

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-	9	.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	.RCTRLD
15-	1	.HRESET
15-	9	.SRESET
16-	1	.SAVESTATUS
17-	1	.REOPEN
18-	1	.PURGE
19-	1	.CLOSZ
20-	1	.WAIT
21-	1	.GTIM
21-	13	.DATE
21-	23	.SDTTM
22-	1	.TWAIT
23-	1	.MRKT
24-	1	.CMKT
26-	1	.CSTAT
26-	59	.CDFN
27-	1	.GVAL
29-	1	.EXIT
30-	1	TTEMT -- Terminal control EMT
32-	1	SNMSG -- Send a message to a line
33-	1	GTMSBF -- Get free system message buffer
34-	1	QMSG -- Queue a message for a job
35-	1	ASTXIT -- Exit from completion routine
36-	1	.SPCPS -- Alter exit address from a completion routine
37-	1	EMT376 -- EMT 376 Processing
38-	1	JBINFO -- Get information about a specific job
40-	1	Get terminal type code
40-	39	Get Project-Programmer number
41-	1	MISC. TSX EMT'S
43-	1	Return system (swap) file specification
44-	1	Set transmit/receive speed for a line
46-	1	Access TSX system tables
47-	1	Get or set user name
48-	1	Request operator privilege
49-	1	Establish Break key sentinel control
50-	1	CHKTT -- CHECK I/O TO TT DEVICE
51-	1	CKIOST -- See if scheduler wants to stop I/O
52-	1	SETQ -- Do setup of I/O queue element
53-	1	SETCR -- Set completion routine address in queue element
54-	1	CVTUAD -- Convert user address to physical address

Table of contents

55-	1	LDIO	-- Process I/O to logical disks
57-	1	INDIO	-- Perform I/O for IND data segment
58-	1	EMTTRC	-- Trace EMT execution
59-	1	PRTOCT	-- Print an octal value

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

000000
000000 020550
177776

```

        .TITLE  TSEM2  TSX-Plus EMT Overlay
        .ENABL  LC
        .ENABL  AMA
        .DSABL  GBL
;
; Copyright (C) 1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,
;           1986,1987,1988,1989.
;
; S&H Computer Systems, Inc.
; Nashville, Tennessee
;
; This software is furnished under a license for use only
; on a single computer system and may be copied only with
; the inclusion of the above copyright notice. This
; software, or any other copies thereof, may not be provided
; or otherwise made available to any other person except
; for use on such system and to one who agrees to these
; license terms. Title to and ownership of the software
; shall at all times remain with S&H Computer Systems, Inc.
;
        .CSECT  TSEM2
TSEM2:  .RAD50  /EM2/      ;Overlay id
PS      =      177776    ;Processor Status Word
;-----
; Macros to enable and disable interrupts.
;
        .MACRO  DISABL      ;DISABLE INTERRUPTS
        BIS    #340, @#PS
        .ENDM  DISABL
;
        .MACRO  ENABL      ;ENABLE INTERRUPTS
        BIC    INTPRI, @#PS
        .ENDM  ENABL
;
; Macro to print an error message when a system crash occurs.
;
; Arguments:
; MSG = Name of error message to print.
; ARG = (Optional) argument value to display with error message.
;
        .GLOBL  DIEMSG, DIEARG, SYSHLT
        .MACRO  DIE      MSG, ARG
        MOV    MSG, @#DIEMSG
        .IF    NB, ARG
        MOV    ARG, @#DIEARG
        .ENDC
        CALL  @#SYSHLT
        .ENDM  DIE
;
; Macro definition for calling global routines residing in mapped
; system regions.
;
        .MACRO  OCALL  ENTADD
        .IF    B, ENTADD
        .ERROR ;OCALL SPECIFIED WITH NO ENTRY ADDRESS
        .MEXIT
    
```

```

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

```

```

115 . GLOBL S$TTFN, S$HICP, S$RT, S$WFM, S$LOW, LPARBS, VPAR5
116 . GLOBL LSTHL
117 . GLOBL SCHED
118 . GLOBL R$CHN, CHNSIZ, R$XCHN, Q. ICSW, Q. UCSW
119 . GLOBL $FORM, CQ$CP, CP$STD
120 . GLOBL RF$WRT, LSW6, $DBGMD
121 . GLOBL EMTMAP, CHNNUM, LJSW, LSTHL
122 . GLOBL CLOSE, CLZERR
123 . GLOBL INDDBL, INDDBS, R$INTC, $INDDW
124 . GLOBL LSW3, LSLEPH, LSLEPL
125 . GLOBL PASLIN, SCHAIN, LSPND
126 . GLOBL CHNADR, EMTCAD, EMTRAD, LSW3
127 . GLOBL CORUSR, CURCP, SPCPS, $CTRLD, EMTCAS
128 . GLOBL CXTRMN, UFPTRP, JCDB, $HARD, $DEAD, CFACFL, R$CFST
129 . GLOBL VTSLCH, LBRKCQ, LBRKCH, SYNAME, IOABFL
130 . GLOBL MXJPRI, LPRI, LBSPRI, CQ$RNS, CQ$PRI
131 . GLOBL ABORT, VSWPFL, VPRIDF, VPRILO, VPRIHI
132 . GLOBL $ODTMD, LSCCA, MXJADR, FPUUSE, CW$FPU
133 . GLOBL PO$MEM, LSW2S, LIOHLD, LIOCNT
134 . GLOBL MAXPRI, LITIME, $NOINT
135 . GLOBL URO, CHKABT, NSPLDV
136 . GLOBL S$NEDQ, DVSTAT
137 . GLOBL JSWLOC, SUTOP, TTOPTS
138 . GLOBL SFSVST, $TTGAG, SB$TXT, SB$END
139 . GLOBL NMUMB, SNMSHD, QNSPND, SB$LNK, SB$PNT
140 . GLOBL LNPRIM, LMSGBF, TRNSTR, LNMAP, MAXSEC, LSECPT
141 . GLOBL EMTERR, LSTATE, S$IOWT, UMODE
142 . GLOBL CS$ERR, CS$NMX, CS$ENT
143 . GLOBL CS$EOF, CS$OPN, CS$NMX, DS$NRD
144 . GLOBL C. CSW, C. USED, Q. WCNT, CINDAT, SFCLS
145 . GLOBL C. NUMQ, Q. BLKN, Q. FUNC
146 . GLOBL C. DEVQ, Q. UNIT
147 . GLOBL C. SBLK, C. LENG
148 . GLOBL UCHAN, CSIARE, DVLBIT
149 . GLOBL LBASE, CS$SPL, SFRSST
150 . GLOBL RMNBAS
151 . GLOBL S$WSMB
152 . GLOBL LSTPL, $DILUP
153 . GLOBL LCONTM, LSW9
154 . GLOBL S$IOFN, LSW7, S$OTFN
155 . GLOBL S$TMWT
156 . GLOBL QHDSPN, NUMDEV
157 . GLOBL MRKTHD, CQ$HOT, CQ$LOT
158 . GLOBL LSTD L
159 . GLOBL R$CHN, R$DATE
160 . GLOBL LEMTPC, LSW11, $V52EM, $UDSPC
161 . GLOBL VT52, VT100, HAZEL, ADM3A, LA36, LA120
162 . GLOBL DIABLO, QUME, LTRMTP, VT2007, VT2008
163 . GLOBL $FORM
164 . GLOBL $IOMAP
165 . GLOBL CONFIG, LNSBLK
166 . GLOBL JIVLN, JIDLN, JIMLOK, JIPRIV
167 . GLOBL LSTSL, $KINIT, $MLOCK, LNBLKS, MINTIM
168 . GLOBL LPRG1, LPRG2, LCPUIH, LCPULO
169 . GLOBL S$INWT, S$OTWT, S$SPND, S$SFWT, S$MSWT
170 . GLOBL S$QUSR, S$SFWT, S$SPDB, S$SPCB
171 . GLOBL S$CPU

```

```
172          .GLOBL SETC
173          .GLOBL MAXCC
174          .GLOBL CSIARE
175          .GLOBL RC$BAS
176          .GLOBL SETRBF, CMDB, CMDC, GTSPAC, CMDE, CMDF, CMDG, CMDH
177          .GLOBL CMDI, CMDJ, CMDK, CMDL, CMDM, CMDN, CMDO, RSSPAC, SFWAC
178          .GLOBL CMDR, CMDS, CMDT, CMDU, SFWL, CMDW, CMDX, CMDY, CMDZ
179          .GLOBL SWDBLK, SPLBLK, RSFBLK, UCLDAT, INDFIL
180          .GLOBL GETDSS, SETDSS, MS$DTR
181          ; Globals used for system mapped regions
182          .GLOBL OVRHC
183          ;
184          ; EMT error code definitions
185          ;
186          000000 ECO      = 0
187          000001 EC1     = 1
188          000002 EC2     = 2
189          000003 EC3     = 3
190          000004 EC4     = 4
191          ;
```

```

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  000000G  EMTLST: .WORD  DELETE      ; 0  DELETE
9 000004  000000G          .WORD  LOOKUP     ; 1  LOOKUP
10 000006  000000G         .WORD  ENTER      ; 2  ENTER
11 000010  002762'         .WORD  TRPSET     ; 3  SET TRAP
12 000012  000000G         .WORD  RENAME     ; 4  RENAME
13 000014  003520'         .WORD  SVSTAT     ; 5  SAVE STATUS
14 000016  003644'         .WORD  REOPEN     ; 6  REOPEN
15 000020  000000G         .WORD  CLOSE     ; 7  CLOSE
16 000022  001674'         .WORD  READ      ; 10 READ(C)(W)
17 000024  001106'         .WORD  WRITE     ; 11 WRITE(C)(W)
18 000026  004236'         .WORD  DOWAIT    ; 12 WAIT
19 000030  000000G         .WORD  BADEMT    ; 13 COPY CHANNEL
20 000032  000000G         .WORD  RTDEV     ; 14 DEVICE
21 000034  005332'         .WORD  CDFN     ; 15 DEFINE CHANNELS
22 000036  000554'         .WORD  EMT16     ; 16 GROUP 16 EMTS
23 000040  000000G         .WORD  BADEMT    ; 17
24 000042  003254'         .WORD  GTJB      ; 20 GTJB
25 000044  004316'         .WORD  GETTIM    ; 21 GET TIME OF DAY
26 000046  004542'         .WORD  MRKT      ; 22 MARK TIME
27 000050  004702'         .WORD  CMKT      ; 23 CANCEL MARK TIME
28 000052  004456'         .WORD  TIMWAT    ; 24 TIMED WAIT
29 000054  000000G         .WORD  BADEMT    ; 25 SEND DATA
30 000056  000000G         .WORD  BADEMT    ; 26 RECEIVE DATA
31 000060  005170'         .WORD  CHSTAT    ; 27 CHANNEL STATUS
32 000062  003002'         .WORD  SFPA      ; 30 SET FPP EXCEPTION
33 000064  000000G         .WORD  EMTXIT    ; 31 PROTECT VECTORS
34 000066  002452'         .WORD  SPFUN     ; 32 SPECIAL DEV FUNCTIONS
35 000070  000000G         .WORD  EMTXIT    ; 33 SET EXTRA SWAP ADDRESSES
36 000072  005344'         .WORD  GVAL      ; 34 .GVAL
37 000074  002736'         .WORD  SCCA      ; 35 .SCCA
38 000076  000750'         .WORD  EMTPLX    ; 36 .CRRG, .CRAW, .MAP, etc.
39 000100  000000G         .WORD  BADEMT    ; 37 Multi-terminal support
40 000102  004370'         .WORD  SDTTM     ; 40 .SDTTM
41 000104  007202'         .WORD  ESPCPS    ; 41 .SPCPS
42 000106  000000G         .WORD  SFDATE    ; 42 .SFDATE -- Set file date
43 000110  000000G         .WORD  SFPROT    ; 43 .FPROT -- Set file protection
44 000112  000000G         .WORD  EMTXIT    ; 44 UNKNOWN!!!
45 000114  004142'         .WORD  CLOSZ     ; 45 .CLOSZ
46          000046  E375MX = <.-EMTLST>/2
47          ;
48          ;  EMT 374 FUNCTIONS
49 000116  004236'  EMT374: .WORD  DOWAIT    ; 0  WAIT
50 000120  000000G         .WORD  RTSPND    ; 1  SUSPEND THIS JOB
51 000122  000000G         .WORD  RTRSUM    ; 2  RESUME THIS JOB
52 000124  004014'         .WORD  PURGE    ; 3  PURGE A CHANNEL
53 000126  002410'         .WORD  SERR     ; 4  SET SOFT ERRORS
54 000130  002432'         .WORD  HERR     ; 5  SET HARD ERRORS
55 000132  000000G         .WORD  CLOSE    ; 6  CLOSE
56 000134  000000G         .WORD  EMTXIT    ; 7  TEST & LOCK USR
57 000136  003336'         .WORD  DOCIN     ; 10 CHAIN

