS,-(SP) ; HIGH MEMORY LIMIT
11	003246	106620		MTPD (R0)+ ; [WORD 2]
12	003250	005046		CLR -(SP) ; LOW MEMORY
13	003252	106620		MTPD (R0)+ ; [WORD 3]
14	003254	012746	0000000	MOV #RMNBAS+R\$CHN-2,-(SP); START OF JOB CHANNEL AREA
15	003260	106620		MTPD (R0)+ ; [WORD 4]
16	003262	005046		CLR -(SP) ; ADDRESS OF JOB'S IMPURE AREA
17	003264	106620		MTPD (R0)+ ; [WORD 5]
18	003266	010146		MOV R1,-(SP) ; JOB'S TERMINAL NUMBER
19	003270	006216		ASR (SP)
20	003272	106620		MTPD (R0)+ ; [WORD 6]
21	003274	013746	0000000	MOV ODTBAS,-(SP) ; VIRTUAL HIGH LIMIT
22	003300	106620		MTPD (R0)+ ; [WORD 7]
23	003302	000137	0000000	GTJJ: JMP EMTXIT

.CHAIN

```

1          .SBTTL .CHAIN
2          ;-----
3          ; PROCESS A .CHAIN REQUEST.
4          ;
5 003306 105237 0000000 DDCIN: INCB CINFLG ; SAY .CHAIN IN PROGRESS
6          ;
7          ; Move chain data to job context area.
8          ;
9 003312 012702 000500      MOV      #500,R2      ; POINT TO START OF CHAIN DATA AREA
10 003316 012703 0000000     MOV      #CINDAT,R3    ; POINT TO CHAIN DATA SAVE AREA IN CONTEXT BLK
11 003322 012704 000140     MOV      #<<1000-500>/2>,R4 ; GET # WORDS TO MOVE
12 003326 106522          1$: MFPD      (R2)+      ; GET A WORD FROM USER'S CHAIN DATA AREA
13 003330 012623          MOV      (SP)+,(R3)+    ; MOVE TO CONTEXT BLOCK AREA
14 003332 077403          SOB      R4,1$
15 003334 004737 0000000     CALL     STOP        ; EXIT TO KMON TO DO .CHAIN
16
17          .SBTTL .RCTRLO
18          ;-----
19          ; PROCESS THE .RCTRLO EMT
20          ;
21 003340 032761 0000000 0000000 RCTRLO: BIT      ##CFOPN,LSW4(R1); Is input coming from a command file?
22 003346 001003          BNE      1$             ; Br if yes -- Don't reset ctrl-D
23 003350 042761 0000000 0000000 BIC      ##CTRLO,LSW3(R1); Reset Ctrl-D flag
24 003356 000137 0000000          1$: JMP      EMTXIT

```

.HRESET

```

1          .SBTTL .HRESET
2          -----
3          ; Process the .HRESET EMT
4          ; (Halt all I/O for job and then purge all I/O channels)
5          ;
6 003362 004737 000000G HRESET: CALL IOHALT      ;HALT ALL I/O FOR JOB
7 003366 000400          BR          SRESET      ;NOW DO .SRESET
8
9          .SBTTL .SRESET
10         -----
11         ; PROCESS THE .SRESET AND .HRESET EMTS.
12         ;
13 003370 SRESET:
14         ;
15         ; Reset .SPND/.RSUM counter for job
16         ;
17 003370 005061 000000G CLR LSPND(R1)      ;RESET .SPND COUNT FOR JOB
18         ;
19         ; Cancel any pending .MRKT requests
20         ;
21 003374 004737 004724' CALL CANMKT      ;CANCEL PENDING MARK-TIME REQUESTS
22         ;
23         ; Cancel any job monitoring requests
24         ;
25 003400 DCALL MONABT      ;Cancel any job monitoring
26         ;
27         ; Cancel any message-read completion routine requests
28         ;
29 003406 005737 000000G TST VMAXMC      ;Is message support genned in?
30 003412 001403 BEQ 4$      ;Br if not
31 003414 DCALL MSGABT      ;Abort any message-read cpl routines
32         ;
33         ; Purge I/O channels
34         ;
35 003422 012702 177777G 4$: MOV #NUCHN-1,R2 ;GET HIGHEST USER CHANNEL #
36 003426 020227 000017 1$: CMP R2,#17      ;IS THIS CHANNEL 17?
37 003432 001004 BNE 2$      ;BR IF NOT
38 003434 032761 000000G 000000G BIT #OVLBIT,LJSW(R1); IS THE PROGRAM OVERLAYED?
39 003442 001004 BNE 3$      ;IF IT IS, DON'T CLOSE CHANNEL 17
40 003444 2$: .PURGE R2      ;PURGE EACH ONE
41 003454 005302 3$: DEC R2
42 003456 002363 BGE 1$
43         ;
44         ; Undo .CDFN
45         ;
46 003460 005037 000000G CLR UCHAN      ;RESET USER CHANNEL DEFINITIONS
47 003464 000137 000000G JMP EMTXIT

```

```

1          .SBTTL .SAVESTATUS
2          ;-----
3          ; Save channel status.
4          ;
5 003470 013703 0000000 SVSTAT: MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6 003474 032763 0000000 0000000 BIT     #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN?
7 003502 001417          BEQ     4$              ;BR IF NOT OPEN NOW
8          ; Make sure file was opened with a lookup.
9 003504 032763 0000000 0000000 BIT     #CS$ENT,C.CSW(R3); OPENED WITH LOOKUP OR ENTER?
10 003512 001404          BEQ     2$              ;BR IF LOOKUP
11 003514 012700 0000001          MOV     #1,R0      ;RETURN ERROR CODE 1
12 003520 000137 0000000          JMP     SETERR
13          ;
14          ; Wait for any I/O activity on the channel to finish
15          ;
16 003524 004737 0000000 2$:      CALL     IOWAIT      ;WAIT FOR I/O TO FINISH
17          ;
18          ; Tell TSX shared file logic that savestatus is being done
19          ;
20 003530 005737 0000000          TST     JCDB        ;Any shared file channels open for job?
21 003534 001402          BEQ     4$              ;Br if not
22 003536 004777 0000000          CALL     @SFSVST      ;TELL SHARED FILE SYSTEM
23          ;
24          ; Move channel block to user's area
25          ;
26 003542 010301 4$:      MOV     R3,R1              ;SAVE ADDRESS OF CHANNEL AREA
27 003544 013700 0000020          MOV     EMTBLK+2,R0    ;GET ADDRESS OF USER'S SAVE AREA
28 003550 004737 0000000          CALL     VALADW      ;VALIDATE THE ADDRESS
29 003554 012704 0000005          MOV     #5,R4        ;GET # WORDS TO MOVE
30 003560 012346 3$:      MOV     (R3)+,-(SP)        ;MOVE DATA TO USER'S AREA
31 003562 106620          MTPD     (R0)+
32 003564 077403          SOB      R4,3$
33          ;
34          ; Mark channel as closed
35          ;
36 003566 032761 0000000 0000000 BIT     #CS$OPN,C.CSW(R1); IS THE CHANNEL OPEN?
37 003574 001404          BEQ     5$              ;BR IF CHANNEL IS NOT OPEN
38 003576 005061 0000000          CLR     C.CSW(R1)    ;PURGE THE CHANNEL
39 003602 000137 0000000          JMP     EMTXIT      ;FINISHED
40 003606 005000 5$:      CLR     R0                ;RETURN ERROR CODE OF 0 IF CHANNEL CLOSED
41 003610 000137 0000000          JMP     SETERR
42

```

.REOPEN

```

1          .SBTTL .REOPEN
2          ;-----
3          ; Reopen a channel using saved status information.
4          ;
5 003614 013703 0000000 REOPEN: MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6          ;
7          ; Make sure channel is not now open.
8          ;
9 003620 032763 0000000 0000000 BIT     #CS$OPN,C.CSW(R3); IS CHANNEL OPEN NOW?
10 003626 001403 BEQ     1$              ;BR IF NOT
11 003630 005000 CLR     R0              ;RETURN ERROR CODE OF 0
12 003632 000137 0000000 JMP     SETERR
13          ;
14          ; Channel is closed. Restore status.
15          ;
16 003636 013700 0000020 1$: MOV     EMTBLK+2,R0      ;GET ADDRESS OF USER'S SAVE STATUS BLOCK
17 003642 004737 0000000 CALL    VALADW      ;VALIDATE THE ADDRESS
18 003646 012704 0000005 MOV     #5,R4      ;GET # WORDS TO MOVE
19 003652 106520 3$: MFPD    (R0)+      ;GET A WORD FROM USER'S AREA
20 003654 012623 MOV     (SP)+,(R3)+      ;MOVE INTO CHANNEL BLOCK
21 003656 077403 SOB     R4,3$      ;MOVE ALL
22          ;
23          ; Check consistency of savestatus information that we restored.
24          ;
25 003660 013703 0000000 MOV     CHNADR,R3      ;POINT TO CHANNEL BLOCK
26 003664 016300 0000000 MOV     C.CSW(R3),R0      ;GET CHANNEL STATUS WORD (CSW)
27 003670 032700 0000000 BIT     #CS$OPN,R0      ;CHANNEL SHOULD BE OPEN NOW
28 003674 001425 BEQ     9$              ;ERROR IF NOT
29 003676 032700 0000000 BIT     #CS$ENT,R0      ;CHANNEL SHOULD HAVE BEEN OPENED VIA LOOKUP
30 003702 001022 BNE     9$              ;ERROR IF NOT
31 003704 105063 0000000 CLRB    C.NUMQ(R3)      ;THERE SHOULD BE NO ACTIVE I/O OPERATIONS
32 003710 126327 0000000 0000007 CMPB   C.DEVQ(R3),#7      ;CHECK VALIDITY OF UNIT NUMBER
33 003716 101014 BHI     9$              ;ERROR IF TOO BIG
34 003720 042700 0000000 BIC     #^C<CS$NMX>,R0      ;GET DEVICE INDEX NUMBER
35 003724 020037 0000000 CMP     R0,NUMDEV      ;MAKE SURE IT IS A VALID NUMBER
36 003730 101007 BHI     9$              ;BR IF TOO BIG
37          ;
38          ; Tell shared file system about the reopen.
39          ;
40 003732 005737 0000000 TST     JCDB          ;Does job have any shared file channels?
41 003736 001402 BEQ     4$              ;Br if not
42 003740 004777 0000000 CALL    @SFRSST      ;WE MAY BE REOPENING A SHARED FILE
43          ;
44          ; Finished
45          ;
46 003744 000137 0000000 4$: JMP     EMTXIT
47          ;
48          ; Error -- Savestatus info we restored was invalid.
49          ; Return error code of 1.
50          ;
51 003750 005063 0000000 9$: CLR     C.CSW(R3)      ;PURGE THE CHANNEL
52 003754 012700 0000001 MOV     #1,R0      ;RETURN ERROR CODE 1
53 003760 000137 0000000 JMP     SETERR

```

.PURGE

```

1          .SBTTL .PURGE
2          ;-----
3          ; Purge a channel.
4          ;
5 003764 013703 0000000 PURGE: MOV CHNADR,R3 ;GET ADDRESS OF CHANNEL BLOCK
6 003770 032763 0000000 0000000 BIT #CS*OPN,C.CSW(R3); IS THE CHANNEL OPEN NOW?
7 003776 001434 BEQ 1$ ;BR IF NOT OPEN
8 004000 004737 0000000 CALL IOWAIT ;WAIT FOR I/O ON CHANNEL TO FINISH
9 004004 113701 0000000 MOVB CHNNUM,R1 ;GET CHANNEL NUMBER
10 004010 105737 0000000 TSTB NSPLDV ;Are there any spooled devices?
11 004014 001403 BEQ 2$ ;BR IF NOT
12 004016 DCALL SDCLOS ;SEE IF THIS IS A SPOOLED DEVICE
13 004024 005737 0000000 2$: TST JCDB ;Does job have any shared file channels?
14 004030 001402 BEQ 4$ ;Br if not
15 004032 004777 0000000 CALL @SFCLS ;SEE IF IT IS A SHARED FILE
16          ;
17          ; If we are purging a channel opened to a mag tape and I/O abort
18          ; entry point code is turned off, then treat the purge like a close.
19          ;
20 004036 016300 0000000 4$: MOV C.CSW(R3),R0 ;GET CHANNEL STATUS WORD
21 004042 042700 0000000 BIC #^C<CS$NMX>,R0 ;MASK OUT ALL BUT DEVICE INDEX NUMBER
22 004046 032760 0000000 0000000 BIT #DS$NRD,DVSTAT(R0); IS DEVICE A MAGNETIC TAPE?
23 004054 001405 BEQ 1$ ;BR IF NOT
24 004056 005737 0000000 TST IOABFL ;IS IO ABORT IN EFFECT?
25 004062 001002 BNE 1$ ;BR IF YES
26 004064 000137 0000000 JMP CLOSE ;TREAT PURGE ON MAG TAPE LIKE CLOSE
27 004070 005063 0000000 1$: CLR C.CSW(R3) ;MARK CHANNEL AS CLOSED
28 004074 000137 0000000 JMP EMTXIT ;FINISHED

```

.WAIT

```

1          .SBTTL .WAIT
2          ;-----
3          ; .WAIT -- Wait for all I/O to finish on a channel.
4          ;
5 004100 013703 00000000 DOWAIT: MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL
6          ;
7          ; Make sure channel is open
8          ;
9 004104 032763 00000000 00000000      BIT     #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN?
10 004112 001003          BNE     1$      ;BR IF YES
11 004114 005000          CLR     R0      ;RETURN ERROR CODE OF 0 IF NOT OPEN
12 004116 000137 00000000          JMP     SETERR
13          ;
14          ; Wait for all I/O to finish
15          ;
16 004122 004737 00000000 1$: CALL     IOWAIT      ;WAIT FOR I/O TO FINISH
17          ;
18          ; Check for hardware errors
19          ;
20 004126 032763 00000000 00000000      BIT     #CS$ERR,C.CSW(R3); DID A HARD ERROR OCCUR?
21 004134 001407          BEQ     9$      ;BR IF NOT
22 004136 042763 00000000 00000000      BIC     #CS$ERR,C.CSW(R3); CLEAR THE ERROR FLAG IN THE CSW
23 004144 012700 00000001          MOV     #1,R0      ;Return error code 1
24 004150 000137 00000000          JMP     SETERR
25          ;
26          ; Finished
27          ;
28 004154 000137 00000000 9$: JMP     EMTXIT      ;RETURN

```

.GTIM

```

1          .SBTTL .GTIM
2          ;-----
3          ; Get time of day.
4          ;
5 004160 013700 0000020 GETTIM: MOV EMTBLK+2,R0 ;GET ADDR OF USER'S TIME BUFFER
6 004164 004737 0000000 CALL VALADW ;VALIDATE THE ADDRESS
7 004170 013746 0000000 MOV SYTIMH,-(SP) ;RETURN HIGH-ORDER TIME
8 004174 106620 MTPD (R0)+
9 004176 013746 0000000 MOV SYTIML,-(SP) ;RETURN LOW-ORDER TIME
10 004202 106610 MTPD (R0)
11 004204 000403 BR UPDATE ;GO UPDATE DATE CELL
12
13          .SBTTL .DATE
14          ;-----
15          ; Get current date (return to user in R0).
16          ;
17 004206 013737 0000000 0000000 DATE: MOV SYSDAT,URO ;RETURN DATE TO USER IN R0
18          ; Move current date value to cell in user's simulated monitor vector table.
19 004214 013702 0000000 UPDATE: MOV CXTRMN,R2 ;GET BASE ADDRESS OF JOB VECTOR
20 004220 013762 0000000 0000000 MOV SYSDAT,R$DATE(R2);SET DATE VALUE
21 004226 000137 0000000 JMP EMTXIT
22
23          .SBTTL .SDTTM
24          ;-----
25          ; .SDTTM -- Set system date and time.
26          ;
27 004232 032737 0000000 0000000 SDTTM: BIT #P0$OPR,PRIVCO ;IS THIS A PRIVILEGED USER RUNNING?
28 004240 001003 BNE 1$ ;BR IF YES
29 004242 005000 CLR R0 ;RETURN ERROR CODE 0
30 004244 000137 0000000 JMP SETERR
31
32          ; Set date
33          ;
34 004250 013702 0000020 1$: MOV EMTBLK+2,R2 ;GET ADDRESS OF DATE/TIME VALUE BLOCK
35 004254 010200 MOV R2,R0 ;VALIDATE THE ADDRESS
36 004256 004737 0000000 CALL VALADW
37 004262 106522 MFPD (R2)+ ;GET NEW DATE VALUE
38 004264 012600 MOV (SP)+,R0
39 004266 002402 BLT 2$ ;IF NEGATIVE THEN DON'T CHANGE DATE
40 004270 010037 0000000 MOV R0,SYSDAT ;SET NEW SYSTEM DATE
41
42          ; Set time
43          ;
44 004274 106522 2$: MFPD (R2)+ ;GET HIGH-ORDER TIME VALUE
45 004276 012600 MOV (SP)+,R0
46 004300 002405 BLT 3$ ;IF NEGATIVE, DON'T CHANGE TIME VALUE
47 004302 010037 0000000 MOV R0,SYTIMH ;SET NEW HIGH-ORDER TIME VALUE
48 004306 106522 MFPD (R2)+ ;GET NEW LOW-ORDER TIME VALUE
49 004310 012637 0000000 MOV (SP)+,SYTIML ;SET LOW-ORDER TIME VALUE
50
51          ; Finished
52          ;
53 004314 000137 0000000 3$: JMP EMTXIT

```

.TWAIT

```

1          .SBTTL .TWAIT
2          ;-----
3          ; Do a timed wait
4          ;
5 004320 013700 0000020 TIMWAT: MOV     EMTBLK+2,R0      ;GET POINTER TO TIME VALUE BLOCK
6 004324 004737 0000000      CALL    VALADW        ;VALIDATE THE ADDRESS
7 004330 106520      MFPD    (R0)+                ;GET HIGH-ORDER TIME VALUE
8 004332 012661 0000000      MOV     (SP)+,LSLEPH(R1)
9 004336 106510      MFPD    (R0)                ;GET LOW-ORDER TIME VALUE
10 004340 012661 0000000      MOV     (SP)+,LSLEPL(R1)
11          ;
12          ; Suspend job for timed interval
13          ;
14 004344 004737 0000000 1$: CALL    CHKABT        ;MAKE SURE WE HAVEN'T BEEN ABORTED
15 004350 005761 0000000      TST     LSLEPH(R1)    ;IS THERE STILL SOME SLEEP TIME LEFT?
16 004354 002411      BLT     3$                    ;BR IF NOT (TIME HAS GONE NEGATIVE)
17 004356 003003      BGT     2$                    ;BR IF SOME TIME LEFT
18 004360 005761 0000000      TST     LSLEPL(R1)    ;CHECK LOW-ORDER VALUE
19 004364 001405      BEQ     3$                    ;BR IF TIME IS UP
20 004366 012700 0000000 2$: MOV     #S$TMWT,R0    ;STATE BECOMES TIMED WAIT
21 004372 004737 0000000      CALL    QHDSPN        ;SUSPEND THE JOB
22 004376 000762      BR       1$
23          ;
24          ; Timed wait is finished
25          ;
26 004400 000137 0000000 3$:  JMP     EMTXIT

```

```

1          .SBTTL .MRKT
2          ;-----
3          ; Start a mark-time request.
4          ;
5 004404    MRKT:
6          ;
7          ; Get an I/O queue entry.
8          ;
9 004404    004737    00000000          CALL    GETQ          ;GET A FREE I/O QUEUE ELEMENT
10         ;
11         ; Set up the queue element as a completion queue element.
12         ; (R1 = Address of queue element)
13         ;
14 004410    113761    00000000 00000000      MOVB    CORUSR,CQ$JOB(R1);SET JOB # IN QUEUE ELEMENT
15         ; Set time value.
16 004416    013700    00000020      MOV     EMTBLK+2,R0      ;GET ADDRESS OF TIME VALUE ARG BLOCK
17 004422    004737    00000000      CALL    UACHKW          ;MAKE SURE ADDRESS IS LEGAL
18 004426    103440          BCS     9$          ;BR IF INVALID ADDRESS
19 004430    106520          MFPD     (R0)+        ;GET HIGH-ORDER TIME VALUE
20 004432    012661    00000000      MOV     (SP)+,CQ$HOT(R1);SET HIGH-ORDER TIME VALUE
21 004436    106510          MFPD     (R0)        ;GET LOW-ORDER TIME VALUE
22 004440    012661    00000000      MOV     (SP)+,CQ$LOT(R1)
23         ; Set address of completion routine
24 004444    013700    00000040      MOV     EMTBLK+4,R0      ;GET ADDRESS OF COMPLETION ROUTINE
25 004450    004737    00000000      CALL    UACHKW          ;MAKE SURE ADDRESS IS LEGAL
26 004454    103425          BCS     9$          ;BR IF NOT LEGAL
27 004456    010061    00000000      MOV     R0,CQ$RTN(R1)    ;SET ADDRESS OF COMPLETION ROUTINE
28         ; Set mapping for kernel PAR 5 that was in effect when EMT was executed.
29 004462    013761    00000000 00000000      MOV     EMTMAP,CQ$PA5(R1);SET EMT ENTRY MAPPING FOR PAR 5
30         ; Set ID number
31 004470    013761    00000060 00000000      MOV     EMTBLK+6,CQ$RO(R1);SET COMPLETION ID NUMBER
32         ;
33         ; Add queue entry to mark-time list.
34         ;
35 004476          DISABL          ;** DISABLE **
36 004504    013761    00000000 00000000      MOV     MRKTHD,CQ$LNK(R1);ADD TO FRONT OF LIST
37 004512    010137    00000000      MOV     R1,MRKTHD
38 004516          ENABL          ;** ENABLE **
39         ;
40         ; Finished
41         ;
42 004524    000137    00000000          JMP     EMTXIT
43         ;
44         ; Error -- Invalid address in MRKT argument list.
45         ;
46 004530    004737    00000000      9$: CALL    QFREE          ;RELEASE THE QUEUE ELEMENT
47 004534    012700    177766      MOV     #-12,R0          ;RETURN ERROR CODE
48 004540    000137    00000000      JMP     SETERR          ;ABORT EMT