```

Branch vector for processing routines

58	000140	000000G	.WORD	BADEMT	; 11	MESSAGE WAIT
59	000142	004344'	.WORD	DATE	; 12	.DATE
60	000144	000000G	.WORD	ABTIO	; 13	.ABTIO
61		000014	E374MX =	<. -EMT374>/2		
62		000062	RT11EX =	<. -EMTLST>/2		
63						
64						
65						
66	000146	000576'	TSXEMT: .WORD	CLKOPN	; 100	Open shared file channel
67	000150	000616'	.WORD	CUNLK	; 101	Unlock shared file channel
68	000152	000626'	.WORD	RLOCK	; 102	Lock a specific block
69	000154	000636'	.WORD	TRYLK	; 103	Try to lock a block
70	000156	000776'	.WORD	XMSEND	; 104	Send a message
71	000160	001016'	.WORD	XMSGCK	; 105	See if message is pending
72	000162	001036'	.WORD	XMSGWT	; 106	Wait for a message
73	000164	010404'	.WORD	XSPLSP	; 107	Get # of free spool blocks
74	000166	010266'	.WORD	XGTLN	; 110	Get TSX line #
75	000170	010462'	.WORD	XODTMD	; 111	Set/reset ODT mode
76	000172	011120'	.WORD	XGTTAB	; 112	Get table pointer
77	000174	000646'	.WORD	ULK1BK	; 113	Unlock a single shared file block
78	000176	000000G	.WORD	XHIOUT	; 114	High-efficiency tty output
79	000200	000000G	.WORD	XHIIN	; 115	High-efficiency tty input
80	000202	000000G	.WORD	XTERCK	; 116	Check for TTY input error
81	000204	000000G	.WORD	XRDTIM	; 117	Set TT read timeout
82	000206	000000G	.WORD	XHISCT	; 120	Turn high-efficiency mode on/off
83	000210	000656'	.WORD	CSFACT	; 121	Check if shared file has been modified
84	000212	000666'	.WORD	SFSPND	; 122	Suspend shared file channel
85	000214	010524'	.WORD	CKINAC	; 123	Check for input activation chars
86	000216	010542'	.WORD	SITINF	; 124	Get TSX Site information
87	000220	000606'	.WORD	CAOPN	; 125	Open shared file with access protection
88	000222	000000G	.WORD	KMNEMT	; 126	KMON EMT's
89	000224	006464'	.WORD	SNDMSG	; 127	Send a message to a line
90	000226	011172'	.WORD	TBLEMT	; 130	Access internal tsx table
91	000230	011672'	.WORD	REQPRV	; 131	Request operator privilege
92	000232	000000G	.WORD	DETEM	; 132	Detached job control emt
93	000234	011752'	.WORD	GETBRK	; 133	Establish break sentinel control (DCLBRK)
94	000236	000000G	.WORD	MOUNT	; 134	Mount a file structure
95	000240	000000G	.WORD	DISMNT	; 135	Dismount a file structure
96	000242	000000G	.WORD	PMEMT	; 136	Performance monitor EMT's
97	000244	010126'	.WORD	TRMEMT	; 137	Get terminal type
98	000246	000000G	.WORD	RTEMT	; 140	Real-time EMT's
99	000250	003132'	.WORD	SETSIZ	; 141	TSX-Plus Settop
100	000252	010236'	.WORD	GETPPN	; 142	Determine Project-Programmer number
101	000254	000000G	.WORD	SSEMT	; 143	Control shared run-time systems
102	000256	007326'	.WORD	JBINFO	; 144	Get information about a specific line
103	000260	000000G	.WORD	GFINFO	; 145	Get information about a file
104	000262	000000G	.WORD	SFTIME	; 146	Set file creation time
105	000264	011500'	.WORD	EMUNAM	; 147	Get or set user name
106	000266	000000G	.WORD	SETPRI	; 150	Set job execution priority
107	000270	000000G	.WORD	SPLEMT	; 151	Spooling control for current channel
108	000272	006322'	.WORD	TTEMT	; 152	Terminal control
109	000274	010560'	.WORD	INTEMT	; 153	Set interactive/non-interactive mode
110	000276	010704'	.WORD	EMTSPD	; 154	Set terminal transmit/receive speed
111	000300	000000G	.WORD	CLEMT	; 155	CL EMT's
112	000302	000000G	.WORD	ALCEMT	; 156	Allocate or deallocate a device
113	000304	000000G	.WORD	MONEMT	; 157	Monitor status of a job
114	000306	000000G	.WORD	CPYEMT	; 160	Copy file/priv info from another job

Branch vector for processing routines

115	000310	000730'	.WORD	CKWIN	; 161 Display window control
116	000312	000000G	.WORD	VLSEMT	; 162 Switch between processes
117	000314	000000G	.WORD	LDMEMT	; 163 Mount a logical disk structure
118	000316	010614'	.WORD	SYFEMT	; 164 Get system file dev: filspc.ext
119		000147	EMTMAX =	< -EMTLST>/2	

Branch vector for processing routines

```

1
2 ; -----
3 ; BRANCH VECTOR FOR EMTS IN THE RANGE 340 TO 357
4 000320 0000000 ; LST16: . WORD TTYIN ; 340 TTYIN
5 000322 0000000 ; . WORD TTYOUT ; 341 TTYOUT
6 000324 0000000 ; . WORD DSTAT ; 342 DSTATUS
7 000326 0000000 ; . WORD FETCH ; 343 FETCH/REL
8 000330 0000000 ; . WORD CSIGEN ; 344 CSIGEN
9 000332 0000000 ; . WORD CSISPC ; 345 CSISPC
10 000334 0000000 ; . WORD EMTXIT ; 346 LOCK USR
11 000336 0000000 ; . WORD EMTXIT ; 347 RELEASE USR
12 000340 006236' ; . WORD EXIT ; 350 EXIT
13 000342 0000000 ; . WORD PRINT ; 351 PRINT
14 000344 003420' ; . WORD SRESET ; 352 SRESET
15 000346 003046' ; . WORD QSET ; 353 QSET
16 000350 003060' ; . WORD SETTOP ; 354 SETTOP
17 000352 003370' ; . WORD RCTRLD ; 355 RCTRLD
18 000354 007150' ; . WORD ASTXIT ; 356 Completion routine exit
19 000356 003412' ; . 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 000360 113701 000000G DOEMT:  MOVB  CORUSR,R1      ;GET CURRENT JOB INDEX NUMBER
14 000364 032761 000000G 000000G      BIT   #EMTTR,LSW6(R1);IS EMT TRACING WANTED?
15 000372 001406          BEQ   2$           ;BR IF NOT
16 000374 032737 000000G 000000G      BIT   #PO$DBG,PRIVCO ;Are we authorized to do EMT tracing?
17 000402 001402          BEQ   2$           ;Br if not
18 000404 004737 014010'          CALL  EMTTRC      ;TRACE THE EMT
19         ;
20         ; Make sure the EMT function code is ok.
21         ;
22 000410 113705 000001G 2$:      MOVB  EMTBLK+1,R5      ;GET FUNCTION CODE
23 000414 020527 000147          CMP   R5,#EMTMAX    ;FUNCTION CODE OK?
24 000420 103402          BLO   5$           ;BR IF OK
25 000422 000137 000000G          JMP   BADEMT      ;INVALID EMT FUNCTION CODE
26         ;
27         ; Make sure channel number is ok.
28         ;
29 000426 120527 000122 5$:      CMPB  R5,#<140+<RT11EX-100>> ;IS THIS TSX REAL-TIME EMT?
30 000432 001443          BEQ   4$           ;BR IF YES -- ALLOW ANY CHANNEL NUMBER
31 000434 005003          CLR   R3
32 000436 153703 000000G          BISB  EMTBLK,R3      ;GET CHANNEL # WITHOUT SIGN EXTENSION
33 000442 012700 000000G          MOV   #NLCHN,R0     ;MAX # CHANNELS FOR EACH JOB
34 000446 032737 000000G 000000G      BIT   #UMODE,EMTPS  ;WAS EMT DONE IN USER OR KERNEL MODE?
35 000454 001406          BEQ   3$           ;BR IF KERNEL MODE
36 000456 032761 000000G 000000G      BIT   #INKMN,LSW4(R1);IS KMON RUNNING NOW?
37 000464 001002          BNE   3$           ;BR IF YES -- ALLOW ACCESS TO ALL CHANNELS
38 000466 012700 000000G          MOV   #NUCHN,R0     ;USER PROGRAM -- RESTRICT TO CHANS 0-15.
39 000472 020300 3$:      CMP   R3,R0           ;IS THIS CHANNEL # TOO BIG?
40 000474 103034          BHIS  CHNERR      ;BR IF YES
41         ; Save address of CSW for channel.
42 000476 010337 000000G          MOV   R3,CHNUM     ;SAVE USER'S CHANNEL NUMBER
43 000502 013700 000000G          MOV   CXTRMN,R0     ;GET ADDRESS OF SIMULATED RMON
44 000506 062700 000000G          ADD   #R$CHN,R0     ;POINT TO 1ST CHANNEL AREA
45 000512 020327 000021          CMP   R3,#21       ;IS THIS CHANNEL IN EXTENDED CHANNEL AREA?
46 000516 103404          BLO   1$           ;BR IF NOT
47 000520 162703 000021          SUB   #21,R3        ;GET CHANNEL # IN EXTENDED AREA
48 000524 062700 000000C          ADD   #R$XCHN-R$CHN,R0;POINT TO EXTENDED CHANNEL AREA
49 000530 070327 000000G 1$:      MUL   #CHNSIZ,R3    ;MULTIPLY BY # BYTES PER CHANNEL
50 000534 060003          ADD   R0,R3        ;GET ABSOLUTE ADDRESS OF CHANNEL AREA
51 000536 010337 000000G          MOV   R3,CHNADR     ;SAVE CURRENT CHANNEL ADDRESS
52         ;
53         ; Enter EMT processing routine.
54         ; R1 contains user index number.
55         ; URO contains contents of user's R0, put value here to return in R0.
56         ;
57 000542 105037 000000G 4$:      CLRB  CLZERR      ;Clear .CLOSZ error cell
    
```

```
58 000546 006305          ASL    R5          ; GET 2* FUNCTION CODE VALUE
59 000550 000175 000002'   JMP    @EMTLST(R5)    ; JUMP TO EMT PROCESSING ROUTINE
60
61                          ; -----
62                          ; Sub-function jump routine for EMT's in the range 340-357 (group 16).
63                          ;
64 000554 113705 000000G   EMT16:  MOVB    EMTBLK,R5      ; GET SUB-FUNCTION CODE
65 000560 006305          ASL    R5          ; *2
66 000562 000175 000320'   JMP    @LST16(R5)     ; ENTER GROUP-16 PROCESSING ROUTINE
67
68                          ;
69                          ; Invalid channel number
70                          ;
71 000566 012700 177770   CHNERR:  MOV    #-10,R0      ; ERROR CODE #
72 000572 000137 000000G   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 000576 004737 000702' CLKOPN: CALL  CHKSF      ;See if shared file support is available
7 000602 000177 000000G      JMP    @DOCOPN    ;Do the open
8          ;
9          ;-----
10         ; Open a shared file with access protection.
11        ;
12 000606 004737 000702' CAPOPN: CALL  CHKSF      ;See if shared file support is available
13 000612 000177 000000G      JMP    @DOOPAP    ;Open the channel
14        ;
15        ;-----
16        ; PROCESS THE .CUNLK EMT WHICH IS USED TO UNLOCK
17        ; A SHARED FILE CHANNEL.
18        ;
19 000616 004737 000702' CUNLK:  CALL  CHKSF      ;See if shared file support is available
20 000622 000177 000000G      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 000626 004737 000702' RLOCK:  CALL  CHKSF      ;See if shared file support is available
27 000632 000177 000000G      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 000636 004737 000702' TRYLK:  CALL  CHKSF      ;See if shared file support is available
34 000642 000177 000000G      JMP    @DOTLK     ;Try to lock the block
35        ;
36        ;-----
37        ; UNLOCK A SPECIFIC SHARED FILE BLOCK.
38        ;
39 000646 004737 000702' ULK1BK: CALL  CHKSF      ;See if shared file support is available
40 000652 000177 000000G      JMP    @DOULK1    ;Unlock a block
41        ;
42        ;-----
43        ; CHECK FOR WRITES TO A SHARED FILE BY OTHER USERS.
44        ;
45 000656 004737 000702' CSFACT: CALL  CHKSF      ;See if shared file support is available
46 000662 000177 000000G      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 000666 004737 000702' SFSPND: CALL  CHKSF      ;See if shared file support is available
54 000672 004777 000000G      CALL  @SFSVST    ;TELL SHARED FILE LOGIC
55 000676 000137 000000G      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 000702 005737 000000G CHKSF: TST VMXSF ;Is shared file support genned in?
62 000706 001405 BEQ 10$ ;Br if not
63 000710 032737 000000G 000000G BIT #P2$RLK,PRIVC2 ;Are we privileged to access shared files?
64 000716 001401 BEQ 10$ ;Br if not
65 000720 000207 RETURN
66 ;
67 ; We cannot use shared file support
68 ;
69 000722 005000 10$: CLR RO ;Return error code 0
70 000724 000137 000000G JMP SETERR
71
72 -----
73 ; Display window control EMT's
74 ;
75 000730 005737 000000G CKWIN: TST VMXWIN ;Is window support genned in?
76 000734 001003 BNE 1$ ;Br if yes
77 000736 005000 CLR RO ;Return error code 0
78 000740 000137 000000G JMP SETERR
79 000744 000137 000000G 1$: JMP EMTWIN ;Enter window control overlay
```

```
1  
2  
3  
4  
5  
6 000750 005737 000000G  
7 000754 001402  
8 000756 000137 000000G  
9  
10  
11  
12 000762 012700 000007  
13 000766 005037 000000G  
14 000772 000137 000000G
```

```
      .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,RO          ; Return error code 7  
          CLR      URO           ; Return 0 in RO  
          JMP      SETERR
```

```

1          .SBTTL  Message communication EMT's
2          ;-----
3          ; PROCESS THE .MSEND EMT WHICH QUEUES A MESSAGE FOR
4          ; ANOTHER USER.
5          ;
6 000776 004737 001060' XMSEND: CALL  MSGPRV      ;Do we have access to message facility?
7 001002 012700 000000G      MOV    #EMTBLK,RO  ;Point to EMT arg block
8 001006                DCALL  MSEND      ;SEND THE MESSAGE
9 001014 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 001016 004737 001060' XMSGCK: CALL  MSGPRV      ;Do we have access to message facility?
16 001022 012700 000000G      MOV    #EMTBLK,RO  ;POINT TO EMT ARG BLOCK
17 001026                DCALL  MSGCK      ;CHECK FOR MESSAGE
18 001034 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 001036 004737 001060' XMSGWT: CALL  MSGPRV      ;Do we have access to message facility?
25 001042 012700 000000G      MOV    #EMTBLK,RO  ;POINT TO EMT ARG BLOCK
26 001046                DCALL  MSGWT     ;WAIT FOR THE MESSAGE
27 001054 000137 000000G      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 001060 005737 000000G      MSGPRV: TST   VMAXMC     ;Is message facility genned into system?
34 001064 001405                BEQ   10$         ;Br if not
35 001066 032737 000000G 000000G      BIT    #P2$MSG,PRIVC2 ;Are we authorized to use message facility?
36 001074 001401                BEQ   10$         ;Br if not
37 001076 000207                RETURN
38         ;
39         ; We cannot access message facility
40         ;
41 001100 005000                10$:  CLR    RO           ;Return error code 0
42 001102 000137 000000G                JMP   SETERR
    
```