```

```

1          . SBTTL . CMKT
2          ;-----
3          ; Cancel mark-time request.
4          ;
5 004544 010103 CMKT:  MOV     R1,R3          ; COPY JOB INDEX #
6 004546 013702 0000020  MOV     EMTBLK+2,R2      ; GET ID VALUE
7 004552 001460      BEQ     1$              ; BR IF WE SHOULD CANCEL ALL REQUESTS FOR JOB
8          ;
9          ; Cancel a specifiex mark-time request for this job.
10         ;
11 004554 012704 0000000      MOV     #MRKTHD-CQ$LNK,R4;FAKE UP POINTER TO LIST HEAD
12 004560      DISABL          ;** DISABLE **
13 004566 016401 0000000 4$:  MOV     CQ$LNK(R4),R1  ; GET ADDRESS OF NEXT MARK-TIME QUEUE ENTRY
14 004572 001442      BEQ     5$              ; BR IF REACHED END OF LIST
15 004574 120361 0000000      CMPB    R3,CQ$JOB(R1)  ; DOES THIS ENTRY BELONG TO OUR JOB?
16 004600 001007      BNE     2$              ; BR IF NOT
17 004602 132761 0000000 0000000  BITB    #QF$IOT,CQ$FLG(R1); IS THIS A .TIMID ENTRY?
18 004610 001003      BNE     2$              ; BR IF YES
19 004612 020261 0000000      CMP     R2,CQ$RO(R1)  ; DO ID NUMBERS MATCH?
20 004616 001402      BEQ     3$              ; BR IF YES -- FOUND ENTRY TO DELETE
21 004620 010104      2$:  MOV     R1,R4          ; CHAIN FORWARD TO NEXT ENTRY IN LIST
22 004622 000761      BR      4$
23         ;
24         ; Found specified queue entry -- delete from list.
25         ;
26 004624 016164 0000000 0000000 3$:  MOV     CQ$LNK(R1),CQ$LNK(R4);RELINK LIST AROUND US
27 004632      ENABL          ;** ENABLE **
28         ;
29         ; See if we need to return remaining time.
30         ;
31 004640 013700 0000040      MOV     EMTBLK+4,R0    ; DOES USER WANT REMAINING TIME?
32 004644 001411      BEQ     6$              ; BR IF NOT
33 004646 004737 0000000      CALL    UACHKW        ; MAKE SURE RECIEVING AREA IS LEGAL
34 004652 103406      BCS     6$              ; BR IF NOT LEGAL
35 004654 016146 0000000      MOV     CQ$HOT(R1),-(SP);RETURN HIGH-ORDER TIME
36 004660 106620      MTPD    (R0)+
37 004662 016146 0000000      MOV     CQ$LOT(R1),-(SP);RETURN LOW-ORDER TIME
38 004666 106610      MTPD    (R0)
39         ;
40         ; Return queue element to free list.
41         ;
42 004670 004737 0000000 6$:  CALL    QFREE          ; FREE THE Q ELEMENT
43         ;
44         ; Finished
45         ;
46 004674 000137 0000000      JMP     EMTXIT
47         ;
48         ; Could not find specified queue element -- return error code 0.
49         ;
50 004700      5$:  ENABL          ;** ENABLE **
51 004706 005000      CLR     R0              ; RETURN ERROR CODE 0
52 004710 000137 0000000      JMP     SETERR
53         ;
54         ; Cancel all mark-time requests for this job.
55         ;
56 004714 004737 004724' 1$:  CALL    CANMKT        ; CANCEL ALL MARK-TIME REQUESTS FOR JOB
57 004720 000137 0000000      JMP     EMTXIT        ; FINISHED

```

```

1
2 ; -----
3 ; CANMKT is called to cancel all mark-time requests for the current job.
4 ; It is called when a .CMKT EMT is done with an id of 0 or when a job
5 ; exits.
6 ;
7 ; Inputs:
8 ;   MRKTHD = Head of mark-time queue chain.
9
10 004724 010146 CANMKT: MOV R1,-(SP)
11 004726 010446 MOV R4,-(SP)
12 004730 012704 0000000 1$: MOV #MRKTHD-CQ$LNK,R4;DUMMY POINTER TO LIST HEAD
13 004734 016401 0000000 2$: DISABL ;** DISABLE **
14 004742 001423 0000000 MOV CQ$LNK(R4),R1 ;GET NEXT ENTRY IN LIST
15 004746 001423 0000000 BEQ 4$ ;BR IF REACHED END OF LIST
16 004750 123761 0000000 0000000 CMPB CQRUSR,CQ$JOB(R1);IS THIS ENTRY FOR THIS JOB?
17 004756 001015 0000000 0000000 BNE 3$ ;BR IF NOT
18 004760 132761 0000000 0000000 BITB #QF$IOT,CQ$FLG(R1);IS THIS A HANDLER .TIMID ENTRY?
19 004766 001011 0000000 0000000 BNE 3$ ;BR IF YES
20 004770 016164 0000000 0000000 MOV CQ$LNK(R1),CQ$LNK(R4);RELINK LIST AROUND US
21 004776 004737 0000000 ENABL ;** ENABLE **
22 005004 000747 0000000 CALL QFREE ;RETURN QUEUE ELEMENT TO FREE LIST
23 005010 000747 0000000 BR 1$
24 005012 010104 0000000 3$: MOV R1,R4 ;LINK TO NEXT ELEMENT IN LIST
25 005014 000752 0000000 BR 2$
26 ;
27 ; Finished
28
29 005016 004737 0000000 4$: ENABL ;** ENABLE **
30 005024 012604 0000000 MOV (SP)+,R4
31 005026 012601 0000000 MOV (SP)+,R1
32 005030 000207 0000000 RETURN

```

.CSTAT

```

1          .SBTTL .CSTAT
2          ;-----
3          ; Return channel status
4          ;
5 005032 013703 0000000 CHSTAT: MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6          ;
7          ; Make sure channel is open.
8          ;
9 005036 032763 0000000 0000000 BIT     #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN?
10 005044 001003          BNE     1$              ;BR IF YES
11 005046 005000          CLR     R0              ;RETURN ERROR CODE OF 0
12 005050 000137 0000000          JMP     SETERR
13          ;
14          ; Channel is open. Return status.
15          ;
16 005054 013704 0000020 1$:      MOV     EMTBLK+2,R4      ;GET ADDRESS OF USER'S STATUS BLOCK
17 005060 010400          MOV     R4,R0              ;VALIDATE ADDRESS
18 005062 004737 0000000          CALL    VALADW
19          ; Point R0 to user's channel status block passed in EMT argument.
20 005066 010437 0000000          MOV     R4,URO      ;Point R0 to channel status block
21          ; Return CSW
22 005072 016346 0000000          MOV     C.CSW(R3),-(SP)
23 005076 106624          MTPD     (R4)+
24          ; Return starting block number of file
25 005100 016346 0000000          MOV     C.SBLK(R3),-(SP)
26 005104 106624          MTPD     (R4)+
27          ; Return allocated length of file.
28 005106 016346 0000000          MOV     C.LENG(R3),-(SP)
29 005112 106624          MTPD     (R4)+
30          ; Return highest block number written so far
31 005114 016346 0000000          MOV     C.USED(R3),-(SP)
32 005120 106624          MTPD     (R4)+
33          ; Return device unit number.
34 005122 005046          CLR     -(SP)              ;ENSURE HIGH BYTE IS CLEAR
35 005124 116316 0000000          MOVB    C.DEVQ(R3),(SP) ;MOVE UNIT NUMBER INTO LOW BYTE
36 005130 106624          MTPD     (R4)+
37          ; Return Rad50 device name.
38 005132 016301 0000000          MOV     C.CSW(R3),R1      ;GET DEVICE TABLE #
39 005136 042701 177701          BIC     #^C76,R1        ;GET DEV # ONLY
40 005142 016146 0000000          MOV     PNAME(R1),-(SP) ;GET RAD50 DEVICE NAME
41 005146 106624          MTPD     (R4)+
42          ;
43          ; Finished
44          ;
45 005150 000137 0000000          JMP     EMTXIT
46
47          .SBTTL .CDFN
48          ;-----
49          ; Define channels.
50          ;
51          ;
52 005154 013737 0000020 0000000 CDFN:  MOV     EMTBLK+2,UCHAN ;SAVE ADDRESS OF USER CHANNEL SPACE
53 005162 000137 0000000          JMP     EMTXIT

```

```

1          .SBTTL .GVAL
2          ;-----
3          ; .GVAL, .PVAL, .PEEK, .POKE
4          ;
5 005166 113700 0000000 GVAL:  MOVB  EMTBLK,R0      ;GET SUB-FUNCTION CODE (.GVAL/.PVAL etc.)
6 005172 120027 0000003      CMPB  R0,#3        ;MAKE SURE IT IS REASONABLE
7 005176 101404          BLOS  1$            ;BR IF OK
8 005200 012700 177767      MOV   #-11,R0      ;INVALID FUNCTION CODE
9 005204 000137 0000000      JMP   SETERR
10 005210 006300      1$:  ASL    R0            ;GET 2 * FUNCTION CODE
11 005212 013702 0000020      MOV   EMTBLK+2,R2    ;GET OFFSET VALUE
12 005216 013703 0000040      MOV   EMTBLK+4,R3    ;GET VALUE TO STORE (.PVAL & .POKE)
13 005222 000170 006024'      JMP   @GVJMP(R0)     ;JUMP TO PROCESSING ROUTINE
14          ;
15          ; .GVAL -- Get value from RMON table
16          ;
17 005226 032702 0000001 DOGVAL: BIT    #1,R2      ;OFFSET CANNOT BE ODD
18 005232 001005          BNE    2$            ;BR IF ODD
19 005234 005702          TST    R2            ;IS OFFSET VALUE NEGATIVE?
20 005236 002415          BLT    TSXGVL        ;IF YES THEN HE'S GETTING A TSX-PLUS VALUE
21 005240 020227 0000000      CMP    R2,#MAXGVL    ;IS IT TOO BIG?
22 005244 103403          BLO    1$            ;BR IF OK
23 005246 005000      2$:  CLR    R0            ;RETURN ERROR CODE OF 0
24 005250 000137 0000000      JMP   SETERR
25 005254 063702 0000000      1$:  ADD    CXTRMN,R2    ;ADD BASE ADDR OF USER'S RMON AREA
26 005260 016237 0000002 0000000      MOV    2(R2),UR0    ;RETURN VALUE IN USER'S R0
27 005266 000137 0000000      JMP   EMTXIT
28          ;
29          ; Get TSX-Plus system value
30          ;
31 005272 005402      TSXGVL: NEG    R2            ;GET POSITIVE OFFSET VALUE
32 005274 020227 0000042      CMP    R2,#MXTSXV    ;IS IT IN LEGAL RANGE?
33 005300 101403          BLOS  1$            ;BR IF OK
34 005302 005000          CLR    R0            ;RETURN ERROR CODE 0
35 005304 000137 0000000      JMP   SETERR
36 005310 005000      1$:  CLR    R0            ;INITIALIZE VALUE TO ZERO
37 005312 016202 005326'      MOV    TSXGVC(R2),R2    ;GET ADDRESS OF ROUTINE TO SET VALUE
38 005316 004712          CALL   @R2            ;CALL ROUTINE TO GET VALUE
39 005320 010037 0000000      MOV    R0,UR0        ;RETURN VALUE TO USER IN R0
40 005324 000137 0000000      JMP   EMTXIT        ;FINISHED
41          ;
42          ; TSX-Plus value offset vector
43          ;
44          TSXGVC = .-2
45 005330 005372'      .WORD  TGJOBN      ; -2. -- Get job #
46 005332 005400'      .WORD  TGLDIN      ; -4. -- Get Lead-in character
47 005334 005406'      .WORD  TGPRIV      ; -6. -- Determine if this is a privileged job
48 005336 005422'      .WORD  TGIOMP      ; -8. -- Determine if I/O page mapping in effect
49 005340 005436'      .WORD  TGPROJ      ; -10. -- Job's project number
50 005342 005444'      .WORD  TGPROG      ; -12. -- Job's programmer number
51 005344 005452'      .WORD  TGLICN      ; -14. -- TSX-Plus site license number
52 005346 005460'      .WORD  TGCPRI      ; -16. -- Current execution priority
53 005350 005466'      .WORD  TGMPRI      ; -18. -- Maximum authorized priority
54 005352 005510'      .WORD  TGUCLB      ; -20. -- Number of data blocks/job in UCL file
55 005354 005516'      .WORD  TGPRIL      ; -22. -- Primary line #
56 005356 005544'      .WORD  TGSYNM      ; -24. -- Name of physical SY device
57 005360 005474'      .WORD  TGPRLO      ; -26. -- Get value of PRILOW

```

```

58 005362 005502' .WORD TGPRHI ; -28. -- Get value of PRIHI
59 005364 005534' .WORD TGPRNT ; -30. -- Get parent job number
60 005366 005552' .WORD TGVERS ; -32. -- Get system version number
61 005370 005560' .WORD TGRSUB ; -34. -- Get relative subprocess number
62 000042 MXTSXV = <. -TSXGVC>-2 ; Maximum legal offset value
63 ;
64 ; Get job number
65 ;
66 005372 010100 TGJOBN: MOV R1,RO ; GET JOB INDEX NUMBER
67 005374 006200 ASR RO ; CONVERT TO JOB NUMBER
68 005376 000207 RETURN
69 ;
70 ; Get Lead-in character
71 ;
72 005400 113700 0000000 TGLDIN: MOVB VTSLCH,RO ; GET LEAD-IN CHARACTER VALUE
73 005404 000207 RETURN
74 ;
75 ; Determine if this is a privileged job
76 ;
77 005406 032737 0000000 0000000 TGPRIV: BIT #PO$SYS,PRIVCO ; ARE WE PRIVILEGED?
78 005414 001401 BEQ 1$ ; BR IF NOT
79 005416 005200 INC RO ; SET PRIVILEGED FLAG
80 005420 000207 1$: RETURN
81 ;
82 ; Determine if I/O page mapping is in effect
83 ;
84 005422 032761 0000000 0000000 TGIOMP: BIT #$IOMAP,LSW6(R1); IS I/O PAGE MAPPING IN EFFECT?
85 005430 001401 BEQ 1$ ; BR IF NOT
86 005432 005200 INC RO ; SAY MAPPING IS IN EFFECT
87 005434 000207 1$: RETURN
88 ;
89 ; Get Project number
90 ;
91 005436 016100 0000000 TGPROJ: MOV LPROJ(R1),RO ; GET PROJECT #
92 005442 000207 RETURN
93 ;
94 ; Get programmer number
95 ;
96 005444 016100 0000000 TGPROG: MOV LPROG(R1),RO ; GET PROGRAMMER #
97 005450 000207 RETURN
98 ;
99 ; Get TSX-Plus site license number
100 ;
101 005452 013700 0000000 TGLICN: MOV TSXSIT,RO ; GET LICENSE NUMBER
102 005456 000207 RETURN
103 ;
104 ; Get current job execution priority
105 ;
106 005460 116100 0000000 TGCPRI: MOVB LPRI(R1),RO ; Get current job priority
107 005464 000207 RETURN
108 ;
109 ; Get maximum authorized job priority
110 ;
111 005466 113700 0000000 TGMPRI: MOVB MXJPRI,RO ; Get maximum authorized execution priority
112 005472 000207 RETURN
113 ;
114 ; Get value of PRILOW

```

```

115
116 005474 113700 0000000  TGPRLO: MOVB  VPRILO,R0
117 005500 000207          RETURN
118
119          ; Get value of PRIHI
120
121 005502 113700 0000000  TGPRHI: MOVB  VPRIHI,R0
122 005506 000207          RETURN
123
124          ; Get number of blocks per job in TSXUCL data base file
125
126 005510 013700 0000000  TGUCLB: MOV   UCLBLK,R0      ;Get number of blocks per job
127 005514 000207          RETURN
128
129          ; Get number of controlling primary line (0 if we are a primary line)
130
131 005516 016100 0000000  TGPRIL: MOV   LNPRIM(R1),R0  ;Get number of our primary line
132 005522 020001          CMP   R0,R1      ;Are we the primary line?
133 005524 001001          BNE   1$         ;Br if not
134 005526 005000          CLR   R0         ;Return 0 if we are a primary line
135 005530 006200 1$:      ASR   R0         ;Convert from 2*job # to job #
136 005532 000207          RETURN
137
138          ; Get number of parent job
139
140 005534 016100 0000000  TGPRNT: MOV   LPARNT(R1),R0  ;Get parent job index number
141 005540 006200          ASR   R0         ;Convert to job number
142 005542 000207          RETURN
143
144          ; Get physical name of SY device
145
146 005544 013700 0000000  TGSYNM: MOV   SYNAME,R0      ;Get SY physical device name
147 005550 000207          RETURN
148
149          ; Get system version number
150
151 005552 012700 0000000  TGVERS: MOV   #TSXVRS,R0      ;Get system version number
152 005556 000207          RETURN
153
154          ; Get relative subprocess number
155
156 005560 010546          TGRSUB: MOV   R5,-(SP)      ;Save R5
157 005562 005000          CLR   R0         ;Primary and detached are relative 0
158 005564 020127 0000000  CMP   R1,#LSTDLL      ;Is this a virtual line?
159 005570 003414          BLE   8$         ;Branch if not
160 005572 016105 0000000  MOV   LNPRIM(R1),R5      ;Get owner line number
161 005576 016505 0000000  MOV   LSECPT(R5),R5      ;Get pointer to owner's subprocess list
162 005602 005200 1$:      INC   R0         ;Count subprocess number
163 005604 120125          CMPB  R1,(R5)+      ;Is this our line?
164 005606 001405          BEQ   8$         ;Done if line matches
165 005610 020027 0000000  CMP   R0,#MAXSEC      ;Any more possible?
166 005614 002772          BLT   1$         ;Keep checking if so
167 005616 012700 177777  MOV   #-1,R0        ;Whoa! This should never happen!
168 005622 012605 8$:      MOV   (SP)+,R5      ;Restore R5
169 005624 000207          RETURN

```

```

1
2      ; .PVAL -- Store value into monitor table
3
4 005626 032702 000001      PVAL:  BIT    #1,R2      ; IS ADDRESS ODD?
5 005632 001071              BNE     GVBAD        ; BR IF YES -- ERROR
6 005634 020227 000000G     CMP     R2,#MAXGVL    ; IS VALUE IN PROPER RANGE?
7 005640 101066              BHI     GVBAD        ; BR IF NOT
8 005642 063702 000000G     ADD     CXTRMN,R2     ; ADD BASE ADDRESS OF JOB'S RMON AREA
9 005646 016237 000002 000000G  MOV     2(R2),URO  ; RETURN ORIGINAL VALUE IN R0
10 005654 010362 000002      MOV     R3,2(R2)    ; STORE NEW VALUE
11 005660 000137 000000G     JMP     EMTXIT       ; FINISHED
12
13      ; .PEEK -- Get value from specified memory location
14
15 005664 032702 000001      PEEK:  BIT    #1,R2      ; IS ADDRESS ODD?
16 005670 001052              BNE     GVBAD        ; BR IF YES -- ERROR
17 005672 020227 000000G     CMP     R2,#RMNBAS    ; IS ADDRESS WITHIN RMON TABLE?
18 005676 103413              BLO     PKLOW       ; BR IF TOO LOW
19 005700 020227 000000G     CMP     R2,#RMNBAS+MAXGVL ; IS IT TOO BIG?
20 005704 101010              BHI     PKLOW       ; BR IF YES
21 005706 162702 177776G     SUB     #<RMNBAS-2>,R2 ; MAKE ADDRESS INTO OFFSET
22 005712 063702 000000G     ADD     CXTRMN,R2     ; ADD BASE ADDRESS OF JOB'S RMON AREA
23 005716 011237 000000G     PKGET:  MOV     (R2),URO  ; RETURN VALUE
24 005722 000137 000000G     JMP     EMTXIT
25      ; Peek into low memory area or I/O page
26 005726 032737 000000G 000000G  PKLOW:  BIT    #PO$MEM,PRIVCO ; Are we authorized to access phys memory?
27 005734 001430              BEQ     GVBAD        ; BR IF NOT
28 005736 000767              BR      PKGET       ; GO GET THE VALUE
29
30      ; .POKE -- Store value into specified memory location
31
32 005740 032702 000001      POKE:  BIT    #1,R2      ; IS ADDRESS ODD?
33 005744 001024              BNE     GVBAD        ; ERROR IF YES
34 005746 020227 000000G     CMP     R2,#RMNBAS    ; IS ADDRESS WITHIN RMON TABLE?
35 005752 103414              BLO     POLOW       ; BR IF TOO LOW
36 005754 020227 000000G     CMP     R2,#RMNBAS+MAXGVL ; IS IT TOO BIG?
37 005760 101011              BHI     POLOW       ; BR IF YES
38 005762 162702 177776G     SUB     #<RMNBAS-2>,R2 ; CONVERT TO OFFSET WITHIN TABLE
39 005766 063702 000000G     ADD     CXTRMN,R2     ; ADD BASE OF JOB'S RMON AREA
40 005772 011237 000000G     POVAL:  MOV     (R2),URO  ; GET ORIGINAL VALUE
41 005776 010312              MOV     R3,(R2)     ; SET NEW VALUE
42 006000 000137 000000G     JMP     EMTXIT       ; FINISHED
43      ; Poke into low memory or I/O page
44 006004 032737 000000G 000000G  POLOW:  BIT    #PO$MEM,PRIVCO ; Are we authorized to access phys memory?
45 006012 001401              BEQ     GVBAD        ; BR IF NOT
46 006014 000766              BR      POVAL       ; GO STORE VALUE
47
48      ; Invalid offset specified
49
50 006016 005000              GVBAD:  CLR     R0      ; RETURN ERROR # 0
51 006020 000137 000000G     JMP     SETERR
52
53      ; Sub-function jump vector
54
55 006024 005226'            GVJMP:  .WORD  DOGVAL    ; 0 - .GVAL
56 006026 005664'            .WORD  PEEK          ; 1 - .PEEK
57 006030 005626'            .WORD  PVAL          ; 2 - .PVAL