.WRITE

```

1          .SBTTL .WRITE
2          ;-----
3          ; Process a .WRITE EMT.
4          ;
5 001106 004737 012172' WRITE: CALL CKIOST ;SEE IF SCHEDULER WANTS TO STOP I/O
6 001112 013703 000000G MOV CHNADR,R3 ;GET ADDRESS OF CHANNEL
7 001116 032763 000000G 000000G BIT #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN?
8 001124 001004 BNE 6$ ;BR IF YES
9 001126 012700 000002 MOV #2,R0 ;CHANNEL NOT OPEN
10 001132 000137 000000G JMP SETERR
11          ;
12          ; Check to make sure the buffer is on a word boundary
13          ;
14 001136 032737 000001 000004G 6$: BIT #1,EMTBLK+4 ;Is buffer address even?
15 001144 001414 BEQ 15$ ;Br if yes
16 001146 016302 000000G MOV C.CSW(R3),R2 ;Get CSW
17 001152 042702 177701 BIC #^C76,R2 ;Get device table index
18 001156 032762 000000G 000000G BIT #DX$EBA,DVFLAG(R2); Does this device require even buffer addr?
19 001164 001404 BEQ 15$ ;Br if not
20 001166 012700 177766 MOV #-12,R0 ;Get abort code
21 001172 000137 000000G JMP SETERR ;Abort the operation
22          ;
23          ; Make sure word count is ok
24          ;
25 001176 005737 000006G 15$: TST EMTBLK+6 ;Is word count negative (>32727.)?
26 001202 001405 BEQ 17$ ;Br if word count zero
27 001204 100016 BPL 18$ ;Branch if valid size (positive)
28 001206 012700 177766 MOV #-12,R0 ;Get abort code
29 001212 000137 000000G 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 001216 005037 000000G 17$: CLR URO ;Say zero words read
35 001222 023727 000010G 000001 CMP EMTBLK+10,#1 ;Need to run completion routine?
36 001230 101402 BLOS 19$ ;Br if not
37 001232 004737 000000G CALL FAKCMP ;Call completion routine
38 001236 000137 000000G 19$: JMP EMTXIT ;Finished with read
39          ;
40          ; See if we are writing to TT:
41          ;
42 001242 004737 012146' 18$: CALL CHKTT ;IS CHANNEL OPENED TO "TT: "?
43 001246 103402 BCS 10$ ;BR IF NOT WRITING TO TT
44 001250 000137 000000G JMP WRITTT ;ENTER THE TT WRITE ROUTINE IN TSTTY
45          ;
46          ; See if we are writing to a spooled device.
47          ;
48 001254 032763 000000G 000000G 10$: BIT #CS$SPL,C.CSW(R3); IS THIS A SPOOLED DEVICE?
49 001262 001171 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 001264 016302 000000G MOV C.CSW(R3),R2 ;GET CSW
54 001270 032702 000000G BIT #CS$RON,R2 ;IS USER AUTHORIZED FOR READ-ONLY ACCESS?
55 001274 001404 BEQ 9$ ;BR IF NOT READ-ONLY
56 001276 012700 177763 MOV #-15,R0 ;SET ABORT ERROR CODE
57 001302 000137 000000G JMP SETERR ;ABORT EMT

```

.WRITE

```

58 001306 042702 177701          9$:   BIC    #^C76,R2      ;GET DEVICE TABLE INDEX
59 001312 032762 000000G 000000G   BIT    #DS$RON,DVSTAT(R2); IS THIS A READ-ONLY DEVICE?
60 001320 001405          BEQ    5$           ;BR IF NOT READ-ONLY
61 001322 005037 000000G          8$:   CLR    URO         ;SAY 0 WORDS WRITTEN
62 001326 005000          CLR    RO          ;RETURN ERROR CODE OF 0
63 001330 000137 000000G          JMP    SETERR
64
65          ; See if error status is pending in CSW for the channel
66
67 001334 032763 000000G 000000G 5$:   BIT    #CS$ERR,C.CSW(R3);DID A WRITE ERROR OCCUR?
68 001342 001407          BEQ    20$          ;BR IF NOT
69 001344 042763 000000G 000000G   BIC    #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
70 001352 012700 000001          MOV    #1,RO       ;RETURN ERROR CODE OF 1
71 001356 000137 000000G          JMP    SETERR
72
73          ; Check if this is an I/O operation by IND
74
75 001362 032761 000000G 000000G 20$:  BIT    #$INDRN,LSW5(R1); IS IND RUNNING?
76 001370 001402          BEQ    11$          ;BR IF NOT
77 001372 004737 013702'          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 001376 005763 000000G          11$:  TST    C.SBLK(R3)   ;Is device opened non-file-structured mode?
84 001402 001017          BNE    16$          ;Br if not
85 001404 032762 000000G 000000G   BIT    #DS$DIR,DVSTAT(R2); Is this a directory structured device?
86 001412 001413          BEQ    16$          ;Br if not
87 001414 023727 000002G 000000G   CMP    EMTBLK+2,#DH#$LB; Could this block contain directory?
88 001422 101007          BHI    16$          ;Br if not
89 001424 032737 000000G 000000G   BIT    #UMODE,EMTPS  ;Was EMT executed in user mode?
90 001432 001403          BEQ    16$          ;Br if not -- Probably TSUSR doing dir I/O
91 001434          OCALL  CLDIR    ;Clear out directory cache for device
92
93          ; Build an I/O queue entry for the write request.
94
95 001442 004737 000000G          16$:  CALL   GETQ        ;GET AN I/O QUEUE ENTRY
96
97          ; R1 now points to a free I/O queue entry.
98          ; Move arguments from EMTBLK to the queue entry.
99
100 001446 004737 012300'          CALL   SETQ        ;Do common queue setup
101 001452 103473          BCS    3$           ;Br if we must abort operation
102 001454 005761 000000G          TST    Q.WCNT(R1)   ;Is word count for transfer zero?
103 001460 001470          BEQ    3$           ;Br if yes -- Nothing to write
104 001462 020063 000000G          CMP    RO,C.USED(R3) ;Are we writing beyond previous end point?
105 001466 101416          BLOS   2$           ;Br if not
106 001470 032713 000000G          BIT    #CS$ENT,(R3) ;Was file opened with a .ENTER?
107 001474 001413          BEQ    2$           ;Br if not
108 001476 005763 000000G          TST    C.SBLK(R3)   ;Is this a directory operation?
109 001502 001004          BNE    14$          ;Br if not
110 001504 032762 000000G 000000G   BIT    #DS$DIR,DVSTAT(R2); Is this a directory structured device?
111 001512 001004          BNE    2$           ;Br if yes
112 001514 010063 000000G          14$:  MOV    RO,C.USED(R3) ;Remember high-water-mark within file
113 001520 010061 000000C          MOV    RO,Q.ICSW+C.USED(R1);Store into chan blk in Q element too
114 001524 005461 000000G          2$:   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 001530 023737 0000000 0000060      CMP      URO,EMTBLK+6      ;DID WE SHORTEN WRITE REQUEST?
120 001536 001403                      BEQ      7$              ;BR IF NOT
121 001540 052737 0000000 0000000      BIS      #CFLAG,EMTPS    ;SET C-FLAG IN PS
122 ;
123 ; Set address of write completion routine.
124 ;
125 001546 013700 0000100      7$:      MOV      EMTBLK+10,R0      ;GET ADDRESS OF COMPLETION ROUTINE
126 001552 004737 013000'      CALL     SETCR           ;STORE ADDR OF COMPL ROUTINE INTO QUEUE ELEM
127 001556 103431                      BCS      3$              ;BR IF FATAL ERROR
128 ;
129 ; We have built the I/O queue element.
130 ; Initiate the I/O operation.
131 ;
132 001560 004737 0000000      CALL     QIO             ;QUEUE AN I/O REQUEST
133 ;
134 ; See if we are writing to a shared file
135 ;
136 001564 005737 0000000      TST      JCDB           ;Does job have any shared files open?
137 001570 001402                      BEQ      12$            ;Br if not
138 001572 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 001576 005737 0000100      12$:     TST      EMTBLK+10      ;IS THIS AN .WRITW?
143 001602 001015                      BNE      1$            ;BR IF NOT
144 001604 004737 0000000      CALL     IQWAIT         ;WAIT FOR I/O TO FINISH
145 ;
146 ; See if a hardware error occurred on the write
147 ;
148 001610 032763 0000000 0000000      BIT      #CS$ERR,C.CSW(R3);DID A WRITE ERROR OCCUR?
149 001616 001407                      BEQ      1$            ;BR IF NOT
150 001620 042763 0000000 0000000      BIC      #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
151 001626 012700 0000001      MOV      #1,R0         ;RETURN ERROR CODE OF 1
152 001632 000137 0000000      JMP      SETERR
153 ;
154 ; Finished the EMT.
155 ;
156 001636 000137 0000000      1$:      JMP      EMTXIT        ;EXIT FROM EMT
157 ;
158 ; Some fatal error occurred which causes us to have to abort the operation.
159 ;
160 001642 000137 002374'      3$:      JMP      IQABRT        ;ABORT THE I/O REQUEST
161 ;
162 ; Write to a spooled device.
163 ;
164 001646                      WRTSPL: OCALL     SDMOVE      ;WRITE USER'S DATA TO SPOOL FILE
165 ;
166 ; See if we need to queue a completion routine for this write request.
167 ;
168 001654 023727 0000100 0000001      WRTCPL: CMP      EMTBLK+10,#1      ;WAS A COMPLETION ROUTINE SPECIFIED?
169 001662 101402                      BLOS     1$            ;BR IF NOT
170 001664 004737 0000000      CALL     FAKCMP         ;QUEUE A COMPLETION ROUTINE REQUEST
171 ;

```

.WRITE

```
172          ; Finished spooled write
173          ;
174 001670 000137 0000006 1$:      JMP      EMTXIT
```

.READ

```

1          .SBTTL .READ
2          ;-----
3          ; Process a .READ EMT.
4          ;
5 001674 004737 012172' READ: CALL CKIOST ;SEE IF SCHEDULER WANTS TO STOP I/O
6 001700 013703 000000G MOV CHNADR,R3 ;GET ADDRESS OF THE CHANNEL
7 001704 032763 000000G 000000G BIT #CS#OPN,C.CSW(R3); IS THE CHANNEL OPEN?
8 001712 001004 BNE 2$ ;BR IF YES
9 001714 012700 000002 MOV #2,R0 ;RETURN ERROR CODE
10 001720 000137 000000G JMP SETERR
11          ;
12          ; Check to make sure the buffer is on a word boundary
13          ;
14 001724 032737 000001 000004G 2$: BIT #1,EMTBLK+4 ;Is buffer address even?
15 001732 001414 BEQ 15$ ;Br if yes
16 001734 016302 000000G MOV C.CSW(R3),R2 ;Get CSW
17 001740 042702 177701 BIC #^C76,R2 ;Get device table index
18 001744 032762 000000G 000000G BIT #DX#EBA,DVFLAG(R2); Does this device require even buffer addr?
19 001752 001404 BEQ 15$ ;Br if not
20 001754 012700 177766 MOV #-12,R0 ;Get abort code
21 001760 000137 000000G JMP SETERR ;Abort the operation
22          ;
23          ; See if the read request is so large that
24          ; a handler might interpret it as a write
25          ;
26 001764 005737 000006G 15$: TST EMTBLK+6 ;Is word count negative (>32727.)?
27 001770 001405 BEQ 17$ ;Br if word count zero
28 001772 100016 BPL 14$ ;Branch if valid size (positive)
29 001774 012700 177766 MOV #-12,R0 ;Get abort code
30 002000 000137 000000G JMP SETERR ;Abort the operation
31          ;
32          ; Word count is zero ==> Seek.
33          ; Treat seek like nop but call completion routine if requested.
34          ;
35 002004 005037 000000G 17$: CLR URO ;Say zero words read
36 002010 023727 000010G 000001 CMP EMTBLK+10,#1 ;Need to run completion routine?
37 002016 101402 BLOS 19$ ;Br if not
38 002020 004737 000000G CALL FAKCMP ;Call completion routine
39 002024 000137 000000G 19$: JMP EMTXIT ;Finished with read
40          ;
41          ; See if we are reading from TT:
42          ;
43 002030 004737 012146' 14$: CALL CHKTT ;IS CHANNEL OPENED TO "TT: "?
44 002034 103402 BCS 10$ ;BR IF NOT TT READ
45 002036 000137 000000G JMP TTREAD ;TT READ FROM MAPPED TSTTY
46          ;
47          ; We are reading from some device other than TT.
48          ;
49          ; See if we are attempting to read from a write-only device.
50          ;
51 002042 016300 000000G 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 002046 042700 177701 BIC #^C76,R0 ;GET DEVICE TABLE INDEX FOR DEVICE
55 002052 032760 000000G 000000G BIT #DS#WON,DVSTAT(R0); IS THIS A WRITE-ONLY DEVICE?
56 002060 001405 BEQ 7$ ;BR IF NOT WRITE-ONLY
57 002062 005037 000000G 4$: CLR URO ;SAY 0 WORDS READ

```

.READ

```

58 002066 005000          CLR      RO          ;RETURN ERROR CODE 0
59 002070 000137 000000G    JMP      SETERR
60
61          ; See if error flag is pending in CSW for this channel
62          ;
63 002074 032763 000000G 000000G 7$:   BIT      #CS$ERR,C.CSW(R3);DID A HARDWARE ERROR OCCUR ON READ?
64 002102 001407          BEQ      16$          ;BR IF NOT
65 002104 042763 000000G 000000G    BIC      #CS$ERR,C.CSW(R3);ACKNOWLEDGE ERROR STATUS
66 002112 012700 000001          MOV      #1,RO          ;RETURN ERROR CODE OF 1
67 002116 000137 000000G    JMP      SETERR
68
69          ; See if end-of-file status is pending in CSW for this channel
70          ;
71 002122 032763 000000G 000000G 16$:  BIT      #CS$EOF,C.CSW(R3);DID DEVICE REPORT END OF FILE?
72 002130 001410          BEQ      18$          ;BR IF NOT
73 002132 042763 000000G 000000G    BIC      #CS$EOF,C.CSW(R3);ACKNOWLEDGE EOF STATUS
74 002140 005037 000000G          CLR      URO          ;SAY 0 WORDS READ
75 002144 005000          CLR      RO          ;RETURN ERROR CODE OF 0
76 002146 000137 000000G    JMP      SETERR
77
78          ; See if data being read is currently in the data cache
79          ;
80 002152 005737 000000G          18$:   TST      JCDB          ;Any shared file channels open for job?
81 002156 001412          BEQ      8$          ;Br if not
82 002160 005737 000000G          TST      NUMDCD        ;Are we doing shared-file data caching?
83 002164 001407          BEQ      8$          ;Br if not
84 002166 004777 000000G          CALL     @DCRD1        ;SEE IF DATA IS IN DATA CACHE
85 002172 103404          BCS      8$          ;BR IF DATA WAS NOT IN THE CACHE
86 002174 013737 000006G 000000G    MOV      EMTBLK+6,URO ;RETURN WORD COUNT READ IN USER'S RO
87 002202 000624          BR       WRTCPL        ;FINISH UP AND EXIT
88
89          ; Check for I/O operations by IND
90          ;
91 002204 032761 000000G 000000G 8$:   BIT      #INDRN,LSW5(R1);IS IND RUNNING?
92 002212 001402          BEQ      11$         ;BR IF NOT
93 002214 004737 013702'          CALL     INDIO        ;MAP I/O OPERATIONS FOR IND
94
95          ; Build I/O queue element for I/O operation.
96          ;
97 002220 004737 000000G          11$:   CALL     GETQ          ;GET A FREE I/O QUEUE ELEMENT
98          ; R1 now points to a free I/O queue element.
99 002224 004737 012300'          CALL     SETQ          ;DO COMMON QUEUE SETUP
100 002230 103461          BCS      IOABRT        ;BR IF FATAL ERROR DETECTED
101 002232 013700 000010G          MOV      EMTBLK+10,RO ;GET COMPLETION ROUTINE ADDRESS
102 002236 004737 013000'          CALL     SETCR        ;STORE COMPL RTN ADDR IN I/O QUEUE
103 002242 103004          BCC      3$          ;BR IF ADDRESS OK
104 002244 112737 177766 000000G    MOVB     #-12,INTERR   ;SET ERROR CODE
105 002252 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 002254 004737 000000G          3$:   CALL     QIO          ;QUEUE THE REQUEST
111
112          ; See if we need to update data cache
113          ;
114 002260 005737 000000G          TST      JCDB          ;Any shared file channels for job?

```

.READ

```

115 002264 001405          BEQ      12$          ;Br if not
116 002266 005737 000000G   TST      NUMDCD       ;Are we doing shared-file data caching?
117 002272 001402          BEQ      12$          ;Br if not
118 002274 004777 000000G   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 002300 005737 000010G   12$:    TST      EMTBLK+10 ;SHOULD WE WAIT?
123 002304 001031          BNE      1$          ;BR IF NOT
124 002306 004737 000000G   CALL     IOWAIT      ;WAIT FOR I/O OPERATION TO FINISH
125                               ;
126                               ; See if a hardware error occurred on the read
127                               ;
128 002312 032763 000000G 000000G BIT      #CS$ERR,C.CSW(R3);DID A HARDWARE ERROR OCCUR ON READ?
129 002320 001407          BEQ      6$          ;BR IF NOT
130 002322 042763 000000G 000000G BIC      #CS$ERR,C.CSW(R3);ACKNOWLEDGE ERROR STATUS
131 002330 012700 000001          MOV      #1,RO       ;RETURN ERROR CODE OF 1
132 002334 000137 000000G   JMP      SETERR
133                               ;
134                               ; See if the device reported end-of-file.
135                               ;
136 002340 032763 000000G 000000G 6$:    BIT      #CS$EOF,C.CSW(R3);DID DEVICE REPORT END OF FILE?
137 002346 001410          BEQ      1$          ;BR IF NOT
138 002350 042763 000000G 000000G BIC      #CS$EOF,C.CSW(R3);ACKNOWLEDGE EOF STATUS
139 002356 005037 000000G   CLR      URO         ;SAY 0 WORDS READ
140 002362 005000          CLR      RO          ;RETURN ERROR CODE OF 0
141 002364 000137 000000G   JMP      SETERR
142                               ;
143                               ; Finished operation.
144                               ;
145 002370 000137 000000G   1$:    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 002374 004737 000000G   IOABRT: CALL     QFREE          ;RETURN I/O QUEUE ENTRY TO FREE LIST
151 002400 113700 000000G   MOV      INTERR,RO    ;GET ERROR CODE
152 002404 000137 000000G   JMP      SETERR      ;ERROR RETURN

```

.SERR

```

1          .SBTTL .SERR
2          ;-----
3          ; PROCESS THE .SERR EMT
4 002410 113700 000000G SERR:  MOVB  SERFLG,RO      ; GET PREVIOUS STATE
5 002414 010037 000000G      MOV   RO,URO        ; AND RETURN TO USER
6 002420 112737 000001 000000G      MOVB  #1,SERFLG    ; REMEMBER USER ISSUED SERR
7 002426 000137 000000G      JMP   EMTXIT
8
9          .SBTTL .HERR
10         ;-----
11         ; PROCESS THE .HERR EMT
12 002432 113700 000000G HERR:  MOVB  SERFLG,RO      ; GET PREVIOUS STATE
13 002436 010037 000000G      MOV   RO,URO        ; AND RETURN TO USER
14 002442 105037 000000G      CLR   SERFLG      ; TURN OFF .SERR CONTROL
15 002446 000137 000000G      JMP   EMTXIT