```

58 006032 005740'

.WORD POKE

;3 - .POKE

.EXIT

```

1          .SBTTL .EXIT
2          ;-----
3          ; Program exit
4          ;
5 006034 005061 0000000 EXIT: CLR LEMTPC(R1) ; CLEAR ADDRESS OF LAST USER-MODE EMT
6          ;
7          ; See if program wants to pass some commands to TSKMON.
8          ;
9 006040 032761 0000000 0000000 BIT #PASLIN!SCHAIN,LJSW(R1); DOES PROGRAM WANT TO PASS COMMANDS?
10 006046 001422 BEQ 2$ ; BR IF NOT
11          ; Move commands from job chain data area to context area chain data cell.
12 006050 012704 000510 MOV #510,R4 ; ADDRESS OF CELL WITH BYTE COUNT
13 006054 106524 MFDP (R4)+ ; GET COUNT OF # BYTES TO MOVE
14 006056 012602 MOV (SP)+,R2
15 006060 020227 000266 CMP R2,#<1000-512> ; LIMIT TO LARGEST POSSIBLE SIZE
16 006064 101402 BLOS 3$ ; BR IF OK
17 006066 012702 000266 MOV #<1000-512>,R2
18 006072 012703 0000100 3$: MOV #CINDAT+10,R3 ; POINT TO CHAIN DATA CELL IN CONTEXT AREA
19 006076 010223 MOV R2,(R3)+ ; STORE BYTE COUNT
20 006100 001405 BEQ 2$ ; BR IF NULL COMMAND BEING PASSED TO US
21 006102 005202 INC R2 ; ROUND UP
22 006104 006202 ASR R2 ; GET # WORDS TO MOVE
23 006106 106524 1$: MFDP (R4)+ ; GET NEXT WORD OF DATA TO MOVE
24 006110 012623 MOV (SP)+,(R3)+ ; MOVE TO CONTEXT AREA
25 006112 077203 SOB R2,1$ ; MOVE ALL OF COMMANDS
26          ;
27          ; Stop program execution and enter TSKMON
28          ;
29 006114 004737 0000000 2$: CALL STOP ; TERMINATE PROGRAM EXECUTION

```

TTEMT -- Terminal control EMT

```

1          .SBTTL  TTEMT  -- Terminal control EMT
2          ;-----
3          ; The terminal control EMT is used to perform all of the terminal
4          ; control functions that can also be performed by use of the
5          ; lead-in type terminal functions.
6          ; The form of the EMT argument block is:
7          ;
8          ;       .BYTE    0,152
9          ;       .WORD    function_value
10         ;       .WORD    control_value
11         ;
12         ; Where "function_value" corresponds to the value of the ascii character
13         ; that would be used with the lead-in sequence, and "control_value"
14         ; corresponds to the letter that would be sent as an argument to some
15         ; of the control functions.
16         ;
17 006120 013702 0000020  TTEMT:  MOV     EMTBLK+2,R2      ;Get function_value
18 006124 162702 000101    SUB     #'A,R2              ;Convert letter into index value
19 006130 100416          BMI     9$                      ;Br if not legal value
20 006132 020227 0000000    CMP     R2,#MAXCC            ;Is the letter ok?
21 006136 103013          BHI     9$                      ;Br if too big
22 006140 006302          ASL     R2                      ;Convert to word table index
23 006142 013700 0000040    MOV     EMTBLK+4,R0          ;Get control_value
24 006146 016237 006176' 006160'  MOV     TTFUN(R2),1$    ;Set address of processing routine
25 006154 004737 0000000    CALL    OVRHC                ;Call overlay handler
26 006160 000000          1$:   .WORD    0                ;Store address of routine to call here
27 006162 000137 0000000    JMP     EMTXIT              ;Finished
28
29         ; Error -- Invalid function value
30         ;
31 006166 012700 000001    9$:   MOV     #1,R0            ;Return error code 1
32 006172 000137 0000000    JMP     SETERR

```

1					
2					
3					
4	006176	00000000			
5	006200	00000000			
6	006202	00000000			
7	006204	00000000			
8	006206	00000000			
9	006210	00000000			
10	006212	00000000			
11	006214	00000000			
12	006216	00000000			
13	006220	00000000			
14	006222	00000000			
15	006224	00000000			
16	006226	00000000			
17	006230	00000000			
18	006232	00000000			
19	006234	00000000			
20	006236	00000000			
21	006240	00000000			
22	006242	00000000			
23	006244	00000000			
24	006246	00000000			
25	006250	00000000			
26	006252	00000000			
27	006254	00000000			
28	006256	00000000			
29	006260	00000000			

```

-----
; Vector of addresses of processing routines
;
; IFUN: .WORD SETRBF      ; A
;        .WORD CMDB       ; B
;        .WORD CMDC       ; C
;        .WORD GTSPAC     ; D
;        .WORD CMDE       ; E
;        .WORD CMDF       ; F
;        .WORD CMDG       ; G
;        .WORD CMDH       ; H
;        .WORD CMDI       ; I
;        .WORD CMDJ       ; J
;        .WORD CMDK       ; K
;        .WORD CMDL       ; L
;        .WORD CMDM       ; M
;        .WORD CMDN       ; N
;        .WORD CMDO       ; O
;        .WORD RSSPAC     ; P
;        .WORD SFWAC      ; Q
;        .WORD CMDR       ; R
;        .WORD CMDS       ; S
;        .WORD CMDT       ; T
;        .WORD CMDU       ; U
;        .WORD SFWL       ; V
;        .WORD CMDW       ; W
;        .WORD CMDX       ; X
;        .WORD CMDY       ; Y
;        .WORD CMDZ       ; Z

```

SNDMSG -- Send a message to a line

```

1          .SBTTL  SNDMSG -- Send a message to a line
2          ;-----
3          ; Send an asciz message to a line.
4          ;
5          ; EMT argument block format:
6          ;
7          ;     .BYTE  Sub-function,127
8          ;     .WORD  Line-number
9          ;     .WORD  Buffer-address
10         ;
11         ; Where Sub-function is:
12         ;           No-gag-override    Gag-override
13         ; Hang           0              1
14         ; Error         2              3
15         ;
16 006262 032737 0000000 0000000 SNDMSG: BIT    #PO$SND,PRIVCO ; Are we authorized to send messages?
17 006270 001003          BNE      4$          ; Br if yes
18 006272 005000          CLR      R0          ; Return error code 0
19 006274 000137 0000000          JMP      SETERR
20 006300 032737 0000000 0000000 4$: BIT    #PO$OPR,PRIVCO ; Does user have OPER privilege?
21 006306 001003          BNE      5$          ; Br if yes
22 006310 042737 0000001 0000000          BIC    #1,EMTBLK ; Cannot override gag unless oper privilege
23         ;
24         ; See if a valid job number was specified.
25         ;
26 006316 013703 0000020          5$: MOV     EMTBLK+2,R3 ; GET NUMBER OF LINE TO RECEIVE MESSAGE
27 006322 006303          ASL      R3          ; CONVERT TO JOB INDEX NUMBER
28 006324 020327 0000002          CMP     R3,#2 ; CAN'T BE LESS THAN 2
29 006330 103473          BLO      9$          ; BR IF TOO SMALL
30 006332 020327 0000000          CMP     R3,#LSTSL ; MAKE SURE NOT TOO LARGE
31 006336 101070          BHI      9$          ; BR IF TOO LARGE
32 006340 032763 0000000 0000000          BIT    ##DILUP,LSW(R3) ; IS JOB LOGGED ON?
33 006346 001464          BEQ      9$          ; BR IF NOT
34 006350 032763 0000000 0000000          BIT    ##DETCH,LSW(R3) ; IS JOB A DETACH LINE?
35 006356 001060          BNE      9$          ; BR - DETACH LINES CAN'T RECEIVE
36 006360 032763 0000000 0000000          BIT    ##TTGAG,LSW7(R3) ; IS LINE GAGGED?
37 006366 001414          BEQ      3$          ; BR IF NOT
38 006370 032763 0000000 0000000          BIT    ##INKMN,LSW4(R3) ; IS KMON RUNNING NOW?
39 006376 001010          BNE      3$          ; BR IF YES
40 006400 032737 0000001 0000000          BIT    #1,EMTBLK ; DOES HE WANT TO OVERRIDE GAG?
41 006406 001004          BNE      3$          ; BR IF YES
42 006410 012700 0000001          MOV     #1,R0 ; RETURN ERROR CODE 1 IF LINE IS GAGGED
43 006414 000137 0000000          JMP      SETERR
44         ;
45         ; Get a free message buffer.
46         ;
47 006420 004737 006526'          3$: CALL    GTMSBF ; GET A FREE MESSAGE BUFFER
48 006424 103004          BCC      6$          ; CONTINUE IF NO ERROR
49 006426 012700 0000002          MOV     #2,R0 ; RETURN ERROR CODE 2 IF NO MESSAGE BUFFERS
50 006432 000137 0000000          JMP      SETERR
51         ;
52         ; Move message from user's buffer to system message buffer.
53         ;
54 006436 013703 0000040          6$: MOV     EMTBLK+4,R3 ; GET ADDRESS OF USER'S MESSAGE BUFFER
55 006442 010300          MOV     R3,R0 ; VALIDATE ADDRESS
56 006444 004737 0000000          CALL    VALADB
57 006450 010402          MOV     R4,R2 ; GET ADDRESS OF SYSTEM MESSAGE BUFFER BLOCK

```

SNDMSG -- Send a message to a line

```
58 006452 062702 0000000 ADD #SB$TXT,R2 ;POINT TO MESSAGE STORAGE AREA
59 006456 010405 MOV R4,R5
60 006460 062705 0000000 ADD #SB$END,R5 ;POINT TO END OF MESSAGE AREA
61 006464 004737 0000000 1$: CALL GETUCH ;GET CHAR FROM USER'S BUFFER
62 006470 110022 MOVB R0,(R2)+ ;MOVE TO SYSTEM MESSAGE BUFFER
63 006472 001403 BEQ 2$ ;BR IF REACHED END
64 006474 020205 CMP R2,R5 ;REACHED END OF BUFFER SPACE?
65 006476 103772 BLO 1$ ;BR IF NOT
66 006500 105042 CLRB -(R2) ;Store null at end of message
67 ;
68 ; Queue the message for the receiving job.
69 ;
70 006502 013701 0000020 2$: MOV EMTBLK+2,R1 ;GET # OF RECEIVING JOB
71 006506 006301 ASL R1 ;CONVERT TO JOB INDEX #
72 006510 004737 006650' CALL QMSG ;QUEUE MESSAGE FOR JOB
73 ;
74 ; Finished
75 ;
76 006514 000137 0000000 JMP EMTXIT
77 ;
78 ; Invalid line #
79 ;
80 006520 005000 9$: CLR R0 ;RETURN ERROR CODE 0
81 006522 000137 0000000 JMP SETERR
```

```

1          .SBTTL  GTMSBF -- Get free system message buffer
2          ;-----
3          ; GTMSBF is called to get a free system message buffer block.
4          ; If none are available, the job is suspended until one becomes free.
5          ;
6          ; Outputs:
7          ; R4 = Address of message block acquired.
8          ;
9 006526    GTMSBF: DISABL          ; ** DISABLE **
10 006534    005737 0000000        TST      NMUMB          ; ARE THERE ANY AVAILABLE MESSAGE BLOCKS?
11 006540    003004          BGT      3$          ; BR IF SO
12 006542    032737 000002 0000000  BIT      #2,EMTBLK      ; DOES USER WANT TO WAIT?
13 006550    001035          BNE      7$          ; BR IF NOT
14 006552    013704 0000000        3$:  MOV     SNMSHD,R4      ; GET ADDRESS OF 1ST FREE BLOCK
15 006556    001013          BNE      1$          ; BR IF OK
16 006560    032737 000002 0000000  BIT      #2,EMTBLK      ; DOES USER WANT TO WAIT?
17 006566    001026          BNE      7$          ; BR IF NOT
18          ;
19          ; No message buffers available.
20          ; Suspend job until one becomes available.
21          ;
22 006570    012700 0000000        2$:  MOV     #S$WSMB,R0      ; WAIT FOR MESSAGE BUFFER STATE.
23 006574    004737 0000000          CALL     QNSPND          ; WAIT FOR MESSAGE BUFFER ** ENABLE **
24 006600    004737 0000000          CALL     CHKABT          ; WERE WE ABORTED WHILE ASLEEP?
25 006604    000750          BR       GTMSBF          ; TRY AGAIN
26          ;
27          ; Got a buffer.  Unlink from free chain.
28          ;
29 006606    005337 0000000        1$:  DEC     NMUMB          ; ONE LESS FREE BUFFER
30 006612    016437 0000000 0000000  MOV     SB$LNK(R4),SNMSHD; REMOVE FROM FREE LIST
31 006620          ENABL          ; ** ENABLE **
32 006626    010400          MOV     R4,R0          ; INITIALIZE SB$PNT
33 006630    062700 0000000          ADD     #SB$TXT,R0
34 006634    010064 0000000          MOV     R0,SB$PNT(R4)
35 006640    000241          CLC                   ; SAY NO ERRORS
36 006642    000401          BR       9$
37          ;
38          ; No free message buffers, flag error return
39          ;
40 006644    000261        7$:  SEC
41          ;
42          ; Finished
43          ;
44 006646    000207        9$:  RETURN

```

QMSG -- Queue a message for a job

```

1          .SBTTL  QMSG  -- Queue a message for a job
2          ;-----
3          ; QMSG is called to queue a system message for a job.
4          ;
5          ; Inputs:
6          ;   R1 = Job number that is to receive message.
7          ;   R4 = Address of system message buffer.
8          ;
9 006650    010146      QMSG:  MOV      R1,-(SP)
10         ;
11         ; Add message buffer to queue for line
12         ;
13 006652         DISABL          ;** DISABLE **
14 006660    016101    0000000    MOV      LNPRIM(R1),R1  ;OBTAIN THE CURRENT PRIMARY LINE NUMBER
15 006664    016100    0000000    MOV      LMSGBF(R1),R0  ;GET ADDRESS OF HEAD OF MESSAGE LIST FOR JOB
16 006670    001411         BEQ      3$                    ;BR IF NO MESSAGES QUEUED FOR JOB
17 006672    005760    0000000    1$:  TST      SB$LNK(R0)  ;IS THIS LAST ENTRY IN LIST?
18 006676    001403         BEQ      2$                    ;BR IF YES
19 006700    016000    0000000    MOV      SB$LNK(R0),R0  ;CHAIN TO NEXT QUEUED MESSAGE BLOCK
20 006704    000772         BR       1$
21 006706    010460    0000000    2$:  MOV      R4,SB$LNK(R0) ;ADD OUR MESSAGE TO TAIL OF LIST
22 006712    000402         BR       4$
23 006714    010461    0000000    3$:  MOV      R4,LMSGBF(R1) ;ADD OUR MESSAGE BLOCK TO LIST FOR JOB
24 006720    005064    0000000    4$:  CLR      SB$LNK(R4)  ;SAY WE ARE AT END OF LIST
25 006724         ENABL          ;** ENABLE **
26         ;
27         ; Initiate transmission to the line.
28         ;
29 006732    116101    0000000    MOV      LNMAP(R1),R1  ;GET VIRTUAL LINE #
30 006736    004777    0000000    CALL     @TRNSTR      ;INITIATE TRANSMISSION
31         ;
32         ; Finished
33         ;
34 006742    012601         MOV      (SP)+,R1
35 006744    000207         RETURN

```

ASTXIT -- Exit from completion routine

```

1          .SBTTL  ASTXIT -- Exit from completion routine
2          ;-----
3          ; The ASTXIT EMT is used to exit from a user completion routine.
4          ;
5 006746   ASTXIT:
6          ;
7          ; Set up information that will sidetrack control in EMTXIT.
8          ;
9 006746   013702   0000000   MOV     EMTCAD,R2      ;Get pointer to top entry on return stack
10 006752   020227   0000000   CMP     R2,#EMTCAS    ;Were we expecting a completion routine exit?
11 006756   001002           BNE     1$             ;Br if yes
12 006760   000137   0000000   JMP     BADEMT      ;Illegal EMT if nothing on cpl rtn return stk
13 006764   012237   0000000   1$:  MOV     (R2)+,EMTRAD    ;Set return address for compl routine
14 006770   010237   0000000       MOV     R2,EMTCAD    ;Save new stack pointer
15 006774   000137   0000000       JMP     EMTXIT      ;EMTXIT will do the actual exit for us

```

.SPCPS -- Alter exit address from a completion routine

```

1          .SBTTL .SPCPS -- Alter exit address from a completion routine
2          ;-----
3          ; The .SPCPS EMT is used to set a address that is used to control the
4          ; mainline code execution on exit from a completion routine.
5          ; The real work to accomplish this is done in SYSXIT.
6          ;
7 007000 105737 0000000 ESPCPS: TSTB CURCP          ;ARE WE IN A COMPLETION ROUTINE NOW?
8 007004 001003          BNE 1$          ;BR IF YES
9 007006 005000          CLR R0          ;RETURN ERROR CODE 0 IF NOT
10 007010 000137 0000000 JMP SETERR
11 007014 005737 0000000 1$: TST SPCPS          ;IS ANOTHER .SPCPS PENDING NOW?
12 007020 001404          BEQ 2$          ;BR IF NOT
13 007022 012700 000001  MOV #1,R0          ;RETURN ERROR CODE 1 IF YES
14 007026 000137 0000000 JMP SETERR
15 007032 013700 0000020 2$: MOV EMTBLK+2,R0      ;GET ADDRESS OF USER'S INFORMATION BLOCK
16 007036 004737 0000000 CALL VALADW      ;VALIDATE THE ADDRESS
17 007042 010037 0000000 MOV R0,SPCPS      ;SAVE ADDRESS OF INFO BLOCK
18 007046 000137 0000000 JMP EMTXIT      ;EXIT -- SYSXIT WILL DO REAL PROCESSING

```

```

1
2
3
4
5
6
7
8 007052 120427 000376
9 007056 001011
10 007060 013704 000000G
11 007064 106564 000002
12 007070 012600
13 007072 000300
14 007074 120027 000373
15 007100 001402
16 007102 000137 000000G
17
18
19
20 007106 010437 000000G
21 007112 012737 177750 000000G
22 007120 004737 000000G

      .SBTTL  EMT376 -- EMT 376 Processing
      -----
      ; Process EMT 376
      ;
      ; Inputs:
      ;   R4 = EMT function code.
      ;
EMT376: CMPB    R4,#376      ; Is this an EMT 376?
        BNE     1$         ; Br if not
        MOV     EMTADR,R4   ; Get address of EMT instruction
        MFPD    2(R4)       ; Get value that follows instruction
        MOV     (SP)+,R0    ; Get code word
        SWAB    R0         ; Get code byte to low-order
        CMPB    R0,#373    ; Overlay I/O error?
        BEQ     2$         ; Br if yes
        JMP     BADEMT     ; Invalid EMT if not
1$:
      ; Overlay I/O error
      ;
2$:     MOV     R4,ABRTAD    ; Set address of abort
        MOV     #-30,ABRTCD ; Set abort code
        CALL    STOP       ; Abort the job