```

.SPFUN

```

1          .SBTTL .SPFUN
2          ;-----
3          ; Perform special device functions.
4          ;
5 002452 004737 012172' SPFUN: CALL CKIOST ;See if scheduler wants to hold I/O starts
6 002456 013703 000000G MOV CHNADR,R3 ;GET ADDRESS OF CHANNEL BLOCK
7 002462 016302 000000G MOV C,CSW(R3),R2 ;GET CSW
8 002466 032702 000000G BIT #CS#OPN,R2 ;IS CHANNEL OPEN?
9 002472 001004 BNE 1$ ;BR IF YES
10 002474 012700 000002 MOV #2,R0 ;ERROR 2 IF CHANNEL NOT OPEN
11 002500 000137 000000G JMP SETERR
12          ;
13          ; See if device can accept special functions.
14          ;
15 002504 042702 177701 1$: BIC #177701,R2 ;EXTRACT DEVICE TABLE INDEX #
16 002510 032762 000000G 000000G BIT #DS$SFN,DVSTAT(R2); CAN DEVICE HANDLE SPECIAL FUNCTIONS?
17 002516 001002 BNE 2$ ;BR IF YES
18 002520 000137 000000G 4$: JMP EMTXIT ;IGNORE .SPFUN REQUEST
19          ;
20          ; Check function code
21          ;
22 002524 105737 000011G 2$: TSTB EMTBLK+11 ;FUNCTION CODE MUST BE NEGATIVE
23 002530 002403 BLT 3$ ;BR IF OK
24 002532 005000 CLR R0 ;ERROR CODE 0 IF NOT NEGATIVE FUNCTION
25 002534 000137 000000G JMP SETERR
26 002540 112737 000377 000010G 3$: MOVB #377,EMTBLK+10 ;MAKE SURE EVEN BYTE = 377
27          ;
28          ; Make sure user is privileged to do .SPFUN's
29          ;
30 002546 032762 000000G 000000G BIT #DS$DIR,DVSTAT(R2) ;Is this a directory-structured device?
31 002554 001413 BEQ 7$ ;Br if not
32 002556 032737 000000G 000000G BIT #PO$SPF,PRIVCO ;Is user authorized to do .SPFUN?
33 002564 001007 BNE 7$ ;Br if yes
34 002566 120237 000000G CMPB R2,LDDEVX ;Is this a .SPFUN to device LD?
35 002572 001404 BEQ 7$ ;Br if yes -- That is ok
36 002574 012700 177747 MOV #-31,R0 ;Error -31 -- Not authorized for .SPFUN
37 002600 000137 000000G JMP SETERR
38          ;
39          ; Get a free I/O queue element and set it up.
40          ;
41 002604 004737 000000G 7$: CALL GETQ ;GET A FREE I/O QUEUE ELEMENT
42          ;
43          ; R1 now points to I/O queue element.
44          ; Set special function code in queue element.
45          ;
46 002610 113761 000011G 000000G MOVB EMTBLK+11,Q.FUNC(R1); STORE SPECIAL-FUNCTION CODE VALUE
47          ;
48          ; Set up buffer address and other info in queue element
49          ;
50 002616 004737 012300' CALL SETQ ;SET UP BUFFER ADDRESS AND WORD COUNT
51 002622 103443 BCS 9$ ;BR IF SOME ERROR DETECTED
52          ;
53          ; Set up address of completion routine.
54          ;
55 002624 013700 000012G MOV EMTBLK+12,R0 ;GET ADDRESS OF COMPLETION ROUTINE
56 002630 004737 013000' CALL SETCR ;SET UP COMPL ADDR IN QUEUE ELEMENT
57 002634 103436 BCS 9$ ;BR IF INVALID COMPLETION QUEUE ADDRESS

```

```
58 ;  
59 ; Initiate the I/O operation.  
60 ;  
61 002636 004737 000000G CALL QIO ; QUEUE THE I/O REQUEST  
62 ;  
63 ; See if we should wait for the I/O operation to finish.  
64 ;  
65 002642 005737 000012G TST EMTBLK+12 ; IS WAIT REQUESTED?  
66 002646 001002 BNE 13$ ; BR IF NOT  
67 ;  
68 ; Wait for I/O to finish  
69 ;  
70 002650 004737 000000G CALL IOWAIT ; WAIT FOR I/O TO FINISH  
71 ;  
72 ; See if a hardware error or end-of-file occurred.  
73 ;  
74 002654 032763 000000G 000000G 13$: BIT #CS$ERR,C.CSW(R3); DID A HARDWARE ERROR OCCUR?  
75 002662 001407 BEQ 6$ ; BR IF NOT  
76 002664 042763 000000G 000000G BIC #CS$ERR,C.CSW(R3); CLEAR ERROR FLAG IN CSW  
77 002672 012700 000001 MOV #1,R0 ; REPORT ERROR  
78 002676 000137 000000G JMP SETERR  
79 002702 032763 000000G 000000G 6$: BIT #CS$EOF,C.CSW(R3); DID WE HIT END-OF-FILE?  
80 002710 001406 BEQ 5$ ; BR IF NOT  
81 002712 042763 000000G 000000G BIC #CS$EOF,C.CSW(R3); CLEAR END-OF-FILE FLAG IN CSW  
82 002720 005000 CLR R0 ; RETURN ERROR CODE OF 0  
83 002722 000137 000000G JMP SETERR  
84 ;  
85 ; Normal completion of .SPFUN  
86 ;  
87 002726 000137 000000G 5$: JMP EMTXIT ; FINISHED EMT  
88 ;  
89 ; Some fatal error occurred. Abort request.  
90 ;  
91 002732 000137 002374' 9$: JMP IOABRT ; ABORT I/O REQUEST
```

```

1          .SBTTL .SCCA
2          ;-----
3          ; PROCESS THE .SCCA EMT
4          ;
5 002736 016137 000000G 000000G SCCA:  MOV    LSCCA(R1),URO ;RETURN OLD SCCA RTN ADDRESS TO USER IN RO
6 002744 013700 000002G          MOV    EMTBLK+2,RO  ;GET ADDRESS OF SCCA ROUTINE
7 002750 004737 000000G          CALL   VALADW      ;VALIDATE THE ADDRESS
8 002754 010061 000000G          MOV    RO,LSCCA(R1) ;SAVE ADDRESS OF SCCA ROUTINE FOR JOB
9 002760 000406          BR     JNOP
10
11         .SBTTL .TRPSET
12         ;-----
13         ; PROCESS THE .TRPSET EMT
14         ;
15 002762 013700 000002G TRPSET: MOV    EMTBLK+2,RO  ;GET ADDRESS OF USER'S TRAP ROUTINE
16 002766 004737 000000G          CALL   VALADW      ;VALIDATE THE ADDRESS
17 002772 010037 000000G          MOV    RO,UTRPAD    ;SET TRAP ROUTINE ADDRESS FOR JOB
18 002776 000137 000000G JNOP:  JMP    EMTXIT
19
20         .SBTTL .SFPA
21         ;-----
22         ; PROCESS THE .SFPA EMT
23         ;
24 003002 013700 000002G SFPA:  MOV    EMTBLK+2,RO  ;GET ADDRESS OF TRAP ROUTINE
25 003006 004737 000000G          CALL   VALADB      ;VALIDATE IT (NOTE: ADDRESS MAY = 1)
26 003012 010037 000000G          MOV    RO,UFPTRP    ;SET ADDRESS OF TRAP ROUTINE
27 003016 005700          TST    RO            ;IS USER GOING TO USE FPU UNIT?
28 003020 001407          BEQ    2$           ;BR IF NOT
29 003022 032737 000000G 000000G          BIT    #CW$FPU,CONFIG ;DOES SYSTEM HAVE A FPU UNIT?
30 003030 001403          BEQ    2$           ;BR IF NOT
31 003032 110137 000000G          MOVB  R1,FPUUSE     ;REMEMBER USER IS ACCESSING FPU
32 003036 000402          BR     3$
33 003040 105037 000000G 2$:   CLRB  FPUUSE     ;SAY FPU NOT IN USE
34 003044 000754          3$:   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 RO.
41         ;
42 003046 013737 000004G 000000G QSET:  MOV    EMTBLK+4,URO ;Return address of Q buffer in RO
43 003054 000137 000000G          JMP    EMTXIT      ;Finished with EMT
44
45         .SBTTL .SETTOP
46         ;-----
47         ; PROCESS THE .SETTOP EMT
48         ;
49 003060 013700 000000G SETTOP: MOV    URO,RO    ;GET USER'S REQUESTED TOP ADDRESS
50 003064 005200          SETTP1: INC    RO      ;BOUND ADDRESS UP TO WORD IF IT'S ODD
51 003066 001001          BNE    2$           ;BR IF DID NOT OVERFLOW
52 003070 005300          DEC    RO          ;IF OVERFLOWED, FORCE BACK TO 177777
53 003072 042700 000001 2$:   BIC    #1,RO      ;FORCE ADDRESS TO BE EVEN
54 003076 020037 000000G          CMP    RO,ODTBAS   ;DOES REQUEST EXCEED JOB'S LIMIT?
55 003102 103404          BLO    1$           ;BR IF NOT
56 003104 013700 000000G          MOV    ODTBAS,RO  ;GET MAX AMT JOB IS ALLOWED TO USE
57 003110 162700 000002          SUB    #2,RO      ;POINT TO WORD BELOW TOP LIMIT

```

.SETTOP

```

58 003114 010037 000000G      1$:   MOV     RO,URO           ;RETURN TOP ADDRESS IN USER'S RO
59 003120 010046                MOV     RO,-(SP)        ;AND IN LOCATION 50
60 003122 106637 000050      MTPD   @#50
61 003126 000137 000000G      JMP     EMTXIT          ;FINISHED
62                               ;
63                               ;-----
64                               ; The TSX-Plus SETSIZ emt is used to change the memory space allocated
65                               ; to a job.
66                               ;
67                               ; The form of the EMT argument block is:
68                               ;
69                               ; .BYTE    0,141
70                               ; .WORD    .top-addr      ;Requested top address for job
71                               ;
72 003132 013700 000002G      SETSIZ: MOV    EMTBLK+2,RO  ;GET REQUESTED TOP OF JOB ADDRESS
73 003136 105737 000000G      TSTB   VSWPFL           ;IS THIS A NON-SWAPPING SYSTEM?
74 003142 001750                BEQ     SETTP1           ;IF NON-SWAPPING THEN CAN'T CHANGE SIZE
75 003144 120137 000000G      CMPB   R1,EXCJOB        ;DOES JOB HAVE EXCLUSIVE ACCESS TO SYSTEM?
76 003150 001745                BEQ     SETTP1           ;IF YES THEN CAN'T CHANGE SIZE
77 003152 013702 000000G      MOV    MXJADR,R2        ;GET ADDRESS ABOVE TOP OF AVAILABLE SPACE
78 003156 162702 000002      SUB    #2,R2            ;GET ADDRESS OF TOP AVAILABLE WORD
79 003162 020002                CMP     RO,R2           ;IS REQUEST ABOVE TOP ALLOWED?
80 003164 101401                BLOS   1$              ;BR IF NOT
81 003166 010200                MOV    R2,RO           ;CONSTRAIN TO TOP ALLOWED ADDRESS
82 003170 010037 000000G      1$:   MOV    RO,URO           ;RETURN TOP ADDRESS TO USER IN RO
83 003174 010046                MOV    RO,-(SP)        ;AND PUT IN USER'S LOCATION 50
84 003176 106637 000050      MTPD   @#50
85 003202 020027 160000      CMP    RO,#160000      ;IS JOB NOW USING MORE THAN 56KB?
86 003206 103412                BLO   2$              ;BR IF NOT
87 003210 052761 000000G 000000G  BIS    #VIMAGE,LJSW(R1); SET VIRTUAL-JOB FLAG
88 003216 016146 000000G      MOV    LJSW(R1),-(SP)
89 003222 106637 000000G      MTPD   @#JSWLOC
90 003226 052761 000000G 000000G  BIS    ##VIRJB,LSW9(R1); ALSO SET IN INTERNAL SYSTEM TABLE
91 003234 062700 000002      2$:   ADD    #2,RO          ;GET ADDRESS ABOVE TOP WORD WANTED
92 003240 010037 000000G      MOV    RO,ODTBAS       ;CONSTRAIN .SETTOP TO THIS
93 003244 004737 000000G      CALL   SUTOP           ;DO MEMORY ALLOCATION FOR JOB
94 003250 000137 000000G      JMP     EMTXIT          ;FINISHED

```

.GTJB

```

1
2
3
4
5 003254 013700 000002G
6 003260 004737 000000G
7 003264 010146
8 003266 006216
9 003270 106620
10 003272 013746 000000G
11 003276 106620
12 003300 005046
13 003302 106620
14 003304 012746 000000C
15 003310 106620
16 003312 005046
17 003314 106620
18 003316 010146
19 003320 006216
20 003322 106620
21 003324 013746 000000G
22 003330 106620
23 003332 000137 000000G

```

.SBTTL .GTJB

```

; PROCESS THE .GTJB EMT
;
GTJB:  MOV    EMTBLK+2,R0    ; GET ADDRESS OF USER'S BUFFER
      CALL  VALADW         ; VALIDATE BUFFER ADDRESS
      MOV   R1,-(SP)       ; RETURN JOB #
      ASR   (SP)
      MTPD  (R0)+          ; [WORD 1]
      MOV   ODTBAS,-(SP)   ; HIGH MEMORY LIMIT
      MTPD  (R0)+          ; [WORD 2]
      CLR   -(SP)         ; LOW MEMORY
      MTPD  (R0)+          ; [WORD 3]
      MOV   #RMNBAS+R#CHN-2,-(SP); START OF JOB CHANNEL AREA
      MTPD  (R0)+          ; [WORD 4]
      CLR   -(SP)         ; ADDRESS OF JOB'S IMPURE AREA
      MTPD  (R0)+          ; [WORD 5]
      MOV   R1,-(SP)       ; JOB'S TERMINAL NUMBER
      ASR   (SP)
      MTPD  (R0)+          ; [WORD 6]
      MOV   ODTBAS,-(SP)   ; VIRTUAL HIGH LIMIT
      MTPD  (R0)+          ; [WORD 7]
GTJJ:  JMP    EMTXIT

```

.CHAIN

```

1          .SBTTL .CHAIN
2          ;-----
3          ; PROCESS A .CHAIN REQUEST.
4          ;
5 003336 105237 000000G DOCIN: INCB CINFLG ; SAY .CHAIN IN PROGRESS
6          ;
7          ; Move chain data to job context area.
8          ;
9 003342 012702 000500          MOV #500,R2 ; POINT TO START OF CHAIN DATA AREA
10 003346 012703 000000G        MOV #CINDAT,R3 ; POINT TO CHAIN DATA SAVE AREA IN CONTEXT BLK
11 003352 012704 000140          MOV #<<1000-500>/2>,R4 ; GET # WORDS TO MOVE
12 003356 106522          1$: MFPD (R2)+ ; GET A WORD FROM USER'S CHAIN DATA AREA
13 003360 012623          MOV (SP)+,(R3)+ ; MOVE TO CONTEXT BLOCK AREA
14 003362 077403          SOB R4,1$
15 003364 004737 000000G        CALL STOP ; EXIT TO KMON TO DO .CHAIN
16
17          .SBTTL .RCTRL0
18          ;-----
19          ; PROCESS THE .RCTRL0 EMT
20          ;
21 003370 032761 000000G 000000G RCTRL0: BIT #CFOPN,LSW4(R1); Is input coming from a command file?
22          003376          RCTOCF == .-TSEM2
23 003376 001003          BNE 1$ ; Br if yes -- Don't reset ctrl-O
24 003400 042761 000000G 000000G        BIC #CTRL0,LSW3(R1); Reset Ctrl-O flag
25 003406 000137 000000G        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 003412 004737 000000G HRESET: CALL IOHALT ;HALT ALL I/O FOR JOB
7 003416 000400          BR SRESET ;NOW DO .SRESET
8
9          .SBTTL .SRESET
10         ;-----
11         ; PROCESS THE .SRESET AND .HRESET EMTS.
12         ;
13 003420 SRESET:
14         ;
15         ; Reset .SPND/.RSUM counter for job
16         ;
17 003420 005061 000000G CLR LSPND(R1) ;RESET .SPND COUNT FOR JOB
18         ;
19         ; Cancel any pending .MRKT requests
20         ;
21 003424 004737 005062' CALL CANMKT ;CANCEL PENDING MARK-TIME REQUESTS
22         ;
23         ; Cancel any job monitoring requests
24         ;
25 003430 DCALL MONABT ;Cancel any job monitoring
26         ;
27         ; Cancel any message-read completion routine requests
28         ;
29 003436 005737 000000G TST VMAXMC ;Is message support genned in?
30 003442 001403 BEQ 4$ ;Br if not
31 003444 DCALL MSGABT ;Abort any message-read cpl routines
32         ;
33         ; Purge I/O channels
34         ;
35 003452 012702 177777G 4$: MOV #NUCHN-1,R2 ;GET HIGHEST USER CHANNEL #
36 003456 020227 000017 1$: CMP R2,#17 ;IS THIS CHANNEL 17?
37 003462 001004 BNE 2$ ;BR IF NOT
38 003464 032761 000000G 000000G BIT #OVLBIT,LJSW(R1); IS THE PROGRAM OVERLAYED?
39 003472 001004 BNE 3$ ;IF IT IS, DON'T CLOSE CHANNEL 17
40 003474 2$: .PURGE R2 ;PURGE EACH ONE
41 003504 005302 3$: DEC R2
42 003506 002363 BGE 1$
43         ;
44         ; Undo .CDFN
45         ;
46 003510 005037 000000G CLR UCHAN ;RESET USER CHANNEL DEFINITIONS
47 003514 000137 000000G JMP EMTXIT

```

.SAVESTATUS

```

1                                     .SBTTL .SAVESTATUS
2                                     ;-----
3                                     ; Save channel status.
4                                     ;
5 003520 013703 000000G SVSTAT: MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6 003524 032763 000000G 000000G BIT     #CS#OPN,C.CSW(R3); IS THE CHANNEL OPEN?
7 003532 001417 BEQ     4$              ;BR IF NOT OPEN NOW
8                                     ; Make sure file was opened with a lookup.
9 003534 032763 000000G 000000G BIT     #CS#ENT,C.CSW(R3); OPENED WITH LOOKUP OR ENTER?
10 003542 001404 BEQ     2$              ;BR IF LOOKUP
11 003544 012700 000001 MOV     #1,R0          ;RETURN ERROR CODE 1
12 003550 000137 000000G JMP     SETERR
13                                     ;
14                                     ; Wait for any I/O activity on the channel to finish
15                                     ;
16 003554 004737 000000G 2$: CALL   IOWAIT      ;WAIT FOR I/O TO FINISH
17                                     ;
18                                     ; Tell TSX shared file logic that savestatus is being done
19                                     ;
20 003560 005737 000000G TST     JCDB          ;Any shared file channels open for job?
21 003564 001402 BEQ     4$              ;Br if not
22 003566 004777 000000G CALL   @SFSVST      ;TELL SHARED FILE SYSTEM
23                                     ;
24                                     ; Move channel block to user's area
25                                     ;
26 003572 010301 4$: MOV     R3,R1          ;SAVE ADDRESS OF CHANNEL AREA
27 003574 013700 MOV     EMTBLK+2,R0      ;GET ADDRESS OF USER'S SAVE AREA
28 003600 004737 000000G CALL   VALADW      ;VALIDATE THE ADDRESS
29 003604 012704 000005 MOV     #5,R4          ;GET # WORDS TO MOVE
30 003610 012346 3$: MOV     (R3)+,-(SP) ;MOVE DATA TO USER'S AREA
31 003612 106620 MTPD   (R0)+
32 003614 077403 SOB     R4,3$
33                                     ;
34                                     ; Mark channel as closed
35                                     ;
36 003616 032761 000000G 000000G BIT     #CS#OPN,C.CSW(R1); IS THE CHANNEL OPEN?
37 003624 001404 BEQ     5$              ;BR IF CHANNEL IS NOT OPEN
38 003626 005061 000000G CLR     C.CSW(R1)    ;PURGE THE CHANNEL
39 003632 000137 000000G JMP     EMTXIT      ;FINISHED
40 003636 005000 5$: CLR     R0          ;RETURN ERROR CODE OF 0 IF CHANNEL CLOSED
41 003640 000137 000000G JMP     SETERR
42

```

.REOPEN

```

1          .SBTTL .REOPEN
2          ;-----
3          ; Reopen a channel using saved status information.
4          ;
5 003644 013703 000000G REOPEN: MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6          ;
7          ; Make sure channel is not now open.
8          ;
9 003650 032763 000000G 000000G          BIT     #CS$OPN,C.CSW(R3); IS CHANNEL OPEN NOW?
10 003656 001403          BEQ     1$          ;BR IF NOT
11 003660 005000          CLR     R0          ;RETURN ERROR CODE OF 0
12 003662 000137 000000G          JMP     SETERR
13         ;
14         ; Channel is closed. Restore status.
15         ;
16 003666 013700 000002G 1$:      MOV     EMTBLK+2,R0      ;GET ADDRESS OF USER'S SAVE STATUS BLOCK
17 003672 004737 000000G          CALL    VALADW          ;VALIDATE THE ADDRESS
18 003676 012704 0000005          MOV     #5,R4          ;GET # WORDS TO MOVE
19 003702 106520          3$:      MFPD    (R0)+          ;GET A WORD FROM USER'S AREA
20 003704 012623          MOV     (SP)+,(R3)+      ;MOVE INTO CHANNEL BLOCK
21 003706 077403          SOB     R4,3$          ;MOVE ALL
22         ;
23         ; Check consistency of savestatus information that we restored.
24         ;
25 003710 013703 000000G          MOV     CHNADR,R3      ;POINT TO CHANNEL BLOCK
26 003714 016300 000000G          MOV     C.CSW(R3),R0   ;GET CHANNEL STATUS WORD (CSW)
27 003720 032700 000000G          BIT     #CS$OPN,R0    ;CHANNEL SHOULD BE OPEN NOW
28 003724 001425          BEQ     9$          ;ERROR IF NOT
29 003726 032700 000000G          BIT     #CS$ENT,R0    ;CHANNEL SHOULD HAVE BEEN OPENED VIA LOOKUP
30 003732 001022          BNE     9$          ;ERROR IF NOT
31 003734 105063 000000G          CLRB   C.NUMQ(R3)     ;THERE SHOULD BE NO ACTIVE I/O OPERATIONS
32 003740 126327 000000G 0000007  CMPB   C.DEVQ(R3),#7   ;CHECK VALIDITY OF UNIT NUMBER
33 003746 101014          BHI     9$          ;ERROR IF TOO BIG
34 003750 042700 000000C          BIC    #^C<<CS$NMX>>,R0 ;GET DEVICE INDEX NUMBER
35 003754 020037 000000G          CMP     R0,NUMDEV     ;MAKE SURE IT IS A VALID NUMBER
36 003760 101007          BHI     9$          ;BR IF TOO BIG
37         ;
38         ; Tell shared file system about the reopen.
39         ;
40 003762 005737 000000G          TST     JCDB          ;Does job have any shared file channels?
41 003766 001402          BEQ     4$          ;Br if not
42 003770 004777 000000G          CALL    @SFRSST      ;WE MAY BE REOPENING A SHARED FILE
43         ;
44         ; Finished
45         ;
46 003774 000137 000000G 4$:      JMP     EMTXIT
47         ;
48         ; Error -- Savestatus info we restored was invalid.
49         ; Return error code of 1.
50         ;
51 004000 005063 000000G 9$:      CLR     C.CSW(R3)    ;PURGE THE CHANNEL
52 004004 012700 0000001          MOV     #1,R0          ;RETURN ERROR CODE 1
53 004010 000137 000000G          JMP     SETERR