```

```

1          .SBTTL  JBINFO -- Get information about a specific job
2          ;-----
3          ; The JBINFO EMT is used to obtain information about a specific job.
4          ; The form of the EMT argument block is:
5          ; +-----+
6          ; | 144 | 0 |
7          ; +-----+
8          ; |Sub Fun. | Job # |
9          ; +-----+
10         ; | Result buf addr |
11         ; +-----+
12         ;
13         ; Get Subfunction number and check its range
14         ;
15 007124 113705 0000030 JBINFO: MOVB EMTBLK+3,R5 ;GET SUB-FUNCTION NUMBER
16 007130 020527 000011      CMP R5,#MXJIFN ;IS IT TOO BIG?
17 007134 101404      BLOS 1$ ;BR IF OK
18 007136 012700 000001      MOV #1,R0 ;ERROR CODE 1 IF TOO BIG
19 007142 000137 0000000      JMP SETERR
20         ;
21         ; Subfunction number is valid.
22         ; Check job number.
23         ;
24 007146 113701 0000020 1$: MOVB EMTBLK+2,R1 ;GET SPECIFIED JOB NUMBER
25 007152 001404      BEQ 2$ ;ZERO IS INVALID
26 007154 006301      ASL R1 ;CONVERT TO LINE INDEX NUMBER
27 007156 020127 0000000      CMP R1,#LSTSL ;IS IT A VALID NUMBER?
28 007162 101404      BLOS 3$ ;BR IF YES
29 007164 012700 000002 2$: MOV #2,R0 ;ERROR 2 IF INVALID LINE NUMBER
30 007170 000137 0000000      JMP SETERR
31         ;
32         ; See if line is currently logged on
33         ;
34 007174 032761 0000000 0000000 3$: BIT ##KINIT,LSW(R1) ;IS THE LINE LOGGED ON NOW?
35 007202 001003      BNE 4$ ;BR IF YES
36 007204 005000      CLR R0 ;ERROR # 0 IF NOT
37 007206 000137 0000000      JMP SETERR
38         ;
39         ; Check the address specified for the user's result buffer
40         ;
41 007212 013704 0000040 4$: MOV EMTBLK+4,R4 ;GET USER'S VIRTUAL ADDRESS
42 007216 010400      MOV R4,R0
43 007220 004737 0000000      CALL VALADW ;MAKE SURE ADDRESS IS REASONABLE
44         ;
45         ; Everything looks good.
46         ; Branch off to processing routine based on sub-function number.
47         ; At this point the following registers are set up:
48         ; R1 = Line index number for job about which information is desired.
49         ; R4 = Address of user's buffer where result is to be placed.
50         ; R5 = Sub function index number.
51         ;
52 007224 006305      ASL R5 ;CONVERT SUB-FUNCTION # TO WORD INDEX
53 007226 000175 007562'      JMP @JIJMPX(R5) ;ENTER PROCESSING ROUTINE
54         ;
55         ; #0 -- Get job status flags
56         ;
57 007232 005000 JISTAT: CLR R0 ;DEVELOP FLAGS IN R0

```

```

58          ; See if job is detached or virtual
59 007234 020127 0000000  CMP      R1,#LSTPL      ; PRIMARY LINE?
60 007240 101410          BLOS     1$          ; BR IF YES
61 007242 020127 0000000  CMP      R1,#LSTD L      ; DETACHED LINE?
62 007246 101403          BLOS     2$          ; BR IF YES
63 007250 052700 0000000  BIS      #JIVLN,R0      ; SET VIRTUAL-LINE FLAG
64 007254 000402          BR       1$
65 007256 052700 0000000  2$: BIS      #JIDL N,R0      ; SET DETACHED-LINE FLAG
66          ; See if job is locked in memory
67 007262 032761 0000000 0000000 1$: BIT      ##MLOCK,LSW6(R1); IS JOB LOCKED IN MEMORY?
68 007270 001402          BEQ      4$          ; BR IF NOT
69 007272 052700 0000000  BIS      #JIMLOK,R0      ; SET MEMORY-LOCKED FLAG
70          ; See if job has operator privilege
71 007276 010046          4$: MOV      R0,-(SP)      ; SAVE OUR STATUS CODES
72 007300          OCALL     GETCXT      ; GET EXCLUSIVE ACCESS TO CONTEXT BUFFER
73 007306 103002          BCC      6$          ; BR IF GOT IT
74 007310 004737 0000000  CALL     STOP      ; JOB WAS ABORTED WHILE WAITING
75 007314 010102          6$: MOV      R1,R2      ; GET INDEX # OF JOB WE ARE COPYING FROM
76 007316 012703 0000000  MOV      #PRIVCO,R3      ; ADDR OF INFO TO GET
77 007322 012700 0000002  MOV      #2,R0      ; GET # BYTES TO COPY
78 007326          OCALL     REDCXT      ; GET INFO FROM CONTEXT BLOCK OF OTHER JOB
79 007334 032777 0000000 0000000  BIT      #PO$SYS,@CXTBUF ; IS THIS A PRIVILEGED JOB?
80 007342 001402          BEQ      5$          ; BR IF NOT
81 007344 052716 0000000  BIS      #JIPRIV,(SP)      ; SET PRIVILEGE FLAG
82 007350          5$: OCALL     FRECXT      ; FREE THE CONTEXT BUFFER
83 007356 012600          MOV      (SP)+,R0      ; GET STATUS FLAGS
84 007360 000422          BR       JIRET1      ; GO RETURN VALUE
85          ;
86          ; #1 -- Get job's run state
87          ;
88 007362 012700 007606'  JIRUN: MOV      #JIRNST,R0      ; Point to state table
89 007366 116102 0000000  MOVVB   LSTATE(R1),R2      ; Get job's current execution state
90 007372 120220          1$: CMPB    R2,(R0)+      ; Search for state in table
91 007374 001406          BEQ      2$          ; Br if found it
92 007376 005200          INC      R0      ; Skip over value
93 007400 020027 007672'  CMP      R0,#JIRNND      ; Checked all entries?
94 007404 103772          BLD      1$          ; Br if yes
95 007406 005000          CLR      R0      ; Don't recognize state, return 0
96 007410 000406          BR       JIRET1      ; Return the value
97 007412 111000          2$: MOVVB   (R0),R0      ; Get value to return
98 007414 000404          BR       JIRET1      ; Return the value
99          ;
100         ; #2 -- Get amount of memory in use by job
101         ;
102 007416 016100 0000000  JIMEM: MOV      LNBLKS(R1),R0      ; GET # 256-WORD BLOCKS USED BY JOB
103 007422 066100 0000000  ADD      LNSBLK(R1),R0      ; ADD # BLOCKS USED BY PLAS REGIONS
104 007426 010046          JIRET1: MOV     R0,-(SP)      ; STORE VALUE INTO USER'S BUFFER
105 007430 106614          MTPD     (R4)
106 007432 000137 0000000  JMP      EMTXIT      ; FINISHED WITH EMT
107         ;
108         ; #3 -- Get job connect time (minutes)
109         ;
110 007436 013700 0000000  JICONT: MOV     MINTIM,R0      ; GET CURRENT SYSTEM UP-TIME
111 007442 166100 0000000  SUB      LCONTM(R1),R0      ; GET # MINUTES JOB HAS BEEN CONNECTED
112 007446 000767          BR       JIRET1
113         ;
114         ; #4 -- Get base 256-word block # assigned to job area

```

JBINFO -- Get information about a specific job

```

115
116 007450 016100 0000000 JIBASE: MOV     LBASE(R1),R0    ;GET BASE BLOCK #
117 007454 000764          BR      JIRET1
118
119      ; #5 -- Get name of program being run by job
120
121 007456 016102 0000000 JIPROG: MOV     LPRG1(R1),R2    ;GET FIRST 3 CHARS OF NAME
122 007462 016103 0000000      MOV     LPRG2(R1),R3    ;GET SECOND 3 CHARS OF NAME
123 007466 000411          BR      JIRET2
124
125      ; #6 -- Get project/programmer number
126
127 007470 016102 0000000 JIPPN:  MOV     LPROJ(R1),R2    ;GET PROJECT NUMBER
128 007474 016103 0000000      MOV     LPROG(R1),R3    ;GET PROGRAMMER NUMBER
129 007500 000404          BR      JIRET2
130
131      ; #7 -- Get CPU time used by job
132
133 007502 016102 0000000 JICPU:  MOV     LCPUHI(R1),R2    ;GET HIGH-ORDER CPU TIME
134 007506 016103 0000000      MOV     LCPULO(R1),R3    ;GET LOW-ORDER CPU TIME
135 007512 010246          JIRET2: MOV     R2,-(SP)      ;RETURN FIRST WORD OF RESULT
136 007514 106624          MTPD      (R4)+
137 007516 010346          MOV     R3,-(SP)      ;RETURN SECOND WORD OF RESULT
138 007520 106624          MTPD      (R4)+
139 007522 000137 0000000      JMP      EMTXIT          ;FINISHED WITH EMT
140
141      ; #8 -- Job execution priority
142
143 007526 116100 0000000 JIPRIO: MOVB     LPRI(R1),R0    ;Get current execution priority for job
144 007532 000735          BR      JIRET1          ;Return
145
146      ; #9 -- Job name
147
148 007534 010105          JINAME: MOV     R1,R5        ;Copy job index number
149 007536 012702 0000006      MOV     #6,R2          ;6 words per name
150 007542 070502          MUL      R2,R5            ;Name should be 12 bytes long
151 007544 062705 0000000      ADD     #LUNAME,R5      ;Convert to name table index
152 007550 012546          1$:  MOV     (R5)+,-(SP)    ;Fetch next two bytes of name
153 007552 106624          MTPD      (R4)+            ;Move to user's buffer
154 007554 077203          SOB      R2,1$             ;Loop through whole name
155 007556 000137 0000000      JMP      EMTXIT          ;Finished with EMT
156
157      ; Branch table used to enter job information routines
158
159 007562 007232'          JIUMPX: .WORD    JISTAT      ; 0 -- GET JOB STATE FLAGS
160 007564 007362'          .WORD    JIRUN          ; 1 -- GET JOB RUN STATE CODE
161 007566 007416'          .WORD    JIMEM          ; 2 -- GET AMT OF MEMORY USED BY JOB
162 007570 007436'          .WORD    JICONT          ; 3 -- GET CONNECT TIME FOR JOB
163 007572 007450'          .WORD    JIBASE          ; 4 -- GET BASE ADDRESS FOR JOB
164 007574 007456'          .WORD    JIPROG          ; 5 -- GET NAME OF PROGRAM BEING RUN
165 007576 007470'          .WORD    JIPPN          ; 6 -- GET PROJECT/PROGRAMMER #
166 007600 007502'          .WORD    JICPU          ; 7 -- GET CPU TIME USED BY JOB
167 007602 007526'          .WORD    JIPRIO          ; 8 -- GET JOB EXECUTION PRIORITY
168 007604 007534'          .WORD    JINAME          ; 9 -- GET JOB NAME
169      000011          MXJIFN = <<. -JIUMPX>/2>-1

```

```

1      ;
2      ; Table used to convert TSX scheduler state codes into
3      ; externally defined run states.
4      ;
5 007606      0000      001      JIRNST: . BYTE      S$TWFN,1      ; Timed wait completion
6 007610      0000      001      . BYTE      S$OTLO,1      ; Terminal output buffer almost empty
7 007612      0000      001      . BYTE      S$IOFN,1      ; I/O completion
8 007614      0000      002      . BYTE      S$CPU,2      ; Normal priority execution
9 007616      0000      003      . BYTE      S$LOW,3      ; Fixed low priority execution
10 007620      0000      004      . BYTE      S$INWT,4      ; Waiting for terminal input
11 007622      0000      005      . BYTE      S$OTWT,5      ; Waiting for terminal output buffer space
12 007624      0000      006      . BYTE      S$TMWT,6      ; Waiting for timed interval
13 007626      0000      007      . BYTE      S$SPND,7      ; Doing . SPND
14 007630      0000      010      . BYTE      S$SFWT,8      ; Shared file wait
15 007632      0000      011      . BYTE      S$MSWT,9      ; Waiting for message
16 007634      0000      012      . BYTE      S$QUSR,10      ; Waiting for USR for file operation
17 007636      0000      013      . BYTE      S$IQWT,11      ; Waiting for I/O operation
18 007640      0000      013      . BYTE      S$NEDQ,11      ; Waiting for I/O queue element
19 007642      0000      013      . BYTE      S$QMID,11      ; Waiting for mapped I/O buffer
20 007644      0000      013      . BYTE      S$QCCB,11      ; Waiting for cache control block
21 007646      0000      013      . BYTE      S$QSPD,11      ; Waiting for special device data base
22 007650      0000      013      . BYTE      S$WSMB,11      ; Waiting for system message buffer
23 007652      0000      014      . BYTE      S$SPDB,12      ; Waiting for spooled device control block
24 007654      0000      014      . BYTE      S$SPCB,12      ; Waiting for spooled device control block
25 007656      0000      015      . BYTE      S$TTSC,13      ; Single char terminal input complete
26 007660      0000      015      . BYTE      S$TTFN,13      ; Non single char terminal input complete
27 007662      0000      015      . BYTE      S$OTFN,13      ; Terminal buffer empty
28 007664      0000      015      . BYTE      S$HICP,13      ; Interactive job execution
29 007666      0000      016      . BYTE      S$RT,14      ; Fixed high priority
30 007670      0000      017      . BYTE      S$WFM,15      ; Waiting for memory expansion
31 007672      JIRNND:
32      . EVEN      ; End of table

```

Get terminal type code

```

1
2
3
4
5
6
7
8 007672 012702 007756'
9 007676 016104 0000000
10 007702 116100 0000000
11 007706 032760 0000000 0000000
12 007714 001402
13 007716 012704 0000000
14 007722 020422
15 007724 001405
16 007726 020227 010002'
17 007732 101773
18 007734 005002
19 007736 000403
20 007740 162702 007756'
21 007744 006202
22 007746 010237 0000000
23 007752 000137 0000000
24
25
26
27 007756 0000000
28 007760 0000000
29 007762 0000000
30 007764 0000000
31 007766 0000000
32 007770 0000000
33 007772 0000000
34 007774 0000000
35 007776 0000000
36 010000 0000000
37 010002
38
39
40
41
42
43
44 010002 013700 0000020
45 010006 004737 0000000
46 010012 016146 0000000
47 010016 106620
48 010020 016146 0000000
49 010024 106620
50 010026 000137 0000000

      .SBTTL  Get terminal type code
      -----
; This EMT returns in R0 a numeric value that indicates the type of
; terminal from which the job is being run.
; The terminal type is declared in the line definition in TSGEN or
; by the SET TERMINAL keyboard command.
;
TRMEMT: MOV      #LOTRM,R2      ;POINT TO START OF TERMINAL TYPE TABLE
        MOV      LTRMTP(R1),R4  ;GET TERMINAL TYPE FLAGS FOR OUR JOB
        MOVB     LNPRIM(R1),R0  ;Get out primary line number
        BIT      ##V52EM,LSW11(R0);Are we emulating a VT52?
        BEQ      1$            ;Br if not
        MOV      #VT52,R4      ;Say terminal is a VT52
1$:      CMP      R4,(R2)+      ;SEARCH FOR MATCHING PATTERN IN TABLE
        BEQ      2$            ;BR IF FOUND
        CMP      R2,#HITRM     ;HAVE WE SEARCHED ALL OF TABLE?
        BLOS     1$            ;BR IF NOT
        CLR      R2            ;SAY UNKNOWN TERMINAL TYPE
        BR       3$
2$:      SUB      #LOTRM,R2     ;CONVERT TABLE OFFSET TO TERMINAL CODE NUMBER
        ASR      R2
3$:      MOV      R2,UR0        ;RETURN TERMINAL TYPE CODE # IN USER'S R0
        JMP      EMTXIT        ;FINISHED
;
; Table to translate terminal type flags into code number.
;
LOTRM:  .WORD     VT52          ; 1 - VT52
        .WORD     VT100        ; 2 - VT100
        .WORD     HAZEL        ; 3 - Hazeltine
        .WORD     ADM3A        ; 4 - ADM3A
        .WORD     LA36         ; 5 - LA36
        .WORD     LA120        ; 6 - LA120
        .WORD     DIABLO       ; 7 - Diablo
        .WORD     QUME         ; 8 - Qume
        .WORD     VT2007       ; 9 - VT200 with 7 bit control codes
        .WORD     VT2008       ; 10 - VT200 with 8 bit control codes
;
HITRM:
;
      .SBTTL  Get Project-Programmer number
      -----
; This emt is used to obtain the current Project-Programmer number
; that a job is running under.
;
GETPPN: MOV      EMTBLK+2,R0    ;GET ADDRESS OF CELL TO RECEIVE PPN
        CALL     VALADW        ;VALIDATE THE ADDRESS
        MOV      LPROJ(R1),-(SP);MOVE PROJECT # TO USER'S BUFFER
        MTPD     (R0)+
        MOV      LPROG(R1),-(SP);
        MTPD     (R0)+        ;MOVE PROGRAMMER # TO USER'S BUFFER
        JMP      EMTXIT        ;FINISHED

```

```

1          .SBTTL  MISC. TSX EMT'S
2
3          ;-----
4          ; THE .LNUM EMT RETURNS IN R0 THE NUMBER OF THE TSX
5          ; LINE WHICH THE USER IS CONNECTED TO, OR THE LINE NUMBER
6          ; OF THE SPECIFIED SUBPROCESS
7
8 010032 105737 0000000  XGTLN:  TSTB  EMTBLK      ;WANT CURRENT LINE OR RELATIVE
9 010036 001440          BEQ    1$          ;BR IF CURRENT
10
11          ; THIS IS A REQUEST TO RETURN THE REAL LINE NUMBER OF A SUBPROCESS
12
13 010040 020127 0000000          CMP    R1,#LSTPL      ;ARE WE A PRIMARY LINE?
14 010044 101406          BLOS   4$          ;BR IF OK
15 010046 020127 0000000          CMP    R1,#LSTD L      ;ARE WE A VIRTUAL LINE?
16 010052 101003          BHI    4$          ;BR IF OK
17 010054 012700 0000002          MOV    #2,R0          ;DETACHED JOB'S DON'T HAVE SUBPROCESSES
18 010060 000425          BR      6$          ;GO COMPLAIN ABOUT IT
19 010062 016101 0000000  4$:  MOV    LNPRIM(R1),R1      ;GET OUR PARENT INDEX
20 010066 013705 0000020          MOV    EMTBLK+2,R5      ;WANTS LINE NUMBER FOR THIS SUBPROCESS
21 010072 001002          BNE    2$          ;BR IF NOT ASKING FOR PARENT
22 010074 010105          MOV    R1,R5          ;COPY PARENT'S PROCESS INDEX
23 010076 000411          BR      3$          ;DON'T NEED SUBPROCESS MAPPING
24 010100 020527 0000000  2$:  CMP    R5,#MAXSEC      ;VALID SUBPROCESS NUMBER?
25 010104 101403          BLOS   5$          ;BR IF OK
26 010106 012700 0000001          MOV    #1,R0          ;SIGNAL INVALID SUBPROCESS #
27 010112 000410          BR      6$          ;AND GO REPORT ERROR
28 010114 066105 0000000  5$:  ADD    LSECPT(R1),R5      ;POINT TO PRIMARY LINE'S SUBPROCESS TABLE
29 010120 114505          MOVB   -(R5),R5          ;GET SPECIFIED LINE'S INDEX (^1 IS OFFSET 0)
30 010122 006205          3$:  ASR    R5          ;CONVERT INDEX TO LINE #
31 010124 010537 0000000          MOV    R5,URO          ;PASS TO USER
32 010130 001012          BNE    XJNOP          ;IF IN USE, RETURN LINE # TO USER
33 010132 005000          CLR    R0          ;ELSE REPORT LINE NOT ACTIVE
34 010134 000137 0000000  6$:  JMP    SETERR          ;RETURN WITH ERROR
35
36          ; THIS IS A REQUEST TO GET OUR CURRENT LINE NUMBER
37
38 010140 006201          1$:  ASR    R1          ;CONVERT TO LINE #
39 010142 010137 0000000          MOV    R1,URO          ;RETURN TO USER IN R0
40 010146 000403          BR      XJNOP
41
42          ;-----
43          ; THE .SPLSP EMT RETURNS IN R0 THE NUMBER OF FREE BLOCKS
44          ; IN THE SPOOL FILE.
45
46 010150 013737 0000000 0000000 XSPLSP: MOV    NFRESB,URO      ;RETURN # OF BLOCKS TO USER IN R0
47 010156 000137 0000000          XJNOP:  JMP    EMTXIT
48
49          ;-----
50          ; SET/RESET ODT CHARACTER ACTIVATION MODE
51
52 010162 105737 0000000          XODTMD: TSTB  EMTBLK      ;SET OR RESET MODE?
53 010166 001407          BEQ    1$          ;BR IF RESET
54 010170 052761 0000000 0000000          BIS    #ODTMD,LSW4(R1);TURN ON ODT MODE
55 010176 052761 0000000 0000000          BIS    #DBGMD,LSW6(R1);SAY WE ARE IN DEBUGGER TERMINAL CONTROL MODE
56 010204 000764          BR      XJNOP
57 010206 042761 0000000 0000000 1$:  BIC    #ODTMD,LSW4(R1);TURN OFF ODT MODE

```

```
58 010214 042761 0000000 0000000 BIC    #$DBGMD,LSW6(R1);RESET DEBUGGER TERMINAL MODE
59 010222 000755 BR      XJNOP
60 ;
61 ;-----
62 ; Check to see if any input activation characters are pending.
63 ; Return with c-flag set if not.
64 ;
65 010224 005761 0000000 CKINAC: TST    LACTIV(R1)    ;ANY PENDING ACTIVATION CHARS?
66 010230 001002 BNE      1$          ;BR IF YES
67 010232 000137 0000000 JMP      SETC      ;RETURN WITH C-FLAG SET
68 010236 000137 0000000 1$: JMP      EMTXIT    ;RETURN WITH C-FLAG CLEAR
69 ;
70 ;-----
71 ; Return in R0 the TSX license number of this site.
72 ;
73 010242 013737 0000000 0000000 GTLICN: MOV    TSXSIT,URO    ;PUT LICENSE # IN USER'S R0
74 010250 000742 BR      XJNOP    ;RETURN
```

```

1          ; -----
2          ; Set interactive or non-interactive mode for current program.
3          ;
4 010252  005737  0000020  INTEMP: TST      EMTBLK+2      ;Set or reset interactive mode?
5 010256  001404          BEQ      1$          ;Reset interactive mode
6          ;
7          ; Set interactive mode
8          ;
9 010260  042761  0000000 0000000  BIC      ##NOINT,LSW7(R1);Clear non-interactive mode flag
10 010266  000405          BR      9$
11         ;
12         ; Reset interactive mode
13         ;
14 010270  052761  0000000 0000000 1$:  BIS      ##NOINT,LSW7(R1);Run job non-interactively
15 010276  005061  0000000          CLR      LITIME(R1)      ;Clear interactive timer
16         ;
17         ; Finished
18         ;
19 010302  000137  0000000 9$:  JMP      EMTXIT          ;Finished