```

.PURGE

```

1          .SBTTL .PURGE
2          ;-----
3          ; Purge a channel.
4          ;
5 004014 013703 000000G PURGE:  MOV    CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6 004020 032763 000000G 000000G BIT    #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN NOW?
7 004026 001434          BEQ    1$          ;BR IF NOT OPEN
8 004030 004737 000000G          CALL   IOWAIT        ;WAIT FOR I/O ON CHANNEL TO FINISH
9 004034 113701 000000G          MOVVB  CHNNUM,R1      ;GET CHANNEL NUMBER
10 004040 105737 000000G          TSTB   NSPLDV        ;Are there any spooled devices?
11 004044 001403          BEQ    2$          ;BR IF NOT
12 004046          DCALL  SDCLOS        ;SEE IF THIS IS A SPOOLED DEVICE
13 004054 005737 000000G 2$:   TST    JCDB          ;Does job have any shared file channels?
14 004060 001402          BEQ    4$          ;Br if not
15 004062 004777 000000G          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 004066 016300 000000G 4$:   MOV    C.CSW(R3),R0      ;GET CHANNEL STATUS WORD
21 004072 042700 000000C          BIC    #^C<<CS$NMXX>,R0 ;MASK OUT ALL BUT DEVICE INDEX NUMBER
22 004076 032760 000000G 000000G BIT    #DS$NRD,DVSTAT(R0); IS DEVICE A MAGNETIC TAPE?
23 004104 001405          BEQ    1$          ;BR IF NOT
24 004106 005737 000000G          TST    IOABFL        ;IS IO ABORT IN EFFECT?
25 004112 001002          BNE    1$          ;BR IF YES
26 004114 000137 000000G          JMP    CLOSE        ;TREAT PURGE ON MAG TAPE LIKE CLOSE
27 004120 005063 000000G 1$:   CLR    C.CSW(R3)        ;MARK CHANNEL AS CLOSED
28          ;
29          ; See if we need to report an error from .CLOSZ
30          ;
31 004124 113700 000000G          MOVVB  CLZERR,R0      ;Do we need to report .CLOSZ error?
32 004130 001402          BEQ    3$          ;Br if not
33 004132 000137 000000G          JMP    SETERR        ;And report to user
34          ;
35          ; Successful completion
36          ;
37 004136 000137 000000G 3$:   JMP    EMTXIT        ;If no errors, done
38

```

.CLOSZ

```

1          .SBTTL .CLOSZ
2          ;-----
3          ; .CLOSZ -- Close a file at a specified size
4          ;
5 004142 013703 000000G CLOSZ:  MOV      CHNADR,R3      ;Get address of channel
6          ;
7          ; Make sure channel is open
8          ;
9 004146 032763 000000G 000000G      BIT      #CS*OPN,C.CSW(R3) ;Is the channel open?
10 004154 001002      BNE      1$          ;Br if so
11 004156 000137 000000G      JMP      EMTXIT      ;If not, ignore request
12          ;
13          ; Channel must have been opened with .ENTER
14          ;
15 004162 032763 000000G 000000G 1$:  BIT      #CS*ENT,C.CSW(R3) ;Was it opened with a .ENTER?
16 004170 001005      BNE      2$          ;Br if yes
17 004172 112737 000002 000000G      MOVB    #EC2,CLZERR ;If not, report error and just purge
18 004200 000137 004014'      JMP      PURGE      ;Go do purge
19          ;
20          ; Specified size must not exceed size of file when created
21          ;
22 004204 013700 000002G 2$:  MOV      EMTBLK+2,R0      ;Get requested size
23 004210 020063 000000G      CMP      R0,C.LENG(R3) ;Is it within valid range?
24 004214 101003      BHI      3$          ;Br if specified size beyond end of file
25 004216 010063 000000G      MOV      R0,C.USED(R3) ;If OK, make sure .CLOSE uses specified size
26 004222 000403      BR      4$          ;and go do close
27 004224 112737 000001 000000G 3$:  MOVB    #EC1,CLZERR ;If too big, keep C.USED and report error
28 004232 000137 000000G 4$:  JMP      CLOSE      ;Go do close
29

```

.WAIT

```

1          .SBTTL .WAIT
2          ;-----
3          ; .WAIT -- Wait for all I/O to finish on a channel.
4          ;
5 004236 013703 000000G DOWAIT: MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL
6          ;
7          ; Make sure channel is open
8          ;
9 004242 032763 000000G 000000G          BIT     #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN?
10 004250 001003          BNE     1$          ;BR IF YES
11 004252 005000          CLR     R0          ;RETURN ERROR CODE OF 0 IF NOT OPEN
12 004254 000137 000000G          JMP     SETERR
13          ;
14          ; Wait for all I/O to finish
15          ;
16 004260 004737 000000G 1$: CALL     IQWAIT      ;WAIT FOR I/O TO FINISH
17          ;
18          ; Check for hardware errors
19          ;
20 004264 032763 000000G 000000G          BIT     #CS$ERR,C.CSW(R3); DID A HARD ERROR OCCUR?
21 004272 001407          BEQ     9$          ;BR IF NOT
22 004274 042763 000000G 000000G          BIC     #CS$ERR,C.CSW(R3); CLEAR THE ERROR FLAG IN THE CSW
23 004302 012700 000001          MOV     #1,R0          ;Return error code 1
24 004306 000137 000000G          JMP     SETERR
25          ;
26          ; Finished
27          ;
28 004312 000137 000000G 9$: JMP     EMTXIT      ;RETURN

```

```

1          . SBTTL .GTIM
2          ;-----
3          ; Get time of day.
4          ;
5 004316 013700 000002G GETTIM: MOV EMTBLK+2,R0 ;GET ADDR OF USER'S TIME BUFFER
6 004322 004737 000000G CALL VALADW ;VALIDATE THE ADDRESS
7 004326 013746 000000G MOV SYTIMH,-(SP) ;RETURN HIGH-ORDER TIME
8 004332 106620 MTPD (R0)+
9 004334 013746 000000G MOV SYTIML,-(SP) ;RETURN LOW-ORDER TIME
10 004340 106610 MTPD (R0)
11 004342 000403 BR UPDATE ;GO UPDATE DATE CELL
12
13         . SBTTL .DATE
14         ;-----
15         ; Get current date (return to user in R0).
16         ;
17 004344 013737 000000G 000000G 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 004352 013702 000000G UPDATE: MOV CXTRMN,R2 ;GET BASE ADDRESS OF JOB VECTOR
20 004356 013762 000000G 000000G MOV SYSDAT,R#DATE(R2);SET DATE VALUE
21 004364 000137 000000G JMP EMTXIT
22
23         . SBTTL .SDTTM
24         ;-----
25         ; .SDTTM -- Set system date and time.
26         ;
27 004370 032737 000000G 000000G SDTTM: BIT #PO$OPR,PRIVCO ;IS THIS A PRIVILEGED USER RUNNING?
28 004376 001003 BNE 1$ ;BR IF YES
29 004400 005000 CLR RO ;RETURN ERROR CODE 0
30 004402 000137 000000G JMP SETERR
31
32         ; Set date
33         ;
34 004406 013702 000002G 1$: MOV EMTBLK+2,R2 ;GET ADDRESS OF DATE/TIME VALUE BLOCK
35 004412 010200 MOV R2,RO ;VALIDATE THE ADDRESS
36 004414 004737 000000G CALL VALADW
37 004420 106522 MFPD (R2)+ ;GET NEW DATE VALUE
38 004422 012600 MOV (SP)+,RO
39 004424 002402 BLT 2$ ;IF NEGATIVE THEN DON'T CHANGE DATE
40 004426 010037 000000G MOV RO,SYSDAT ;SET NEW SYSTEM DATE
41
42         ; Set time
43         ;
44 004432 106522 2$: MFPD (R2)+ ;GET HIGH-ORDER TIME VALUE
45 004434 012600 MOV (SP)+,RO
46 004436 002405 BLT 3$ ;IF NEGATIVE, DON'T CHANGE TIME VALUE
47 004440 010037 000000G MOV RO,SYTIMH ;SET NEW HIGH-ORDER TIME VALUE
48 004444 106522 MFPD (R2)+ ;GET NEW LOW-ORDER TIME VALUE
49 004446 012637 000000G MOV (SP)+,SYTIML ;SET LOW-ORDER TIME VALUE
50
51         ; Finished
52         ;
53 004452 000137 000000G 3$: JMP EMTXIT
    
```

.TWAIT

```

1          .SBTTL .TWAIT
2          ;-----
3          ; Do a timed wait
4          ;
5 004456 013700 000002G TIMWAT: MOV     EMTBLK+2,R0    ; GET POINTER TO TIME VALUE BLOCK
6 004462 004737 000000G      CALL    VALADW      ; VALIDATE THE ADDRESS
7 004466 106520          MFPD    (R0)+      ; GET HIGH-ORDER TIME VALUE
8 004470 012661 000000G      MOV     (SP)+,LSLEPH(R1)
9 004474 106510          MFPD    (R0)      ; GET LOW-ORDER TIME VALUE
10 004476 012661 000000G      MOV     (SP)+,LSLEPL(R1)
11         ;
12         ; Suspend job for timed interval
13         ;
14 004502 004737 000000G 1$:   CALL    CHKABT      ; MAKE SURE WE HAVEN'T BEEN ABORTED
15 004506 005761 000000G      TST    LSLEPH(R1)    ; IS THERE STILL SOME SLEEP TIME LEFT?
16 004512 002411          BLT    3$           ; BR IF NOT (TIME HAS GONE NEGATIVE)
17 004514 003003          BGT    2$           ; BR IF SOME TIME LEFT
18 004516 005761 000000G      TST    LSLEPL(R1)    ; CHECK LOW-ORDER VALUE
19 004522 001405          BEQ    3$           ; BR IF TIME IS UP
20 004524 012700 000000G 2$:   MOV     #S$TMWT,R0    ; STATE BECOMES TIMED WAIT
21 004530 004737 000000G      CALL    QHDSPN      ; SUSPEND THE JOB
22 004534 000762          BR     1$
23         ;
24         ; Timed wait is finished
25         ;
26 004536 000137 000000G 3$:   JMP     EMTXIT