```

Return system (swap) file specification

```

1          .SBTTL  Return system (swap) file specification
2          ;-----
3          ; Return dev:filspc.ext for system swap files
4          ;
5          ; The form of the EMT argument block is:
6          ;   .BYTE   file_index,164
7          ;   .WORD   buffer_address
8          ;
9 010306 113705 0000000 SYFEMT: MOVB    EMTBLK,R5      ;GET SYSTEM FILE INDEX
10 010312 020527 0000004      CMP      R5,#MXSYFN      ;VALID FILE INDEX NUMBER?
11 010316 101403      BLOS     1$                    ;BRANCH IF OK
12 010320 005000      CLR      R0                      ;ERROR CODE 0 IF BAD INDEX
13 010322 000137 0000000      JMP      SETERR
14          ;
15          ; Convert system file index number to table offset
16          ;
17 010326 006305      1$:      ASL      R5                ;CONVERT INDEX TO TABLE OFFSET
18 010330 016504 010364'      MOV      SYFTBL(R5),R4    ;GET POINTER TO FILE NAME
19          ;
20          ; Get and validate user's return buffer
21          ;
22 010334 013705 0000020      MOV      EMTBLK+2,R5      ;GET USER'S VIRTUAL ADDRESS
23 010340 010500      MOV      R5,R0
24 010342 004737 0000000      CALL     VALADW           ;MAKE SURE ADDRESS IS OK
25          ;
26          ; Now return 4 RAD50 words of dev:filspc.ext to user buffer
27          ;
28 010346 012700 0000004      MOV      #4,R0            ;MOVE 4 WORDS
29 010352 012446      2$:      MOV      (R4)+,-(SP)      ;GET NEXT RAD50 WORD
30 010354 106625      MTPD     (R5)+                    ;MOVE TO USER BUFFER
31 010356 077003      SOB      R0,2$
32 010360 000137 0000000      JMP      EMTXIT           ;FINISHED
33          ;
34          ; System file specification pointers
35          ;
36 010364 0000000 SYFTBL: .WORD   SWDBLK                ; 0 SWAP FILE
37 010366 0000000      .WORD   SPLBLK                ; 1 SPOOL FILE
38 010370 0000000      .WORD   RSFBLK                ; 2 PLAS REGION SWAP FILE
39 010372 0000000      .WORD   UCLDAT                ; 3 USER DEFINED COMMAND (UCL) FILE
40 010374 0000000      .WORD   INDFIL                ; 4 IND TEMP FILE
41          MXSYFN = <<.-SYFTBL>/2>-1

```

Set transmit/receive speed for a line

```

1          .SBTTL Set transmit/receive speed for a line
2          ;-----
3          ; Set transmit/receive speed for a line.
4          ;
5          ; The form of the EMT argument block is:
6          ; .BYTE    n,154
7          ; .WORD    line_number
8          ; .WORD    speed_code
9          ; where n=0 is set speed,n=1 is reset xoff,n=2 is set dtr,n=3 is clear dtr
10         ;
11 010376   EMTSPD:
12         ;
13         ; Get line number and see if it is valid
14         ;
15 010376   013702 0000020      MOV     EMTBLK+2,R2      ;Get line number
16 010402   006302      ASL      R2                    ;Convert to line index number
17 010404   001001      BNE      1$                    ;Br if line number specified
18 010406   010102      MOV     R1,R2                    ;Use current line
19 010410   020227 0000000      1$:  CMP     R2,#LSTSL      ;Primary or virtual line?
20 010414   101002      BHI      10$                    ;Br if not
21 010416   016202 0000000      MOV     LNPRIM(R2),R2    ;Map virtual line # to real line #
22 010422   020227 0000000      10$:  CMP     R2,#LSTHL      ;Is this a valid line number?
23 010426   101010      BHI      5$                    ;Br if not
24 010430   032762 0000000 0000000  BIT     ##HARD,LSW3(R2) ;Line connected to hardware?
25 010436   001404      BEQ      5$                    ;Br if not
26 010440   032762 0000000 0000000  BIT     ##DEAD,LSW3(R2) ;Is line installed?
27 010446   001404      BEQ      4$                    ;Br if yes
28 010450   012700 0000002      5$:  MOV     #2,R0        ;Return error code 2 for invalid line #
29 010454   000137 0000000      JMP      SETERR
30         ;
31         ; See if we are privileged to perform this operation
32         ;
33 010460   032737 0000000 0000000  4$:  BIT     #P2$TRM,PRIVC2 ;Do we have privilege to change speed?
34 010466   001011      BNE      11$                    ;Br if yes
35 010470   105737 0000000      TSTB    EMTBLK          ;EMT to reset XOFF status or line speed EMT?
36 010474   001402      BEQ      12$                    ;Br if changing line speed
37 010476   020201      CMP     R2,R1                    ;Resetting XOFF for our own line?
38 010500   001404      BEQ      11$                    ;Br if yes
39 010502   012700 0000001      12$:  MOV     #1,R0        ;Return error code 1
40 010506   000137 0000000      JMP      SETERR
41         ;
42         ; See if this is actually the EMT to reset terminal XOFF status
43         ;
44 010512   010201      11$:  MOV     R2,R1                    ;Get line index number to R1
45 010514   123727 0000000 0000001  CMPB    EMTBLK,#1      ;Set line speed, reset XOFF or DTR?
46 010522   002416      BLT      2$                    ;Br if want to set speed
47 010524   001422      BEQ      20$                    ;Br if want to reset XOFF
48         ;
49         ; Want to raise or lower DTR, get current data set status
50         ;
51 010526   004737 0000000      CALL    GETDSS          ;Get data set status into R0
52 010532   042700 0000000      BIC     #MS$DTR,R0      ;Assume we want to lower DTR
53 010536   123727 0000000 0000002  CMPB    EMTBLK,#2      ;Should we raise DTR?
54 010544   001002      BNE      13$                    ;Br if not
55 010546   052700 0000000      BIS     #MS$DTR,R0      ;Set DTR up bit
56 010552   004737 0000000      13$:  CALL    SETDSS          ;Reset data set status
57 010556   000413      BR       30$                    ;And exit

```

Set transmit/receive speed for a line

```
58 ;
59 ; Call hardware-dependent routine to set the line speed
60 ;
61 010560 013700 0000040 2$: MOV EMTBLK+4,R0 ;Get speed code
62 010564 004737 0000000 CALL SETSPD ;Set the line speed
63 010570 000406 BR 30$ ;And exit
64 ;
65 ; Reset XOFF status and start transmitter
66 ;
67 010572 042761 0000000 0000000 20$: BIC #$CTRLS,LSW3(R1);Reset XOFF flag for line
68 010600 004777 0000000 CALL @TRNSTR ;Try to start output to the line
69 010604 000400 BR 30$ ;Finished
70 ;
71 ; Finished
72 ;
73 010606 000137 0000000 30$: JMP EMTXIT ;Finished
```

```

1
2 ; -----
3 ; The .GTTAB emt is an obsolete TSX emt that is supposed to return in R0
4 ; the address of a vector of pointers to tables within TSX.
5 ; This emt is simulated here by moving some info into the low memory
6 ; region of the job (372+) and returning that as the base of the
7 ; pointer vector.
8 010612 012702 000400 XGTTAB: MOV #400,R2 ; STORE VALUES HERE
9 010616 012703 000372 MOV #372,R3 ; STORE VECTOR HERE
10 ; TSLICH -- TSX command lead-in character.
11 010622 113746 0000000 MOV VTSLCH,-(SP) ; GET LEAD-IN CHARACTER
12 010626 106623 MTPD (R3)+ ; MOVE TO JOB SPACE
13 ; Name of running program.
14 010630 010246 MOV R2,-(SP) ; SET POINTER TO PROGRAM NAME
15 010632 106623 MTPD (R3)+ ; IN VECTOR
16 010634 012700 000004 MOV #4,R0 ; GET # WORDS TO MOVE
17 010640 012704 0000000 MOV #RUNDEV,R4 ; POINT TO NAME OF RUNNING PROGRAM
18 010644 012446 1$: MOV (R4)+,-(SP) ; MOVE NAME TO JOB SPACE
19 010646 106622 MTPD (R2)+
20 010650 077003 SOB R0,1$
21 ;
22 ; Tell user that vector is located at 350.
23 ;
24 010652 012737 000350 0000000 MOV #350,U0 ; RETURN VECTOR ADDRESS IN USER'S R0
25 010660 000137 0000000 JMP EMTXIT ; FINISHED

```

```

1          .SBTTL  Access TSX system tables
2          -----
3          ; EMT's to get a table value, store a table value, set bits and clear bits.
4          ; Operator privilege is required to modify a table.
5          ;
6          ; EMT channel byte (byte 0 in arg block) indicates function desired...
7          ;
8          ; 0 ==> Get value from table.
9          ; 1 ==> Store value into table.
10         ; 2 ==> Set bits in table.
11         ; 3 ==> Clear bits in table.
12         ; 4 ==> Special table operation.
13         ;
14 010664 113705 0000000  TBLEMT: MOVB    EMTBLK,R5      ;GET SUB-FUNCTION CODE
15 010670 001412          BEQ      1$              ;BR IF IT IS GET-VALUE
16         ;
17         ; Operator privilege is required to modify a table.
18         ;
19 010672 032737 0000000 0000000  BIT      #P0$SYS,PRIVCO  ;IS THIS USER PRIVILEGED?
20 010700 001004          BNE      2$              ;BR IF YES
21 010702 012700 0000001  MOV      #1,R0          ;RETURN ERROR CODE OF 1 IF NOT
22 010706 000137 0000000  JMP      SETERR
23 010712 013703 0000040  2$:      MOV      EMTBLK+4,R3      ;GET VALUE FOR STORE
24 010716 013702 0000020  1$:      MOV      EMTBLK+2,R2      ;GET TABLE INDEX
25         ;
26         ; If subfunction code = 4 then branch off to special processing routine
27         ;
28 010722 020527 0000004  CMP      R5,#4          ;IS THIS A SPECIAL TABLE OPERATION?
29 010726 001004          BNE      11$             ;BR IF NOT
30 010730 006302          ASL      R2              ;GET WORD TABLE INDEX FOR OPERATION
31 010732 020227 0000000  CMP      R2,#MXSPTF      ;MAKE SURE RANGE IS OK
32 010736 101043          BHI      9$              ;BR IF INVALID
33 010740 000172 011136'  JMP      @SPLTF(R2)        ;ENTER PROCESSING ROUTINE
34         ;
35         ; This is not a special table function
36         ;
37 010744 006302 11$:      ASL      R2              ;CONVERT TO WORD TABLE VALUE
38 010746 002007          BGE      4$              ;BR IF POS TABLE
39         ;
40         ; Negative table index
41         ;
42 010750 005402          NEG      R2              ;MAKE TABLE INDEX POSITIVE
43 010752 020227 0000006  CMP      R2,#MXNGTX      ;IS VALUE TOO BIG?
44 010756 103033          BHI8     9$              ;BR IF TOO BIG
45 010760 016204 011164'  MOV      NEGTAB(R2),R4      ;GET TABLE ADDRESS
46 010764 000406          BR       3$              ;
47         ;
48         ; Positive table index
49         ;
50 010766 020227 0000024  4$:      CMP      R2,#MXPSTX      ;IS TABLE INDEX TOO BIG?
51 010772 103025          BHI8     9$              ;BR IF TOO BIG
52 010774 016204 011140'  MOV      POSTAB(R2),R4      ;GET TABLE ADDRESS
53 011000 060104          ADD      R1,R4            ;INDEX BY LINE NUMBER
54         ;
55         ; Perform requested operation on table
56         ;
57 011002 120527 0000001  3$:      CMPB     R5,#1          ;CHECK SUB-FUNCTION CODE

```

```

58 011006 003006          BGT      5$          ;BR IF BIT SET OR RESET
59 011010 001403          BEQ      6$          ;BR IF STORE VALUE
60                          ; Fetch table value
61 011012 011437 00000006      MOV      @R4,UR0      ;RETURN TABLE VALUE IN USER'S R0
62 011016 000411          BR      10$
63                          ; Store value into table.
64 011020 010314      6$:      MOV      R3,@R4          ;STORE INTO TABLE
65 011022 000407          BR      10$
66 011024 020527 0000002      5$:      CMP      R5,#2          ;BIT SET OR RESET?
67 011030 001002          BNE      7$          ;BR IF BIT CLEAR
68                          ; Set bits in table
69 011032 050314          BIS      R3,@R4          ;SET BITS
70 011034 000402          BR      10$
71                          ; Clear bits in table
72 011036 040314      7$:      BIC      R3,@R4          ;CLEAR BITS IN TABLE
73 011040 000400          BR      10$
74                          ;
75                          ; Finished
76                          ;
77 011042 000137 00000006      10$:      JMP      EMTXIT
78                          ;
79                          ; Invalid table index
80                          ;
81 011046 005000      9$:      CLR      R0          ;RETURN ERROR CODE 0
82 011050 000137 00000006      JMP      SETERR
83                          ;
84                          ; Special table function routines
85                          ;
86                          ; 1 -- Set maximum execution priority for job
87                          ;
88 011054 005703      SETMXP: TST      R3          ;PRIORITY MUST BE > 0
89 011056 003403          BLE      1$          ;BR IF INVALID
90 011060 020327 00000006      CMP      R3,#MAXPRI      ;MAY NOT EXCEED THIS
91 011064 101402          BLOS      2$          ;BR IF OK
92 011066 113703 00000006      1$:      MOVB      VPRIDF,R3      ;IF BAD, SET TO NORMAL PRIORITY
93 011072 110337 00000006      2$:      MOVB      R3,MXJPRI      ;SET MAX PRIORITY FOR JOB
94 011076 120361 00000006      CMPB      R3,LBSPRI(R1)      ;COMPARE WITH CURRENT JOB PRIORITY
95 011102 103013          BHIS      3$          ;BR IF CURRENT PRIORITY IS OK
96 011104 110361 00000006      MOVB      R3,LBSPRI(R1)      ;SET BASE PRIORITY FOR JOB
97 011110 110361 00000006      MOVB      R3,LPRI(R1)      ;SET CURRENT PRIORITY FOR JOB
98 011114 032761 00000006 00000006      BIT      #$VNOTT,LSW(R1) ;Is this a subprocess disconnected from term?
99 011122 001403          BEQ      3$          ;Br if not
100 011124          OCALL      SPPRED          ;Reduce prio of disconnected subprocess
101 011132 000137 00000006      3$:      JMP      EMTXIT          ;FINISHED
102                          ;
103                          ; Jump table for special table functions (subfunction code = 4)
104                          ;
105 011136 011054'      SPLTF: .WORD      SETMXP          ; 0 -- Set maximum job priority
106          000000      MXSPTF =      <.-SPLTF>-2
107                          ;
108                          ; Table of table addresses for positive offsets.
109                          ;
110 011140 00000006      POSTAB: .WORD      LCONTM          ;+00 -- CONNECT TIME
111 011142 00000006          .WORD      LSW          ;+01 -- LSW
112 011144 00000006          .WORD      LSW2          ;+02 -- LSW2
113 011146 00000006          .WORD      LSW3          ;+03 -- LSW3
114 011150 00000006          .WORD      LSW4          ;+04 -- LSW4

```

115	011152	0000000	.WORD	LSW5	;+05 -- LSW5
116	011154	0000000	.WORD	LSW9	;+06 -- LSW9
117	011156	0000000	.WORD	LPROJ	;+07 -- LPROJ
118	011160	0000000	.WORD	LPROG	;+10 -- LPROG
119	011162	0000000	.WORD	LSW2S	;+11 -- LSW2S
120		000024	MXPSTX	=	.-POSTAB
121					
122					; Table of addresses for negative offsets.
123					
124	011164	0000000	NEGTAB:	.WORD	0 ; -00
125	011166	0000000		.WORD	UPPN ; -01 -- PROJECT #
126	011170	0000020		.WORD	UPPN+2 ; -02 -- PROGRAMMER NUMBER
127		000006	MXNGTX	=	.-NEGTAB

Get or set user name

```

1          .SBTTL  Get or set user name
2          ;-----
3          ; Get or set user name for this job, or get or set the program name.
4          ; Byte 0 indicates whether we are getting the user name (0), setting the
5          ; user name (1), getting the program name (2), setting the program
6          ; name (3).
7          ;
8 011172 010105      EMUNAM: MOV      R1,R5          ;GET JOB INDEX NUMBER
9 011174 013700 0000020  MOV      EMTBLK+2,R0      ;GET ADDRESS OF USER'S BUFFER
10 011200 004737 0000000  CALL     VALADW         ;VALIDATE THE ADDRESS
11 011204 123727 0000000 000002  CMPB     EMTBLK,#2      ;GET OR SET PROGRAM NAME?
12 011212 002030      BGE      3$              ;BR IF GET/SET PROGRAM NAME
13 011214 012702 0000006  MOV      #6,R2          ;6 WORDS PER USER NAME
14 011220 070502      MUL      R2,R5              ;* 12 BYTES PER USER NAME
15 011222 062705 0000000  ADD      #LUNAME,R5      ;POINT TO USER NAME TABLE ENTRY
16 011226 105737 0000000  TSTB     EMTBLK         ;IS THIS A REQUEST TO GET OR SET USER NAME
17 011232 001414      BEQ      1$              ;BR IF GETTING THE NAME
18          ;
19          ; This is a request to set the user name.
20          ; SETNAME privilege is required to do this.
21          ;
22 011234 032737 0000000 0000000  BIT      #P0$NAM,PRIVCO  ;IS THIS JOB PRIVILEGED?
23 011242 001004      BNE      2$              ;BR IF YES
24 011244 012700 0000001  MOV      #1,R0          ;RETURN ERROR IF NOT
25 011250 000137 0000000  JMP      SETERR
26 011254 106520      2$:  MFPD     (R0)+          ;GET NEXT WORD OF USER NAME
27 011256 012625      MOV      (SP)+,(R5)+        ;MOVE TO SYSTEM TABLE
28 011260 077203      SOB      R2,2$              ;LOOP TO MOVE ALL OF NAME
29 011262 000432      BR       9$
30          ;
31          ; This is a request to get the user name.
32          ;
33 011264 012546      1$:  MOV      (R5)+,-(SP)      ;STACK A WORD OF THE USER NAME
34 011266 106620      MTPD     (R0)+          ;MOVE TO USER'S BUFFER
35 011270 077203      SOB      R2,1$
36 011272 000426      BR       9$
37          ;
38          ; This is a request to get or set the program name
39          ;
40 011274 003007      3$:  BGT      4$              ;BR IF SETTING
41 011276 016146 0000000  MOV      LPRG1(R1),-(SP)  ;GET FIRST 3 CHARS OF PROG NAME
42 011302 106620      MTPD     (R0)+          ;MOVE TO USER'S BUFFER
43 011304 016146 0000000  MOV      LPRG2(R1),-(SP)  ;GET SECOND 3 CHARS OF PROG NAME
44 011310 106620      MTPD     (R0)+          ;MOVE TO USER'S BUFFER
45 011312 000416      BR       9$
46          ;
47          ; This is a request to set the program name
48          ;
49 011314 106520      4$:  MFPD     (R0)+          ;GET FIRST 3 CHARS
50 011316 012605      MOV      (SP)+,R5          ;OF NEW PGM NAME
51 011320 020527 175000  CMP      R5,#175000      ;IS THIS A VALID RAD50 VALUE
52 011324 103013      BHIS     5$              ;BR IF NOT
53 011326 010561 0000000  MOV      R5,LPRG1(R1)    ;SET FIRST 3 CHARS OF PROG NAME
54 011332 106520      MFPD     (R0)+          ;GET SECOND 3 CHARS
55 011334 012605      MOV      (SP)+,R5          ;OF NEW PGM NAME
56 011336 020527 175000  CMP      R5,#175000      ;IS THIS A VALID RAD50 VALUE
57 011342 103004      BHIS     5$              ;BR IF YES

```

Get or set user name

58	011344	010561	0000000		MOV	R5,LPRG2(R1)	;SET SECOND 3 CHARS OF PROG NAME
59	011350	000137	0000000	9\$:	JMP	EMTXIT	;FINISHED
60							
61	011354	012700	0000002	5\$:	MOV	#2,R0	;SIGNAL ERROR CODE
62	011360	000137	0000000		JMP	SETERR	

Request operator privilege

```

1          .SBTTL Request operator privilege
2          ;-----
3          ; Request operator privilege.
4          ; This is only legal if user has SYSPRV privilege.
5          ;
6 011364 123727 0000000 000001 REQPRV: CMPB   EMTBLK,#1      ;Request to release locked program?
7 011372 001417          BEQ     2$              ;Br if yes
8 011374 032737 0000000 0000000          BIT     #PO$SPV,PRIVCO ;Does job have SETPRV privilege?
9 011402 001410          BEQ     1$              ;Br if not
10 011404 052737 0000000 0000000          BIS     #<PO$SPV!PO$SYS!PO$NAM!PO$BYP>,PRIVCO ;ENABLE PRIVILEGES
11 011412 052761 0000000 0000000          BIS     ##PRGLK,LSW5(R1);Lock program to line
12 011420 000137 0000000          JMP     EMTXIT
13 011424 005000          1$:   CLR     R0              ;RETURN ERROR CODE OF 0
14 011426 000137 0000000          JMP     SETERR
15          ;
16          ; Release program lock
17          ;
18 011432 042761 0000000 0000000 2$:   BIC     ##PRGLK,LSW5(R1);Program no longer locked to line
19 011440 000137 0000000          JMP     EMTXIT

```

Establish Break key sentinel control

```

1          .SBTTL Establish Break key sentinel control
2          ;-----
3          ; Set up control so that pressing the Break key will cause an
4          ; asynchronous completion routine to be entered.
5          ;
6 011444   GETBRK:
7          ;
8          ; See if we are connecting a completion routine to a break character
9          ; or to any input activation condition.
10         ;
11 011444   105737 0000000   TSTB   EMTBLK      ;Connecting to break character?
12 011450   001402         BEQ     3$           ;Br if yes
13 011452   000137 0000000   JMP     STTCPL     ;Go connect compl to activation condition
14         ;
15         ; Set completion routine for break character.
16         ; Cancel any existing break control.
17         ; Return address of existing breakpoint routine to user in R0.
18         ;
19 011456   005061 0000000   3$:   CLR     LBRKCH(R1) ; NO BREAK CHARACTER
20 011462   005037 0000000         CLR     URO      ; ASSUME NO BREAK ROUTINE NOW
21 011466   016100 0000000         MOV     LBRKCQ(R1),R0 ; GET ADDRESS OF BREAK COMPLETION QUEUE ENTRY
22 011472   001410         BEQ     1$           ; BR IF NONE
23 011474   016037 0000000 0000000   MOV     CQ$RTN(R0),URO ; RETURN ADDRESS OF CURRENT BREAKPOINT ROUTINE
24 011502   005061 0000000         CLR     LBRKCQ(R1) ; SAY NO COMPLETION ENTRY
25 011506   010001         MOV     R0,R1        ; GET ADDRESS OF QUEUE ELEMENT TO R1 FOR QFREE
26 011510   004737 0000000         CALL    QFREE     ; RELEASE THE QUEUE ELEMENT
27         ;
28         ; See if we need to establish a completion routine.
29         ;
30 011514   005737 0000040   1$:   TST     EMTBLK+4 ; DID HE SPECIFY THE ADDRESS OF A COMPL RTN?
31 011520   001437         BEQ     2$           ; BR IF NOT
32         ;
33         ; Get a queue element and set up as a completion queue element.
34         ;
35 011522   004737 0000000         CALL    GETQ      ; GET A QUEUE ELEMENT
36 011526   113702 0000000         MOV     CORUSR,R2 ; GET USER INDEX NUMBER
37 011532   110261 0000000         MOV     R2,CQ$JOB(R1) ; SET JOB NUMBER IN QUEUE ELEMENT
38 011536   112761 0000000 0000000   MOV     #S$TWFN,CQ$RNS(R1); SET EXECUTION STATE
39 011544   116261 0000000 0000000   MOV     LPRI(R2),CQ$PRI(R1); SET EXECUTION PRIORITY
40 011552   112761 0000000 0000000   MOV     #CP$STD,CQ$CP(R1); Set compl routine class priority
41 011560   013700 0000040         MOV     EMTBLK+4,R0 ; GET ADDRESS OF COMPL ROUTINE
42 011564   004737 0000000         CALL    UACHKW    ; MAKE SURE ADDRESS IS VALID
43 011570   103415         BCS     9$           ; BR IF INVALID
44 011572   010061 0000000         MOV     R0,CQ$RTN(R1) ; SET COMPL ROUTINE ADDRESS IN QUEUE ELEMENT
45 011576   013761 0000000 0000000   MOV     EMTMAP,CQ$PA5(R1); SET EMT ENTRY MAPPING FOR PAR 5
46 011604   010162 0000000         MOV     R1,LBRKCQ(R2) ; SET ADDRESS OF BREAK QUEUE ELEMENT
47         ;
48         ; Remember special user-defined break character.
49         ;
50 011610   113700 0000020         MOV     EMTBLK+2,R0 ; GET USER-DEFINED BREAK CHARACTER
51 011614   010062 0000000         MOV     R0,LBRKCH(R2) ; SAVE FOR TT INTERRUPT PROCESSING
52         ;
53         ; Finished
54         ;
55 011620   000137 0000000   2$:   JMP     EMTXIT
56         ;
57         ; Error -- Invalid argument address.

```

```
58  
59 011624 004737 0000000 ;  
60 011630 012700 177766 9$: CALL QFREE ;FREE THE QUEUE ELEMENT  
61 011634 000137 0000000 MOV #-12,R0 ;SET ERROR CODE  
JMP SETERR ;ABORT THE EMT
```

```

1          .SBTTL  CHKTT -- CHECK I/O TO TT DEVICE
2          ;-----
3          ;  CHKTT is called to see if the current I/O operation is being directed
4          ;  to device TT or some other device.
5          ;
6          ;  Inputs:
7          ;  CHNADR = Address of current channel block
8          ;
9          ;  Outputs:
10         ;  C-flag cleared if I/O is directed to TT
11         ;  C-flag set if I/O not directed to TT
12         ;
13 011640 017700 0000000  CHKTT:  MOV     @CHNADR,R0      ; GET DEV TABLE INDEX
14 011644 002005          BGE     1$                    ; BRANCH IF NOT OPEN
15 011646 032700 000076          BIT     #76,R0          ; IS THIS TT DEVICE (INDEX # = 0)?
16 011652 001002          BNE     1$                    ; BR IF NOT TT
17 011654 000241          CLC                               ; SAY THIS IS TT
18 011656 000207          RETURN
19 011660 000261          1$:   SEC
20 011662 000207          RETURN

```

CKIOST -- See if scheduler wants to stop I/O

```

1          .SBTTL  CKIOST -- See if scheduler wants to stop I/O
2          ;-----
3          ; If the job scheduler wants to stop all I/O for a job so that
4          ; it can have a chance to swap the job, it sets the LIOHLD flag.
5          ; The CKIOST routine checks to see if this flag is set
6          ; and if so, suspends the job until all I/O is finished.
7          ;
8 011664 010146 CKIOST: MOV      R1, -(SP)
9 011666 032737 0000000 0000000 BIT      #UMODE,EMTPS      ; Was this EMT executed in kernel mode?
10 011674 001434      BEQ      2$              ; Br if yes -- Don't stop system EMT's
11 011676 105737 0000000      TSTB     CURCP        ; Was EMT executed by a completion routine?
12 011702 001031      BNE      2$              ; Br if yes -- I/O hold flag stops cpl rtns
13 011704 113701 0000000      MOVB     CORUSR,R1     ; Get current job index #
14 011710 005761 0000000 1$: TST      LIOHLD(R1)     ; Is I/O hold flag set for job?
15 011714 001424      BEQ      2$              ; Br if not
16 011716      DISABL                     ; *** Disable interrupts ***
17 011724 105761 0000000      TSTB     LIOCNT(R1)     ; Is any I/O in progress now?
18 011730 001407      BEQ      3$              ; Br if not
19 011732 012700 0000000      MOV      #S$IDWT,RO    ; Get I/O wait state code
20 011736 004737 0000000      CALL     QNSPND        ; Enqueue in I/O wait state (enable ints)
21 011742 004737 0000000      CALL     CHKABT        ; See if we were aborted while asleep
22 011746 000760      BR        1$              ; See if we should continue waiting
23 011750 005061 0000000 3$: CLR      LIOHLD(R1)     ; Reset I/O hold flag
24 011754      ENABL                      ; *** Enable interrupts ***
25 011762 004737 0000000      CALL     SCHED        ; Call scheduler to probably swap job
26          ;
27          ; Finished
28          ;
29 011766 012601 2$: MOV      (SP)+, R1
30 011770 000207      RETURN

```

SETQ -- Do setup of I/O queue element

```

1          .SBTTL  SETQ  -- Do setup of I/O queue element
2          ;-----
3          ; SETQ is called to set up parameters in an I/O queue element.
4          ;
5          ; Inputs:
6          ; R1 = Address of queue element
7          ; EMTBLK = EMT arguments
8          ; CHNNUM = User's channel number
9          ; CHNADR = Address of CSW for channel
10         ;
11         ; Outputs:
12         ; RO = Highest block number that will be accessed by I/O
13         ; C-flag set if a fatal error is detected (Error code is returned in INTERR).
14         ; URO = Number of words that will be transferred.
15         ;
16 011772 010246 SETQ:  MOV     R2,-(SP)
17 011774 010346      MOV     R3,-(SP)
18 011776 010446      MOV     R4,-(SP)
19         ;
20         ; Set up misc. parameters
21         ;
22 012000 013761 0000000 0000000 MOV     CHNNUM,Q.CHAN(R1);SET USER'S CHANNEL #
23 012006 013703 0000000      MOV     CHNADR,R3      ;GET ADDRESS OF CSW FOR CHANNEL
24 012012 010361 0000000      MOV     R3,Q.UCSW(R1)   ;SET POINTER TO CSW
25 012016 013761 0000000 0000000 MOV     @#KPAR6,Q.PA6(R1);Save context blk mapping to access channel
26 012024 016302 0000000      MOV     C.CSW(R3),R2   ;GET CHANNEL STATUS WORD
27 012030 042702 177701      BIC     #^C76,R2      ;EXTRACT DEVICE INDEX NUMBER
28 012034 110261 0000000      MOVVB   R2,Q.DEVX(R1)  ;SAVE IN QUEUE ELEMENT CELL
29 012040 116361 0000000 0000000 MOVVB   C.DEVQ(R3),Q.UNIT(R1);SET UNIT NUMBER
30 012046 113702 0000000      MOVVB   CORUSR,R2      ;GET USER INDEX #
31 012052 006202      ASR     R2      ;Convert to job number
32 012054 110261 0000000      MOVVB   R2,Q.JOB(R1)   ;Store job # into queue element
33 012060 020227 000037      CMP     R2,#31.         ;Will number fit in old Q.JNUM field?
34 012064 101402      BLOS    12$      ;Br if yes
35 012066 012702 000037      MOV     #31.,R2         ;Signal that Q.JNUM field not valid
36 012072 072227 000003 12$:  ASH     #3,R2          ;Position for Q.JNUM field
37 012076 150261 0000000      BISB    R2,Q.JNUM(R1)  ;STORE JOB #
38         ;
39         ; Copy original channel block into internal channel block that is
40         ; part of I/O queue element.
41         ;
42 012102 010100      MOV     R1,RO      ;Get address of I/O queue block
43 012104 062700 0000000      ADD     #Q.ICSW,RO      ;Get address of CSW within queue block
44 012110 010061 0000000      MOV     RO,Q.CSW(R1)   ;Set pointer to internal CSW
45 012114 010302      MOV     R3,R2      ;Get pointer to original channel block
46 012116 012220      MOV     (R2)+,(RO)+      ;Copy original channel block to internal one
47 012120 012220      MOV     (R2)+,(RO)+
48 012122 012220      MOV     (R2)+,(RO)+
49 012124 012220      MOV     (R2)+,(RO)+
50 012126 011210      MOV     (R2),(RO)
51         ;
52         ; Set up buffer address
53         ;
54 012130 013702 0000060      MOV     EMTBLK+6,R2    ;GET # WORDS TO BE TRANSFERED
55 012134 001004      BNE     5$      ;BR IF WORD COUNT SPECIFIED
56 012136 123727 0000010 000032 CMPB    EMTBLK+1,#32    ;IS THIS A .SPFUN REQUEST?
57 012144 001104      BNE     1$      ;BR IF NOT -- THIS MUST BE A SEEK REQUEST

```

SETQ -- Do setup of I/O queue element

```

58 012146 013700 0000040 5$: MOV EMTBLK+4,R0 ;BASE ADDRESS OF BUFFER AREA
59 012152 032737 0000000 0000000 BIT #UMODE,EMTPS ;WAS EMT DONE IN USER OR KERNEL MODE?
60 012160 001417 BEQ 2$ ;BR IF IN KERNEL MODE (ASSUME ADDRESS IS OK)
61 012162 004737 0000000 CALL UACHKB ;MAKE SURE IT IS LEGAL
62 012166 103523 BCS 9$ ;BR IF ILLEGAL
63 012170 123727 0000010 000032 CMPB EMTBLK+1,#32 ;IS THIS A .SPFUN OPERATION?
64 012176 001410 BEQ 2$ ;BR IF YES -- DON'T CHECK WORDCOUNT VALUE
65 012200 006302 ASL R2 ;CVT # WORDS TO # BYTES
66 012202 005302 DEC R2 ;GET ADDRESS OF LAST BYTE TRANSFERED
67 012204 060200 ADD R2,R0
68 012206 004737 0000000 CALL UACHKB ;CHECK HIGH RANGE ADDRESS OF BUFFER
69 012212 103511 BCS 9$ ;BR IF ILLEGAL
70 012214 013700 0000040 MOV EMTBLK+4,R0 ;GET BACK BASE ADDRESS OF BUFFER
71 012220 004737 012524' 2$: CALL CVTUAD ;CONVERT USER ADDRESS TO PHYSICAL ADDRESS
72 012224 010061 0000000 MOV R0,Q.BUFF(R1) ;BUFFER ADDRESS RELATIVE TO Q.PAR
73 012230 010261 0000000 MOV R2,Q.PAR(R1) ;RELOCATION BASE ADDRESS
74 ;
75 ; Set up word count and block number for transfer.
76 ;
77 012234 005037 0000000 CLR URO ;SAY # WORDS TRANSFERED = 0 (IN CASE OF ERROR)
78 012240 013704 0000060 MOV EMTBLK+6,R4 ;GET WORD COUNT
79 012244 062704 000377 ADD #377,R4 ;CONVERT TO BLOCK COUNT
80 012250 105004 CLRB R4
81 012252 000304 SWAB R4
82 012254 013700 0000020 MOV EMTBLK+2,R0 ;GET BLOCK # FOR REQUEST
83 012260 011302 MOV (R3),R2 ;GET CSW
84 012262 042702 177701 BIC #^C76,R2 ;GET DEVICE INDEX
85 012266 032762 0000000 0000000 BIT #DS$DIR,DVSTAT(R2); IS THIS A FILE STRUCTURED DEVICE?
86 012274 001007 BNE 10$ ;BR IF YES
87 012276 060400 ADD R4,R0 ;COMPUTE HIGH BLOCK NUMBER FOR TRANSFER
88 012300 032713 0000000 BIT #CS$EOF,(R3) ;HAS END OF FILE BEEN HIT?
89 012304 001424 BEQ 1$ ;BR IF NOT
90 012306 042713 0000000 BIC #CS$EOF,(R3) ;ACKNOWLEDGE EOF
91 012312 000455 BR 7$ ;RETURN EOF STATUS CODE
92 012314 042713 0000000 10$: BIC #CS$EOF,(R3) ;CLEAR END OF FILE FLAG
93 012320 005763 0000000 TST C.SBLK(R3) ;STARTING BLOCK FOR FILE = 0?
94 012324 001414 BEQ 1$ ;DIR OP IF YES
95 012326 016302 0000000 MOV C.LENG(R3),R2 ;GET # BLOCKS ALLOCATED FOR FILE
96 012332 020002 CMP R0,R2 ;IS THIS A VALID STARTING BLOCK #?
97 012334 103044 BHIS 8$ ;BR IF INVALID STARTING BLOCK #
98 012336 060400 ADD R4,R0 ;COMPUTE ENDING BLOCK NUMBER + 1
99 012340 020002 CMP R0,R2 ;WILL TRANSFER PASS END OF FILE?
100 012342 101405 BLOS 1$ ;BR IF WITHIN FILE
101 012344 010200 MOV R2,R0 ;DON'T PASS END OF FILE
102 012346 163702 0000020 SUB EMTBLK+2,R2 ;CALCULATE # BLOCKS ACTUALLY TO BE TRANSFERRED
103 012352 000302 SWAB R2 ;GET ACTUAL WORD COUNT FOR TRANSFER
104 012354 000402 BR 3$
105 012356 013702 0000060 1$: MOV EMTBLK+6,R2 ;GET REQUEST WORD COUNT
106 012362 010237 0000000 3$: MOV R2,URO ;RETURN ACTUAL WORD COUNT TO USER IN R0
107 012366 010261 0000000 MOV R2,Q.WCNT(R1) ;SET WORD COUNT IN Q ELEMENT
108 012372 013702 0000020 MOV EMTBLK+2,R2 ;GET REQUEST BLOCK #
109 012376 066302 0000000 ADD C.SBLK(R3),R2 ;ADD BASE BLOCK # OF FILE
110 012402 010261 0000000 MOV R2,Q.BLKN(R1) ;SET PHYSICAL BLOCK #
111 ;
112 ; See if this is an I/O operation to a logical disk
113 ;
114 012406 126137 0000000 0000000 CMPB Q.DEVX(R1),LDDEVX; I/O TO LOGICAL DISK?

```

SETQ -- Do setup of I/O queue element

```

115 012414 001006      BNE      11$      ;BR IF NOT
116 012416 010046      MOV      RO,-(SP)   ;SAVE HIGH BLOCK NUMBER
117 012420      OCALL    LDIO      ;ALTER QUEUE ELEMENT FOR LOGICAL DISK I/O
118 012426 012600      MOV      (SP)+,RO   ;RESTORE HIGH BLOCK NUMBER
119 012430 103406      BCS      8$      ;BR IF NOT VALID LOGICAL DISK
120 012432 000241      11$:    CLC      ;SIGNAL SUCCESS ON RETURN
121 012434 000407      BR      4$
122      ; Invalid buffer address
123 012436 112737 177766 0000000 9$:    MOVB    #-12,INTERR ;RETURN ERROR CODE 12
124 012444 000402      BR      6$
125      ; Invalid block number (return EOF status)
126 012446      8$:
127      ;      BIS      #CS$EOF,(R3) ;SET END-OF-FILE FLAG IN CSW
128 012446 105037 0000000 7$:    CLRB    INTERR ;RETURN ERROR CODE OF 0
129 012452 000261      6$:    SEC      ;SIGNAL ERROR ON RETURN
130      ;
131      ; Copy Channel Status Word from user's channel status block into
132      ; internal CSW word in I/O queue element (Q. ICSW).
133      ; (Note, Do not alter C-flag)
134      ;
135 012454 016361 0000000 0000000 4$:    MOV      C.CSW(R3),Q.ICSW(R1);Copy CSW into Q element
136      ;
137      ; Finished
138      ;
139 012462 012604      MOV      (SP)+,R4
140 012464 012603      MOV      (SP)+,R3
141 012466 012602      MOV      (SP)+,R2
142 012470 000207      RETURN

```

SETCR -- Set completion routine address in queue element

```

1          .SBTTL SETCR -- Set completion routine address in queue element
2          ;-----
3          ; SETCR is called to store a completion routine address into an
4          ; I/O queue element.
5          ;
6          ; Inputs:
7          ;   R0 = Completion routine address
8          ;   R1 = Address of I/O queue element
9          ;
10         ; Outputs:
11         ;   c-flag set if a fatal error is detected.
12         ;
13 012472 020027 000001 SETCR:  CMP     R0,#1          ; COMPL RTN ADDR OF 1 ==> .READW/.WRITW
14 012476 101403        BLOS    2$              ; BR IF YES
15 012500 004737 0000000 CALL    UACHKW          ; VALIDATE ADDRESS OF COMPLETION ROUTINE
16 012504 103406        BCS     1$              ; BR IF ILLEGAL
17 012506 010061 0000000 2$:   MOV     R0,Q.COMP(R1) ; STORE ADDRESS INTO QUEUE ELEMENT
18 012512 013761 0000000 0000000 MOV     EMTMAP,Q.PA5(R1); SET EMT ENTRY PAR 5 MAPPING VALUE
19 012520 000241        CLC                    ; SIGNAL GOOD RETURN
20 012522 000207        1$:   RETURN

```

CVTUAD -- Convert user address to physical address

```

1          .SBTTL  CVTUAD -- Convert user address to physical address
2          ;-----
3          ; CVTUAD is called to convert a user specified virtual buffer address into
4          ; the corresponding physical address.
5          ; The type of conversion done depends on whether the EMT was executed in
6          ; user or kernel mode:
7          ;
8          ; User mode EMT:
9          ; The address is assumed to be within the job's virtual address region.
10         ;
11         ; Kernel mode EMT:
12         ; 1. If the virtual address is greater than or equal to 140000 it is mapped
13         ; into the job's context block.
14         ; 2. If the virtual address is greater than or equal to 120000 it is mapped
15         ; into the system mapped region using the value of EMTMAP which contains
16         ; the mapping value that kernel PAR 5 had when the EMT was executed.
17         ; 3. If the virtual address = 0 it is mapped to the base of the program
18         ; space. (Used for reading in SAV files).
19         ; 4. Otherwise the address is assumed to already be physical and is returned
20         ; unchanged.
21         ;
22         ; If the address needs to be mapped, the virtual address returned will be
23         ; in the range 140000-157777 so that it will map through page 6.
24         ;
25         ; Inputs:
26         ; R0 = Virtual address to be mapped.
27         ;
28         ; Outputs:
29         ; R0 = 16-bit offset within page.
30         ; R2 = PAR page base.
31         ;
32 012524 010146 CVTUAD: MOV      R1, -(SP)
33 012526 010346      MOV      R3, -(SP)
34 012530 113701 000000G      MOVB   CORUSR, R1      ; GET JOB INDEX NUMBER
35
36         ; Determine if the EMT was done in user or kernel mode.
37         ;
38 012534 032737 000000G 000000G      BIT      #UMODE, EMTPS      ; WAS EMT DONE IN USER OR KERNEL MODE?
39 012542 001443      BEQ      1$      ; BR IF IN KERNEL MODE
40
41         ; EMT was done in user mode.
42         ; Map user virtual address to physical address.
43         ;
44 012544 032761 000000G 000000G      BIT      ##INKMN, LSW4(R1); IS KMON RUNNING?
45 012552 001405      BEQ      8$      ; BR IF NOT
46 012554 020027 000000G      CMP      R0, #CXTBAS      ; IS KMON DOING I/O TO JOB CONTEXT AREA?
47 012560 103037      BHIS     5$      ; BR IF YES
48 012562 162700 000000G      SUB      #KMNBAS, R0      ; SUBTRACT KMON ADDRESS BIAS
49 012566 010003      B$:      MOV      R0, R3      ; GET VIRTUAL ADDRESS
50 012570 005002      CLR      R2      ; CLEAR FOR SHIFT
51 012572 073227 000003      ASHC     #3, R2      ; GET VIRTUAL PAR # FROM ADDRESS
52 012576 006302      ASL      R2      ; *2 TO GET WORD TABLE INDEX
53 012600 005762 000000G      TST      RPDR(R2)      ; IS THIS PAGE SPECIALLY MAPPED?
54 012604 001405      BEQ      6$      ; BR IF NOT
55 012606 016202 000000G      MOV      RPAR(R2), R2      ; GET BASE 64-BYTE BLOCK # FOR THIS PAGE
56 012612 042700 160000      BIC      #160000, R0      ; CLEAR PAR # FROM VIRTUAL ADDRESS
57 012616 000402      BR       7$

```

```

58 012620 016102 0000000 6$: MOV LPARBS(R1),R2 ;GET BASE 64-BYTE BLOCK # FOR JOB
59 012624 010003 7$: MOV RO,R3 ;GET VIRTUAL ADDRESS
60 012626 000241 CLC ;SHIFT RIGHT 6-BITS WITHOUT SIGN EXTENSION
61 012630 006003 ROR R3
62 012632 072327 177773 ASH #-5,R3
63 012636 060302 ADD R3,R2 ;GET 64-BYTE BLOCK # OF PHYSICAL ADDRESS
64 012640 042700 177700 BIC #^C77,R0 ;GET BYTE-WITHIN-BLOCK FROM VIRTUAL ADDRESS
65 012644 062700 0000000 ADD #VPAR6,R0 ;MAP INTO PAR6 REGION
66 012650 000440 BR 9$
67 ;
68 ; EMT was done in kernel mode.
69 ; Determine which region to map to.
70 ;
71 012652 020027 0000000 1$: CMP RO,#XTBAS ;IS ADDRESS IN JOB CONTEXT BLOCK?
72 012656 103405 BLO 2$ ;BR IF NOT
73 ;
74 ; Map to Job context block.
75 ;
76 012660 016102 0000000 5$: MOV LBASE(R1),R2 ;GET BASE 256-WORD BLOCK # OF CONTEXT BLOCK
77 012664 072227 0000003 ASH #3,R2 ;CONVERT TO 32-WORD BLOCK #
78 ; Note: Virtual address in RO is already in PAR 6 range (140000-157777).
79 012670 000430 BR 9$
80 ;
81 ; See if accessing the system mapped region.
82 ;
83 012672 020027 0000000 2$: CMP RO,#VPAR5 ;IS ADDRESS IN MAPPED SYSTEM REGION?
84 012676 103405 BLO 12$ ;BR IF NOT
85 ;
86 ; Map to the buffer using the current contents of EMTMAP which indicates
87 ; the mapped value for kernel PAR 5 when the EMT was done.
88 ;
89 012700 013702 0000000 MOV EMTMAP,R2 ;Get mapping value that PAR 5 had on entry
90 012704 062700 020000 ADD #20000,R0 ;INCREASE VIRTUAL ADDRESS
91 012710 000420 BR 9$
92 ;
93 ; Not in job context area.
94 ; See if accessing base of job region.
95 ;
96 012712 005700 12$: TST RO ;VIRTUAL ADDRESS = 0?
97 012714 001005 BNE 3$ ;BR IF NOT
98 ;
99 ; Map to base of job space.
100 ;
101 012716 016102 0000000 MOV LPARBS(R1),R2 ;GET PAR OF JOB BASE
102 012722 062700 0000000 ADD #VPAR6,R0 ;PUT VIRTUAL ADDRESS IN PAGE 6 REGION
103 012726 000411 BR 9$
104 ;
105 ; Address is already physical.
106 ;
107 012730 010002 3$: MOV RO,R2 ;GET ACTUAL ADDRESS
108 012732 000241 CLC ;CONVERT TO PAGE #
109 012734 006002 ROR R2
110 012736 072227 177773 ASH #-5,R2
111 012742 042700 177700 BIC #^C77,R0 ;GET BYTE-IN-PAGE TO RO
112 012746 062700 0000000 ADD #VPAR6,R0 ;MAP THROUGH PAR 6
113 ;
114 ; Finished

```

```
115  
116 012752 012603      ;  
117 012754 012601      9$:  MOV      (SP)+, R3  
118 012756 000207      MOV      (SP)+, R1  
                        RETURN
```

LDIO -- Process I/O to logical disks

```

1          .SBTTL LDIO  -- Process I/O to logical disks
2          ;-----
3          ; LDIO is called from the SETQ routine in TSEMT when it determines
4          ; that the queue element being set up is for an I/O operation to
5          ; a logical disk.
6          ; This routine converts the logical device #, unit #, and block #
7          ; into a physical device #, unit #, and block #.
8          ;
9          ; Inputs:
10         ; R1 = Address of I/O queue element
11         ; EMTBLK = Current EMT argument block
12         ;
13 012760 010446 LDIO:  MOV     R4, -(SP)
14         ;
15         ; Change logical device number and unit number into physical
16         ; device number and unit number.
17         ;
18 012762 116104 0000000 MOVB    Q.UNIT(R1),R4  ;GET LOGICAL DISK UNIT NUMBER
19 012766 042704 177770 BIC     #^C7,R4      ;GET UNIT # ONLY
20 012772 006304 ASL      R4              ;CONVERT TO WORD TABLE INDEX
21 012774 016400 0000000 MOV     LDPDEV(R4),R0  ;GET PHYSICAL UNIT # AND DEVICE #
22 013000 001456 BEQ      2$              ;BR IF THIS LOGICAL UNIT NOT IN USE
23 013002 110061 0000000 MOVB    R0,Q.DEVX(R1) ;SET PHYSICAL DEVICE INDEX NUMBER
24 013006 142761 000007 0000000 BICB   #7,Q.UNIT(R1) ;CLEAR OLD UNIT #
25 013014 000300 SWAB     R0              ;GET PHYSICAL UNIT # TO LOW BYTE
26 013016 150061 0000000 BISB    R0,Q.UNIT(R1) ;SET PHYSICAL UNIT #
27         ;
28         ; See if this is a .SPFUN to get device size or LD unit tables
29         ;
30 013022 116100 0000000 MOVB    Q.FUNC(R1),R0  ;IS THIS A .SPFUN REQUEST?
31 013026 002023 BGE      1$              ;BR IF NOT
32 013030 105061 0000000 CLRB    Q.DEVX(R1)   ;TELL QIO NOT TO DO ANY ACTUAL I/O
33 013034 120027 000372 CMPB    R0,#372      ;IS THIS FUNCTION TO GET DEVICE TABLE?
34 013040 001006 BNE      5$              ;BR IF NOT
35 013042 005737 0000060 TST     EMTBLK+6      ;IS THIS A REQUEST TO UPDATE HANDLER TABLE?
36         ; RT V5.4 changed logic to: -wcnt as write, +wcnt as read
37         ; Remove the next line to always read (-wcnt and +wcnt ==> read)
38         ;
39 013046 000240 BPL      4$              ;IGNORE IT IF YES
40 013050 004737 013144' NOP                ;Reserve room to patch BR back in if needed
41 013054 000426 CALL     LDRST      ;PASS DEVICE TABLE INFO TO USER
42 013056 120027 000373 5$: CMPB    R0,#373      ;IS THIS FUNCTION TO GET DEVICE SIZE?
43 013062 001023 BNE      4$              ;BR IF NOT -- IGNORE ALL OTHER SPECIAL FUNCTS
44 013064 016446 0000000 MOV     LDSIZE(R4),-(SP) ;GET LOGICAL DISK SIZE
45 013070 106677 0000040 MTPD    @EMTBLK+4      ;MOVE INTO USER'S BUFFER
46 013074 000416 BR       4$              ;FINISHED
47         ;
48         ; Check to make sure that the transfer is completely within the
49         ; area of the logical disk.
50         ;
51 013076 016100 0000000 1$: MOV     Q.WCNT(R1),R0  ;GET WORD COUNT FOR TRANSFER
52 013102 062700 000377 ADD      #377,R0      ;CONVERT TO BLOCK COUNT
53 013106 105000 CLRB     R0
54 013110 000300 SWAB     R0
55 013112 066100 0000000 ADD     Q.BLKN(R1),R0  ;ADD TRANSFER STARTING BLOCK NUMBER
56 013116 020064 0000000 CMP      R0,LDSIZE(R4) ;IS TRANSFER WITHIN LOGICAL DISK?
57 013122 101000 BHI      2$              ;BR IF NOT

```

LDIO -- Process I/O to logical disks

```
58 ;
59 ; Bias logical block number by base block of logical disk file
60 ;
61 013124 066461 0000000 0000000 ADD LDBASE(R4),Q.BLKN(R1) ;BIAS BLOCK NUMBER
62 ;
63 ; Transfer is ready to go
64 ;
65 013132 000241 4$: CLC ; SIGNAL SUCCESS ON RETURN
66 013134 000401 BR 3$
67 ;
68 ; End of file hit by transfer
69 ;
70 013136 000261 2$: SEC ; SIGNAL ERROR ON RETURN
71 ;
72 ; Finished
73 ;
74 013140 012604 3$: MOV (SP)+,R4
75 013142 000207 RETURN
```

LDIO -- Process I/O to logical disks

```

1      ; -----
2      ; Process the Logical Disk special function code 372 that is used to
3      ; pass the LD data tables to the user.
4      ; The information returned consists of 4 tables with one entry in each
5      ; table for each logical disk unit (0-7).
6      ; The four tables are:
7      ;   FLAGS (1 word per unit)
8      ;       100000 = Unit is allocated
9      ;       020000 = Unit is read-only
10     ;       003400 = Unit number mask
11     ;       000076 = Device index number mask
12     ;   OFFSET (1 word per unit)
13     ;       Starting block number of logical disk on real disk.
14     ;   SIZE (1 word per unit)
15     ;       Number of blocks allocated for logical disk.
16     ;   NAME (4 words per unit)
17     ;       File spec for logical disk file.
18     ;
19 013144 010246      LDRST:  MOV     R2,-(SP)
20 013146 010346      MOV     R3,-(SP)
21 013150 010446      MOV     R4,-(SP)
22     ;
23     ; Pass flag table info to user
24     ;
25 013152 013702 0000040      MOV     EMTBLK+4,R2      ;Get address of user's buffer
26 013156 123727 0000000 0000001      CMPB    LDVERS,#1      ;LD version format 5.3& earlier?
27 013164 001406      BEQ     11$      ;Skip first two words if old format
28 013166 012746 045640      MOV     #^RLD,-(SP)      ;Pass LD device name to user
29 013172 106622      MTPD    (R2)+
30 013174 012746 0000000      MOV     #MAXLD,-(SP)      ;Pass number of LD units to user
31 013200 106622      MTPD    (R2)+
32 013202 005003      11$:  CLR     R3      ;Init LD index
33 013204 016300 0000000      1$:  MOV     LDPDEV(R3),R0      ;Get real disk device # and unit #
34 013210 001417      BEQ     2$      ;Br if unit is not allocated
35 013212 052700 100000      BIS     #100000,R0      ;Set unit-allocated flag in status
36 013216 032763 0000000 0000000      BIT     #LD$RON,LD$FLAG(R3);Is the unit read-only?
37 013224 001411      BEQ     2$      ;Br if not read-only
38 013226 123727 0000000 0000001      CMPB    LDVERS,#1      ;LD version format 5.3& earlier?
39 013234 001403      BEQ     22$      ;Use old mask for old format
40 013236 052700 040000      BIS     #040000,R0      ;Set read-only status flag (5.4&later)
41 013242 000402      BR      2$
42 013244 052700 020000      22$:  BIS     #020000,R0      ;Set read-only status flag (5.3&earlier)
43 013250 010046      2$:  MOV     R0,-(SP)      ;Pass status flags to user
44 013252 106622      MTPD    (R2)+
45 013254 062703 0000002      ADD     #2,R3      ;Increment unit index
46 013260 020327 0000000      CMP     R3,#MAXLD*2      ;Done all units?
47 013264 103747      BLD     1$      ;Br if more to do
48     ;
49     ; Offset info table
50     ;
51 013266 012704 0000000      MOV     #LDBASE,R4      ;Point to table with base values for disks
52 013272 012700 0000000      MOV     #MAXLD,R0      ;Get number of units
53 013276 012446      3$:  MOV     (R4)+,-(SP)      ;Pass offset info to user
54 013300 106622      MTPD    (R2)+
55 013302 077003      SOB     R0,3$
56     ;
57     ; Size info table

```

```
58 ;
59 013304 012704 00000000      MOV      #LDSIZE,R4      ;Point to table with size info
60 013310 012700 00000000      MOV      #MAXLD,R0      ;Get # LD units
61 013314 012446      4$:      MOV      (R4)+,-(SP)      ;Pass size info to user
62 013316 106622      MTPD      (R2)+
63 013320 077003      SOB       R0,4$
64 ;
65 ; File name table
66 ;
67 013322 012704 00000000      MOV      #LDNAME,R4      ;Point to table with file name info
68 013326 012700 00000000      MOV      #MAXLD*4,R0      ;Get # words to move
69 013332 012446      5$:      MOV      (R4)+,-(SP)      ;Pass info to user
70 013334 106622      MTPD      (R2)+
71 013336 077003      SOB       R0,5$
72 ;
73 ; Finished
74 ;
75 013340 012604      MOV      (SP)+,R4
76 013342 012603      MOV      (SP)+,R3
77 013344 012602      MOV      (SP)+,R2
78 013346 000207      RETURN
```

INDIO -- Perform I/O for IND data segment

```

1          .SBTTL  INDIO  -- Perform I/O for IND data segment
2          ;-----
3          ; INDIO is called for each Read and Write operation when IND is running.
4          ; It determines if the read/write is directed to the IND data segment
5          ; overlay and if so, redirects it to the appropriate position in the
6          ; INDTMP.TSX file.
7          ; This is done by altering the block number in EMTBLK and changing
8          ; CHNADR to point to a channel block that is opened to the INDTMP file.
9          ;
10         ; Inputs:
11         ; R1 = Job index number
12         ; CHNNUM = Channel number
13         ; CHNADR = Address of CSW for channel
14         ;
15         ; Outputs:
16         ; If I/O is redirected to INDTMP file...
17         ; R3 = Address of channel block for INDTMP file.
18         ; CHNADR = Address of channel block for INDTMP file.
19         ; EMTBLK+2 = Block # in INDTMP file of area for data segment for this job.
20         ;
21 013350 010544  INDIO:  MOV      R5, -(SP)
22         ;
23         ; Determine if this I/O is directed to the IND data segment overlay
24         ;
25 013352 023727 0000000 000017      CMP      CHNNUM, #17      ; IS I/O BEING DONE TO IND. SAV FILE?
26 013360 001034      BNE      9$      ; BR IF NOT
27 013362 023737 0000020 0000000      CMP      EMTBLK+2, INDDBL ; ARE WE ACCESSING DATA SEGMENT OVERLAY?
28 013370 001030      BNE      9$      ; BR IF NOT
29         ;
30         ; IND is accessing its data segment overlay.
31         ; Determine if the data segment has been written to the temp file yet.
32         ;
33 013372 032761 0000000 0000000      BIT      ##INDDW, LSW6(R1); HAS DATA SEGMENT BEEN WRITTEN TO TEMP FILE?
34 013400 001007      BNE      1$      ; BR IF YES
35 013402 123727 0000010 000010      CMPB     EMTBLK+1, #10      ; IS THIS A READ OR WRITE REQUEST?
36 013410 001420      BEQ      9$      ; BR IF READ -- GO TO SAV FILE TILL 1ST WRITE
37 013412 052761 0000000 0000000      BIS      ##INDDW, LSW6(R1); REMEMBER 1ST WRITE HAS BEEN DONE TO TEMP FILE
38         ;
39         ; Change info to cause I/O to be directed to SY:INDTMP.TSX file.
40         ; The position within the INDTMP file is a function of the Job index #.
41         ;
42 013420 010105 1$:      MOV      R1, R5      ; GET JOB #
43 013422 006205      ASR      R5      ; DIVIDE BY 2
44 013424 005305      DEC      R5      ; MAKE RELATIVE TO 0
45 013426 070537 0000000      MUL      INDDBS, R5      ; TIMES # OF BLOCKS IN THE OVERLAY SEGMENT
46 013432 010537 0000020      MOV      R5, EMTBLK+2      ; REPLACE BLOCK # FOR I/O OPERATION
47 013436 013703 0000000      MOV      CXTRMN, R3      ; GET ADDRESS OF JOB'S SIMULATED RMON
48 013442 062703 0000000      ADD      #R$INTC, R3      ; POINT TO CHANNEL BLOCK FOR INDTMP FILE
49 013446 010337 0000000      MOV      R3, CHNADR      ; DO I/O TO INDTMP FILE (SET NEW CSW ADDRESS)
50         ;
51         ; Finished
52         ;
53 013452 012605 9$:      MOV      (SP)+, R5
54 013454 000207      RETURN

```

```

1          .SBTTL  EMTTRC -- Trace EMT execution
2          ;-----
3          ; EMTTRC is called from TSEMT when each EMT is processed and the
4          ; EMT TRACE mode is turned on.  The EMT information has been
5          ; set up in EMTBLK.
6          ;
7 013456 010246 EMTTRC: MOV      R2,-(SP)
8 013460 010346      MOV      R3,-(SP)
9 013462 010446      MOV      R4,-(SP)
10 013464 010546      MOV      R5,-(SP)
11          ;
12          ; Determine if we should trace this EMT
13          ;
14 013466 032761 000000G 000000G      BIT      ##INKMN,LSW4(R1); IS KMON RUNNING?
15 013474 001101      BNE      9$          ;BR IF YES -- DON'T TRACE KMON
16 013476 032761 000000G 000000G      BIT      ##CCLRN,LSW5(R1); IS CCL RUNNING?
17 013504 001075      BNE      9$          ;BR IF YES -- DON'T TRACE CCL
18 013506 032737 000000G 000000G      BIT      #UMODE,EMTPS      ;WAS EMT EXECUTED IN USER MODE?
19 013514 001471      BEQ      9$          ;BR IF NOT -- DON'T TRACE KERNEL EMT'S
20          ;
21          ; We want to trace this EMT
22          ; Begin display line
23          ;
24 013516 012700 000015      MOV      #15,R0          ;PRINT CR
25 013522          .TTYOUT
26 013526 012700 000012      MOV      #12,R0          ;PRINT LF
27 013532          .TTYOUT
28          ; Print EMT address
29 013536 013705 000000G      MOV      EMTADR,R5      ;GET EMT ADDRESS
30 013542 012703 000005      MOV      #5,R3          ;PRINT 16-BIT VALUE
31 013546 004737 013712'      CALL      PRTOCT
32          ; Print EMT instruction low-order byte
33 013552 113705 000000G      MOVB     CUREMT,R5      ;GET INSTRUCTION ARGUMENT BYTE
34 013556 012703 000003      MOV      #3,R3          ;PRINT 8-BITS
35 013562 004737 013712'      CALL      PRTOCT      ;PRINT IT
36          ; Print function code
37 013566 113705 000001G      MOVB     EMTBLK+1,R5     ;GET FUNCTION CODE
38 013572 123727 000000G 000374      CMPB     CUREMT,#374 ;IS THIS AN EMT 374?
39 013600 001003      BNE      3$          ;BR IF NOT
40 013602 162705 000044      SUB      #E375MX,R5      ;GET 374 FUNCTION CODE
41 013606 000405      BR      2$          ;GO PRINT IT
42 013610 020527 000060      3$:      CMP      R5,#RT11EX ;IS THIS A TSX EMT?
43 013614 103402      BLD      2$          ;BR IF NOT
44 013616 062705 000020      ADD      #<100-RT11EX>,R5 ;GET ORIGINAL FUNCTION CODE VALUE
45 013622 004737 013712'      2$:      CALL      PRTOCT
46          ; Print channel number
47 013626 113705 000000G      MOVB     EMTBLK,R5      ;GET CHANNEL #
48 013632 004737 013712'      CALL      PRTOCT      ;PRINT IT
49          ; Print EMT argument values
50 013636 012704 000002G      MOV      #EMTBLK+2,R4    ;POINT TO ARGUMENT BLOCK
51 013642 012702 000005      MOV      #5,R2          ;PRINT 5 ARGUMENT VALUES
52 013646 010203      MOV      R2,R3          ;PRINT 16-BITS EACH
53 013650 012405      1$:      MOV      (R4)+,R5      ;GET ARGUMENT VALUE
54 013652 004737 013712'      CALL      PRTOCT      ;PRINT IT
55 013656 077204      SOB      R2,1$          ;LOOP IF MORE TO PRINT
56          ; End of line
57 013660 012700 000015      MOV      #15,R0          ;PRINT CR

```

```
58 013664      . TTYOUT
59 013670 012700 000012      MOV      #12,R0      ; PRINT LF
60 013674      . TTYOUT
61      ;
62      ; Finished
63      ;
64 013700 012605 9$:      MOV      (SP)+,R5
65 013702 012604      MOV      (SP)+,R4
66 013704 012603      MOV      (SP)+,R3
67 013706 012602      MOV      (SP)+,R2
68 013710 000207      RETURN
```

```

1          .SBTTL  PRTOCT -- Print an octal value
2          ;-----
3          ; PRTOCT is called to print an octal value on the user's terminal.
4          ;
5          ; Inputs:
6          ;   R5 = Value to be printed.
7          ;   R3 = 8/16-bit flag:  3==>8-bit value,  5==>16-bit value
8          ;
9 013712    010346  PRTOCT: MOV      R3,-(SP)
10 013714    010446      MOV      R4,-(SP)
11 013716    010546      MOV      R5,-(SP)
12          ;
13          ; If this is a 16-bit value, print 1st digit now
14          ;
15 013720    020327    000005      CMP      R3,#5          ; IS THIS A 16-BIT VALUE?
16 013724    001007      BNE      1$          ; BR IF NOT
17 013726    012700    000030      MOV      #30,R0         ; GET 60 SHIFTED
18 013732    006105      ROL      R5          ; GET 1ST BIT OF VALUE
19 013734    006100      ROL      R0          ; INTO R0
20 013736      .TTYOUT          ; PRINT IT
21 013742    000404      BR       2$
22          ;
23          ; This is an 8-bit value, left justify the 8 bit quantity
24          ;
25 013744    042705    177400  1$:      BIC      #^C<377>,R5      ; CLEAR ALL BUT 8 BITS
26 013750    072527    000007      ASH      #7,R5          ; LEFT JUSTIFY THE VALUE
27          ;
28          ; Begin loop to print the value
29          ;
30 013754    012704    000006  2$:      MOV      #6,R4          ; WILL BECOME 60 WHEN SHIFTED
31 013760    073427    000003      ASHC     #3,R4          ; MOVE 3 BITS FROM R5 TO R4
32 013764    010400      MOV      R4,R0         ; GET TO R0 FOR TTYOUT
33 013766      .TTYOUT          ; SEND THE CHARACTER
34 013772    077310      SOB      R3,2$         ; LOOP IF MORE DIGITS TO PRINT
35          ;
36          ; Put in 2 spaces following value
37          ;
38 013774    112700    000040      MOVB     #' ,R0         ; GET SPACE
39 014000      .TTYOUT          ; SEND IT
40 014004      .TTYOUT          ; " "
41          ;
42          ; Finished
43          ;
44 014010    012605      MOV      (SP)+,R5
45 014012    012604      MOV      (SP)+,R4
46 014014    012603      MOV      (SP)+,R3
47 014016    000207      RETURN
48          .END
  
```

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 9466 Words (37 Pages)
 Size of core pool: 17920 Words (70 Pages)

TSEM2 TSX-Plus EMT Overlay MACRO V05.04 Friday 18-Dec-87 14:02 Page 58-1
PRTOCT -- Print an octal value

Operating system: RT-11

Elapsed time: 00:01:24.40

DK: TSEM2, LP: TSEM2=DK: TSEM2, MAC/C/N: SYM

\$CCLRN	1-102	57-16											
\$CFOPN	1-83	14-21											
\$CTRLD	1-106	1-125	14-23										
\$CTRLS	1-84	43-67											
\$DBGMD	1-119	40-55	40-58										
\$DEAD	1-126	43-26											
\$DETCH	1-85	31-34											
\$DILUP	1-150	31-32											
\$EMTTR	1-91	4-14											
\$FORM	1-118	1-161											
\$HARD	1-126	43-24											
\$INDDW	1-121	56-33	56-37										
\$INDRN	1-109	8-75	9-91										
\$INKMN	1-103	4-36	31-38	53-44	57-14								
\$IOMAP	1-162	26-84											
\$KINIT	1-165	37-34											
\$MLOCK	1-165	37-67											
\$NOINT	1-132	41-9	41-14										
\$ODTMD	1-130	40-54	40-57										
\$PRGLK	1-103	47-11	47-18										
\$TTGAG	1-136	31-36											
\$V52EM	1-158	39-11											
\$VIRJB	1-96	12-90											
\$VNOTT	1-84	45-78											
... V1	15-40	57-25	57-27	57-58	57-60	58-20	58-33	58-39	58-40				
... V2	15-40	15-40#											
ABORT	1-129												
ABRTAD	1-101	36-20*											
ABRTCD	1-101	36-21*											
ABTIO	1-87	2-58											
ADM3A	1-159	39-30											
ALCEMT	1-89	2-110											
ASTXIT	3-18	34-5#											
BADEMT	1-75	2-19	2-23	2-29	2-30	2-39	2-56	4-25	34-12	36-16			
C. CSW	1-142	8-7	8-16	8-48	8-53	8-67	8-69*	8-148	8-150*	9-7	9-16	9-51	
	9-63	9-65*	9-71	9-73*	9-128	9-130*	9-136	9-138*	11-7	11-74	11-76*	11-79	
	11-81*	16-6	16-9	16-36	16-38*	17-9	17-26	17-51*	18-6	18-20	18-27*	19-9	
	19-20	19-22*	25-9	25-22	25-38	51-26	51-135						
C. DEVQ	1-144	17-32	25-35	51-29									
C. LENG	1-145	25-28	51-95										
C. NUMQ	1-143	17-31*											
C. SBLK	1-145	8-83	8-108	25-25	51-93	51-109							
C. USED	1-142	8-104	8-112*	8-113*	25-31								
CANMKT	1-73	15-21	23-56	24-9#									
CAOPN	2-85	5-12#											
CDFN	2-21	25-52#											
CFLAG	1-109	8-121											
CHKABT	1-133	21-14	32-24	50-21									
CHKERR	7-9	7-18	7-27#										
CHKSF	5-6	5-12	5-19	5-26	5-33	5-39	5-45	5-53	5-61#				
CHKTT	8-42	9-43	49-13#										
CHKUSP	1-90												
CHNADR	1-124	4-51*	8-6	9-6	11-6	16-5	17-5	17-25	18-5	19-5	25-5	49-13	
	51-23	56-49*											
CHNERR	4-40	4-70#											
CHNNUM	1-120	4-42*	18-9	51-22	56-25								

[illegible]

CSFACT	2-81	5-45#											
CSIARE	1-146	1-172											
CSIGEN	1-90	3-8											
CSISPC	1-90	3-9											
CUNLK	2-65	5-19#											
CURCP	1-125	35-7	50-11										
CUREMT	1-106	57-33	57-38										
CVTUAD	51-71	53-32#											
CW\$FPU	1-130	12-29											
CXTBAS	1-112	53-46	53-71										
CXTBUF	1-84	37-79											
CXTRMN	1-126	4-43	20-19	26-25	27-8	27-22	27-39	56-47					
DATE	2-57	20-17#											
DCRD1	1-110	9-84											
DCRD2	1-111	9-118											
DELETE	1-86	2-8											
DETEMT	1-77	2-90											
DH\$LB	1-106	8-87											
DIABLO	1-160	39-33											
DIEARG	1-41												
DIEMSG	1-41												
DISMNT	1-88	2-93											
DOCIN	2-55	14-5#											
DOCOPN	1-91	5-7											
DOCULK	1-91	5-20											
DOEMT	1-69	4-13#											
DOQVAL	26-17#	27-55											
DOOPAP	1-91	5-13											
DORLK	1-91	5-27											
DOSFCK	1-92	5-46											
DOTLK	1-92	5-34											
DOULK1	1-92	5-40											
DOWAIT	2-18	2-47	19-5#										
DS\$DIR	1-109	8-85	8-110	11-30	51-85								
DS\$NRD	1-141	18-22											
DS\$RON	1-109	8-59											
DS\$SFN	1-96	11-16											
DS\$WON	1-110	9-55											
DSTAT	1-89	3-6											
DVFLAG	1-108	8-18	9-18										
DVSTAT	1-134	8-59	8-85	8-110	9-55	11-16	11-30	18-22	51-85				
DX\$EBA	1-108	8-18	9-18										
E374MX	2-59#												
E375MX	1-68	2-44#	57-40										
EMT16	2-22	4-63#											
EMT374	2-47#	2-59											
EMT376	1-68	36-8#											
EMTADR	1-104	36-10	57-29										
EMTASP	1-90	1-90											
EMTBLK	1-102	4-22	4-32	4-63	7-7	7-16	7-25	8-14	8-25	8-35	8-87	8-119	
	8-125	8-142	8-168	9-14	9-26	9-36	9-86	9-101	9-122	11-22	11-26*	11-46	
	11-55	11-65	12-6	12-15	12-24	12-42	12-72	13-5	16-27	17-16	20-5	20-34	
	21-5	22-16	22-24	22-31	23-6	23-31	25-16	25-52	26-5	26-11	26-12	29-17	
	29-23	31-22*	31-26	31-40	31-54	31-70	32-12	32-16	35-15	37-15	37-24	37-41	
	39-44	40-8	40-20	40-52	41-4	42-9	42-22	43-15	43-35	43-45	43-53	43-61	
	45-14	45-23	45-24	46-9	46-11	46-16	47-6	48-11	48-30	48-41	48-50	51-54	

INTERR	1-110	9-104*	9-151	51-123*	51-128*				
INTPRI	1-99	22-38	23-27	23-50	24-20	24-28	32-31	33-25	50-24
IDABFL	1-127	18-24							
IDABRT	8-160	9-100	9-105	9-150#	11-91				
IDHALT	1-104	15-6							
IDWAIT	1-73	8-144	9-124	11-70	16-16	18-8	19-16		
JBINFO	2-100	37-15#							
JCDB	1-126	8-136	9-80	9-114	16-20	17-40	18-13		
JIBASE	37-116#	37-163							
JICONT	37-110#	37-162							
JICPU	37-133#	37-166							
JIDLN	1-164	37-65							
JIJMPX	37-53	37-159#	37-169						
JIMEM	37-102#	37-161							
JIMLOK	1-164	37-69							
JINAME	37-148#	37-168							
JIPPN	37-127#	37-165							
JIPRIO	37-143#	37-167							
JIPRIV	1-164	37-81							
JIPROG	37-121#	37-164							
JIRET1	37-84	37-96	37-98	37-104#	37-112	37-117	37-144		
JIRET2	37-123	37-129	37-135#						
JIRNND	37-93	38-31#							
JIRNST	37-88	38-5#							
JIRUN	37-88#	37-160							
JISTAT	37-57#	37-159							
JIVLN	1-164	37-63							
JNOP	12-9	12-18#	12-34						
JSWLOC	1-135	12-87*							
KMNBAS	1-112	53-48							
KMNEMT	1-76	2-86							
KPAR6	1-85	51-25							
LA120	1-159	39-32							
LA36	1-159	39-31							
LACTIV	1-99	40-65							
LBASE	1-147	37-116	53-76						
LBRKCH	1-127	48-19*	48-51*						
LBRKCQ	1-127	48-21	48-24*	48-46*					
LBSPRI	1-128	45-94	45-96*						
LCONTM	1-151	37-111	45-110						
LCPUHI	1-166	37-133							
LCPULO	1-166	37-134							
LD*RON	1-97	55-36							
LDBASE	1-97	54-61	55-51						
LDDEVX	1-112	11-34	51-114						
LDFLAG	1-97	55-36							
LDIO	51-117	54-13#							
LDMEMT	1-84	2-115							
LDNAME	1-104	55-67							
LDPDEV	1-97	54-21	55-33						
LDRST	54-40	55-19#							
LDSize	1-97	54-44	54-56	55-59					
LDVERS	1-97	55-26	55-38						
LEMTPC	1-158	28-5*							
LIOCNT	1-131	50-17							
LIOHLD	1-131	50-14	50-23*						

LITIME	1-132	41-15*							
LJSW	1-120	12-87*	12-88	15-38	28-9				
LMSGBF	1-138	33-15	33-23*						
LNBLKS	1-165	37-102							
LNMAP	1-138	33-29							
LNPRIM	1-138	26-131	26-160	33-14	39-10	40-19	43-21		
LNSBLK	1-163	37-103							
LOOKUP	1-86	2-7							
LOTRM	39-8	39-20	39-27#						
LPARBS	1-114	53-58	53-101						
LPARNT	1-105	26-140							
LPRG1	1-166	37-121	46-41	46-53*					
LPRG2	1-166	37-122	46-43	46-58*					
LPRI	1-128	26-106	37-143	45-97*	48-39				
LPROG	1-104	26-96	37-128	39-48	45-118				
LPROJ	1-104	26-91	37-127	39-46	45-117				
LSCCA	1-130	12-5	12-8*						
LSECPT	1-138	26-161	40-28						
LSLEPH	1-122	21-8*	21-15						
LSLEPL	1-122	21-10*	21-18						
LSPND	1-123	15-17*							
LST16	3-4#	4-65							
LSTATE	1-139	37-89							
LSTD1	1-156	26-158	37-61	40-15					
LSTHL	1-115	1-120	43-22						
LSTPL	1-150	37-59	40-13						
LSTSL	1-165	31-30	37-27	43-19					
LSW	1-106	31-32	31-34	37-34	45-98	45-111			
LSW11	1-158	39-11							
LSW2	1-102	45-112							
LSW2S	1-131	45-119							
LSW3	1-122	1-124	14-23*	43-24	43-26	43-67*	45-113		
LSW4	1-103	4-36	14-21	31-38	40-54*	40-57*	45-114	53-44	57-14
LSW5	1-103	8-75	9-91	45-115	47-11*	47-18*	57-16		
LSW6	1-119	4-14	26-84	37-67	40-55*	40-58*	56-33	56-37*	
LSW7	1-152	31-36	41-9*	41-14*					
LSW9	1-151	12-90*	45-116						
LTRMTP	1-160	39-9							
LUNAME	1-105	37-151	46-15						
MAPPLS	1-106								
MAXCC	1-171	29-20							
MAXGVL	1-101	26-21	27-6	27-19	27-36				
MAXLD	1-104	55-30	55-46	55-52	55-60	55-68			
MAXPRI	1-132	45-90							
MAXSEC	1-138	26-165	40-24						
MINTIM	1-165	37-110							
MONABT	1-100	15-25							
MONEMT	1-89	2-111							
MOUNT	1-88	2-92							
MRKT	2-26	22-5#							
MRKTHD	1-155	22-36	22-37*	23-11	24-11				
MS#DTR	1-178	43-52	43-55						
MSEND	1-93	7-8							
MSGABT	1-100	15-31							
MSGCK	1-93	7-17							
MSGPRV	7-6	7-15	7-24	7-33#					

[illegible]

Q. CSW	1-111	51-44*						
Q. DEVX	1-97	51-28*	51-114	54-23*	54-32*			
Q. FUNC	1-143	11-46*	54-30					
Q. ICSW	1-117	8-113*	51-43	51-135*				
Q. JNUM	1-111	51-37*						
Q. JOB	1-111	51-32*						
Q. PA5	1-112	52-18*						
Q. PA6	1-85	51-25*						
Q. PAR	1-112	51-73*						
Q. UCSW	1-117	51-24*						
Q. UNIT	1-144	51-29*	54-18	54-24*	54-26*			
Q. WCNT	1-142	8-102	8-114*	51-107*	54-51			
QF#IOT	1-108	23-17	24-17					
QFREE	1-72	9-150	22-46	23-42	24-21	48-26	48-59	
QHDSFN	1-154	21-21						
QID	1-71	8-132	9-110	11-61				
QMSG	1-79	31-72	33-9#					
QNSPND	1-137	32-23	50-20					
QSET	3-15	12-42#						
QUME	1-160	39-34						
R\$CHN	1-117	1-157	4-44	4-48	13-14			
R\$DATE	1-157	20-20*						
R\$INTC	1-121	56-48						
R\$XCHN	1-117	4-48						
RC\$BAS	1-173							
RCTRLD	3-17	14-21#						
READ	1-69	2-16	9-5#					
REDCXT	1-84	37-78						
RENAME	1-86	2-12						
REOPEN	2-14	17-5#						
REQPRV	2-89	47-6#						
RF\$WRT	1-119							
RLOCK	2-66	5-26#						
RMNBAS	1-148	13-14	27-17	27-19	27-21	27-34	27-36	27-38
RPAR	1-107	53-55						
RPDR	1-107	53-53						
RSFBLK	1-177	42-38						
RSSPAC	1-175	30-19						
RT11EX	1-68	2-60#	4-29	57-42	57-44			
RTDEV	1-86	2-20						
RTEMT	1-88	2-96						
RTRSUM	1-87	2-49						
RTSPND	1-87	2-48						
RUNDEV	1-85	44-17						
S\$CPU	1-169	38-8						
S\$HICP	1-114	38-28						
S\$INWT	1-167	38-10						
S\$IOFN	1-152	38-7						
S\$IDWT	1-139	38-17	50-19					
S\$LOW	1-114	38-9						
S\$MSWT	1-167	38-15						
S\$NEDQ	1-134	38-18						
S\$OTFN	1-152	38-27						
S\$OTLO	1-113	38-6						
S\$OTWT	1-167	38-11						
S\$QCCB	1-113	38-20						

S\$QMIO	1-113	38-19											
S\$QSPD	1-113	38-21											
S\$QUSR	1-168	38-16											
S\$RT	1-114	38-29											
S\$SFWT	1-167	1-168	38-14										
S\$SPCB	1-168	38-24											
S\$SPDB	1-168	38-23											
S\$SPND	1-167	38-13											
S\$TMWT	1-153	21-20	38-12										
S\$TTFN	1-114	38-26											
S\$TTSC	1-113	38-25											
S\$TWFN	1-113	38-5	48-38										
S\$WFM	1-114	38-30											
S\$WSMB	1-149	32-22	38-22										
SB\$END	1-136	31-60											
SB\$LNK	1-137	32-30	33-17	33-19	33-21*	33-24*							
SB\$PNT	1-137	32-34*											
SB\$TXT	1-136	31-58	32-33										
SCCA	2-37	12-5#											
SCHAIN	1-123	28-9											
SCHED	1-116	50-25											
SDCLOS	1-70	18-12											
SDMOVE	1-110	8-164											
SDTTM	2-40	20-27#											
SERFLG	1-105	10-4*	10-10*										
SFRR	2-51	10-4#											
SETC	1-170	40-67											
SETCR	8-126	9-102	11-56	52-13#									
SETDSS	1-178	43-56											
SETERR	1-74	4-71	5-70	5-78	6-14	7-42	8-10	8-21	8-29	8-57	8-63	8-71	
	8-152	9-10	9-21	9-30	9-59	9-67	9-76	9-132	9-141	9-152	11-11	11-25	
	11-37	11-78	11-83	16-12	16-41	17-12	17-53	19-12	19-24	20-30	22-48	23-52	
	25-12	26-9	26-24	26-35	27-51	29-32	31-19	31-43	31-50	31-81	35-10	35-14	
	37-19	37-30	37-37	40-34	42-13	43-29	43-40	45-22	45-82	46-25	46-62	47-14	
	48-61												
SETMXP	45-88#	45-105											
SETPRI	1-88	2-104											
SETQ	8-100	9-99	11-50	51-16#									
SETRBF	1-174	30-4											
SETSIZ	2-97	12-72#											
SETSPD	1-100	43-62											
SETTOP	3-16	12-49#											
SETTP1	12-50#	12-74	12-76										
SFCLS	1-142	18-15											
SFDATE	1-86	2-42											
SFPA	2-32	12-24#											
SFPROT	1-86	2-43											
SFRSST	1-147	17-42											
SFSPND	2-82	5-53#											
SFSVST	1-136	5-54	16-22										

SPCPS	1-125	35-11	35-17*
SPFUN	2-34	11-5#	
SPLBLK	1-177	42-37	
SPLEMT	1-72	2-105	
SPLTF	45-33	45-105#	45-106
SPPRED	1-84	45-100	
SRESET	3-14	15-7	15-13#
SSEMT	1-78	2-99	
STOP	1-106	14-15	28-29 36-22 37-74
STTCPL	1-85	48-13	
SUTOP	1-135	12-93	
SVSTAT	2-13	16-5#	
SWDBLK	1-177	42-36	
SYFEMT	2-116	42-9#	
SYFTBL	42-18	42-36#	42-41
SYNAME	1-127	26-146	
SYSDAT	1-100	20-17	20-20 20-40*
SYSHLT	1-41		
SYTIMH	1-100	20-7	20-47*
SYTIML	1-100	20-9	20-49*
TBLEMT	2-88	45-14#	
TGCPRI	26-52	26-106#	
TGIDMP	26-48	26-84#	
TGJOBN	26-45	26-66#	
TGLDIN	26-46	26-72#	
TOLICN	26-51	26-101#	
TGMPRI	26-53	26-111#	
TGPRHI	26-58	26-121#	
TGPRIL	26-55	26-131#	
TGPRIV	26-47	26-77#	
TGPRLO	26-57	26-116#	
TGPRNT	26-59	26-140#	
TGPROG	26-50	26-96#	
TGPROJ	26-49	26-91#	
TGRSUB	26-61	26-156#	
TGSYNM	26-56	26-146#	
TGUCLB	26-54	26-126#	
TGVERS	26-60	26-151#	
TIMWAT	2-28	21-5#	
TRMEMT	2-95	39-8#	
TRNSTR	1-138	33-30	43-68
TRPSET	2-11	12-15#	
TRYLK	2-67	5-33#	
TSEM2	1-22#	1-69	
TSXEMT	2-64#		
TSXGVC	26-37	26-44#	26-62
TSXGVL	26-20	26-31#	
TSXSIT	1-96	26-101	40-73
TSXVRS	1-85	26-151	
TTEMT	2-106	29-17#	
TTFUN	29-24	30-4#	
TTOPTS	1-135		
TTREAD	1-110	9-45	
TTYIN	1-89	3-4	
TTYDUT	1-89	3-5	
UACHKB	1-111	51-61	51-68

[illegible]

[illegible]