```

```

1          .SBTTL .MRKT
2          ;-----
3          ; Start a mark-time request.
4          ;
5 004542   MRKT:
6          ;
7          ; Get an I/O queue entry.
8          ;
9 004542   004737   000000G      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 004546   113761   000000G 000000G      MOVB   CORUSR,CQ$JOB(R1);SET JOB # IN QUEUE ELEMENT
15         ; Set time value.
16 004554   013700   000002G      MOV    EMTBLK+2,R0      ;GET ADDRESS OF TIME VALUE ARG BLOCK
17 004560   004737   000000G      CALL   UACHKW         ;MAKE SURE ADDRESS IS LEGAL
18 004564   103440                   BCS   9$             ;BR IF INVALID ADDRESS
19 004566   106520                   MFPD  (R0)+          ;GET HIGH-ORDER TIME VALUE
20 004570   012661   000000G      MOV    (SP)+,CQ$HOT(R1);SET HIGH-ORDER TIME VALUE
21 004574   106510                   MFPD  (R0)          ;GET LOW-ORDER TIME VALUE
22 004576   012661   000000G      MOV    (SP)+,CQ$LOT(R1)
23         ; Set address of completion routine
24 004602   013700   000004G      MOV    EMTBLK+4,R0      ;GET ADDRESS OF COMPLETION ROUTINE
25 004606   004737   000000G      CALL   UACHKW         ;MAKE SURE ADDRESS IS LEGAL
26 004612   103425                   BCS   9$             ;BR IF NOT LEGAL
27 004614   010061   000000G      MOV    R0,CQ$RTN(R1)   ;SET ADDRESS OF COMPLETION ROUTINE
28         ; Set mapping for kernel PAR 5 that was in effect when EMT was executed.
29 004620   013761   000000G 000000G      MOV    EMTMAP,CQ$PA5(R1);SET EMT ENTRY MAPPING FOR PAR 5
30         ; Set ID number
31 004626   013761   000006G 000000G      MOV    EMTBLK+6,CQ$RO(R1);SET COMPLETION ID NUMBER
32         ;
33         ; Add queue entry to mark-time list.
34         ;
35 004634                   DISABL                   ;** DISABLE **
36 004642   013761   000000G 000000G      MOV    MRKTHD,CQ$LNK(R1);ADD TO FRONT OF LIST
37 004650   010137   000000G      MOV    R1,MRKTHD
38 004654                   ENABL                    ;** ENABLE **
39         ;
40         ; Finished
41         ;
42 004662   000137   000000G      JMP    EMTXIT
43         ;
44         ; Error -- Invalid address in MRKT argument list.
45         ;
46 004666   004737   000000G      9$:   CALL   QFREE          ;RELEASE THE QUEUE ELEMENT
47 004672   012700   177766      MOV    #-12,R0         ;RETURN ERROR CODE
48 004676   000137   000000G      JMP    SETERR         ;ABORT EMT
    
```

```

1          .SBTTL .CMKT
2          ;-----
3          ; Cancel mark-time request.
4          ;
5 004702 010103          CMKT:  MOV      R1,R3          ; COPY JOB INDEX #
6 004704 013702 000002G      MOV      EMTBLK+2,R2      ; GET ID VALUE
7 004710 001460          BEQ      1$          ; BR IF WE SHOULD CANCEL ALL REQUESTS FOR JOB
8          ;
9          ; Cancel a specific mark-time request for this job.
10         ;
11 004712 012704 000000C      MOV      #MRKTHD-CQ$LNK,R4;FAKE UP POINTER TO LIST HEAD
12 004716          DISABL          ;** DISABLE **
13 004724 016401 000000G      4$:  MOV      CQ$LNK(R4),R1      ; GET ADDRESS OF NEXT MARK-TIME QUEUE ENTRY
14 004730 001442          BEQ      5$          ; BR IF REACHED END OF LIST
15 004732 120361 000000G      CMPB     R3,CQ$JOB(R1)      ; DOES THIS ENTRY BELONG TO OUR JOB?
16 004736 001007          BNE      2$          ; BR IF NOT
17 004740 132761 000000G 000000G  BITB     #QF$IOT,CQ$FLG(R1); IS THIS A .TIMID ENTRY?
18 004746 001003          BNE      2$          ; BR IF YES
19 004750 020261 000000G      CMP      R2,CQ$RO(R1)      ; DO ID NUMBERS MATCH?
20 004754 001402          BEQ      3$          ; BR IF YES -- FOUND ENTRY TO DELETE
21 004756 010104          2$:  MOV      R1,R4          ; CHAIN FORWARD TO NEXT ENTRY IN LIST
22 004760 000761          BR      4$
23         ;
24         ; Found specified queue entry -- delete from list.
25         ;
26 004762 016164 000000G 000000G 3$:  MOV      CQ$LNK(R1),CQ$LNK(R4);RELINK LIST AROUND US
27 004770          ENABL          ;** ENABLE **
28         ;
29         ; See if we need to return remaining time.
30         ;
31 004776 013700 000004G      MOV      EMTBLK+4,RO      ; DOES USER WANT REMAINING TIME?
32 005002 001411          BEQ      6$          ; BR IF NOT
33 005004 004737 000000G      CALL     UACHKW          ; MAKE SURE RECIEVING AREA IS LEGAL
34 005010 103406          BCS      6$          ; BR IF NOT LEGAL
35 005012 016146 000000G      MOV      CQ$HOT(R1),-(SP);RETURN HIGH-ORDER TIME
36 005016 106620          MTPD     (RO)+
37 005020 016146 000000G      MOV      CQ$LOT(R1),-(SP);RETURN LOW-ORDER TIME
38 005024 106610          MTPD     (RO)
39         ;
40         ; Return queue element to free list.
41         ;
42 005026 004737 000000G      6$:  CALL     QFREE          ; FREE THE Q ELEMENT
43         ;
44         ; Finished
45         ;
46 005032 000137 000000G      JMP      EMTXIT
47         ;
48         ; Could not find specified queue element -- return error code 0.
49         ;
50 005036          5$:  ENABL          ;** ENABLE **
51 005044 005000          CLR      RO          ; RETURN ERROR CODE 0
52 005046 000137 000000G      JMP      SETERR
53         ;
54         ; Cancel all mark-time requests for this job.
55         ;
56 005052 004737 005062'      1$:  CALL     CANMKT          ; CANCEL ALL MARK-TIME REQUESTS FOR JOB
57 005056 000137 000000G      JMP      EMTXIT          ; FINISHED
    
```

```

1 ; -----
2 ; CANMKT is called to cancel all mark-time requests for the current job.
3 ; It is called when a .CMKT EMT is done with an id of 0 or when a job
4 ; exits.
5 ;
6 ; Inputs:
7 ; MRKTHD = Head of mark-time queue chain.
8 ;
9 005062 010146 CANMKT: MOV R1,-(SP)
10 005064 010446 MOV R4,-(SP)
11 005066 012704 000000C 1$: MOV #MRKTHD-CQ$LNK,R4; DUMMY POINTER TO LIST HEAD
12 005072 DISABL ;** DISABLE **
13 005100 016401 000000G 2$: MOV CQ$LNK(R4),R1 ; GET NEXT ENTRY IN LIST
14 005104 001423 BEQ 4$ ; BR IF REACHED END OF LIST
15 005106 123761 000000G 000000G CMPB CORUSR,CQ$JOB(R1); IS THIS ENTRY FOR THIS JOB?
16 005114 001015 BNE 3$ ; BR IF NOT
17 005116 132761 000000G 000000G BITB #QF$IOT,CQ$FLG(R1); IS THIS A HANDLER .TIMID ENTRY?
18 005124 001011 BNE 3$ ; BR IF YES
19 005126 016164 000000G 000000G MOV CQ$LNK(R1),CQ$LNK(R4); RELINK LIST AROUND US
20 005134 ENABL ;** ENABLE **
21 005142 004737 000000G CALL QFREE ; RETURN QUEUE ELEMENT TO FREE LIST
22 005146 000747 BR 1$
23 005150 010104 3$: MOV R1,R4 ; LINK TO NEXT ELEMENT IN LIST
24 005152 000752 BR 2$
25 ;
26 ; Finished
27 ;
28 005154 4$: ENABL ;** ENABLE **
29 005162 012604 MOV (SP)+,R4
30 005164 012601 MOV (SP)+,R1
31 005166 000207 RETURN
    
```

.CSTAT

```

1          .SBTTL .CSTAT
2          ;-----
3          ; Return channel status
4          ;
5 005170 013703 000000G CHSTAT: MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6          ;
7          ; Make sure channel is open or at least has device index
8          ;
9 005174 032763 000000G 000000G          BIT     #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN?
10 005202 001004          BNE     1$              ;BR IF YES
11 005204 032763 000000G 000000G          BIT     #CS$NMX,C.CSW(R3); IS THERE ANY DEVICE INFORMATION?
12 005212 001444          BEQ     7$              ;IMMEDIATE ERROR EXIT IF NOT
13          ;
14          ; Channel is open. Return status.
15          ; Also if there is device information available, but then return error.
16          ;
17 005214 013704 000002G 1$:      MOV     EMTBLK+2,R4      ;GET ADDRESS OF USER'S STATUS BLOCK
18 005220 010400          MOV     R4,R0              ;VALIDATE ADDRESS
19 005222 004737 000000G          CALL    VALADW
20          ; Point R0 to user's channel status block passed in EMT argument.
21 005226 010437 000000G          MOV     R4,URO          ;Point R0 to channel status block
22          ; Return CSW
23 005232 016346 000000G          MOV     C.CSW(R3),-(SP)
24 005236 106624          MTPD   (R4)+
25          ; Return starting block number of file
26 005240 016346 000000G          MOV     C.SBLK(R3),-(SP)
27 005244 106624          MTPD   (R4)+
28          ; Return allocated length of file.
29 005246 016346 000000G          MOV     C.LENG(R3),-(SP)
30 005252 106624          MTPD   (R4)+
31          ; Return highest block number written so far
32 005254 016346 000000G          MOV     C.USED(R3),-(SP)
33 005260 106624          MTPD   (R4)+
34          ; Return device unit number.
35 005262 005046          CLR     -(SP)              ;ENSURE HIGH BYTE IS CLEAR
36 005264 116316 000000G          MOVB   C.DEVQ(R3),(SP) ;MOVE UNIT NUMBER INTO LOW BYTE
37 005270 106624          MTPD   (R4)+
38          ; Return Rad50 device name.
39 005272 016301 000000G          MOV     C.CSW(R3),R1      ;GET DEVICE TABLE #
40 005276 042701 177701          BIC     #^C76,R1         ;GET DEV # ONLY
41 005302 016146 000000G          MOV     PNAME(R1),-(SP) ;GET RAD50 DEVICE NAME
42 005306 106624          MTPD   (R4)+
43          ;
44          ; Normal or error return?
45          ;
46 005310 032763 000000G 000000G          BIT     #CS$OPN,C.CSW(R3); IS CHANNEL OPEN?
47 005316 001402          BEQ     7$              ;ERROR RETURN IF NOT
48          ;
49          ; Normal exit
50          ;
51 005320 000137 000000G          JMP     EMTXIT
52          ;
53          ; Error return
54          ;
55 005324 005000 7$:      CLR     R0              ;RETURN ERROR CODE OF 0
56 005326 000137 000000G          JMP     SETERR
57

```

58

59

.SBTTL .CDFN

60

61

; Define channels.

62

;

63 005332 013737 0000020 0000000 CDFN: MOV EMTBLK+2, UCHAN ;SAVE ADDRESS OF USER CHANNEL SPACE

64 005340 000137 0000000 JMP EMTXIT

.GVAL

```

1          .SBTTL .GVAL
2          ;-----
3          ; .GVAL, .PVAL, .PEEK, .POKE
4          ;
5 005344 113700 000000G GVAL:  MOVB  EMTBLK,R0      ;GET SUB-FUNCTION CODE (.GVAL/.PVAL etc.)
6 005350 120027 000003      CMPB  R0,#3          ;MAKE SURE IT IS REASONABLE
7 005354 101404      BLOS  1$           ;BR IF OK
8 005356 012700 177767      MOV   #-11,R0       ;INVALID FUNCTION CODE
9 005362 000137 000000G      JMP   SETERR
10 005366 006300      1$:  ASL   R0          ;GET 2 * FUNCTION CODE
11 005370 013702 000002G      MOV   EMTBLK+2,R2   ;GET OFFSET VALUE
12 005374 013703 000004G      MOV   EMTBLK+4,R3   ;GET VALUE TO STORE (.PVAL & .POKE)
13 005400 000170 006226'      JMP   @GVJMP(R0)    ;JUMP TO PROCESSING ROUTINE
14          ;
15          ; .GVAL -- Get value from RMON table
16          ;
17 005404 032702 000001      DOGVAL: BIT  #1,R2      ;OFFSET CANNOT BE ODD
18 005410 001005      BNE  2$           ;BR IF ODD
19 005412 005702      TST  R2          ;IS OFFSET VALUE NEGATIVE?
20 005414 002427      BLT  TSXGVL       ;IF YES THEN HE'S GETTING A TSX-PLUS VALUE
21 005416 020227 000000G      CMP  R2,#MAXGVL    ;IS IT TOO BIG?
22 005422 103403      BLO  1$           ;BR IF OK
23 005424 005000      2$:  CLR  R0          ;RETURN ERROR CODE OF 0
24 005426 000137 000000G      JMP  SETERR
25 005432 013700 000000G      1$:  MOV  CXTRMN,R0     ;GET RMON OFFSET FOR CMD FILE STATUS WORD
26 005436 005060 000000G      CLR  R#CFST(R0)    ;CLEAR COMMAND FILE STATUS WORD
27 005442 005737 000000G      TST  CFPNT        ;ARE COMMAND FILES ACTIVE?
28 005446 001403      BEQ  3$           ;BR IF NOT
29 005450 052760 000000G 000000G      BIS  #CFACFL,R#CFST(R0);SAY COMMAND FILE ACTIVE
30 005456 063702 000000G      3$:  ADD  CXTRMN,R2     ;ADD BASE ADDR OF USER'S RMON AREA
31 005462 016237 000002 000000G      MOV  2(R2),URO     ;RETURN VALUE IN USER'S R0
32 005470 000137 000000G      JMP  EMTXIT
33          ;
34          ; Get TSX-Plus system value
35          ;
36 005474 005402      TSXGVL: NEG  R2          ;GET POSITIVE OFFSET VALUE
37 005476 020227 000042      CMP  R2,#MXTSXV    ;IS IT IN LEGAL RANGE?
38 005502 101403      BLOS  1$           ;BR IF OK
39 005504 005000      CLR  R0          ;RETURN ERROR CODE 0
40 005506 000137 000000G      JMP  SETERR
41 005512 005000      1$:  CLR  R0          ;INITIALIZE VALUE TO ZERO
42 005514 016202 005530'      MOV  TSXGVC(R2),R2 ;GET ADDRESS OF ROUTINE TO SET VALUE
43 005520 004712      CALL @R2          ;CALL ROUTINE TO GET VALUE
44 005522 010037 000000G      MOV  R0,URO       ;RETURN VALUE TO USER IN R0
45 005526 000137 000000G      JMP  EMTXIT        ;FINISHED
46          ;
47          ; TSX-Plus value offset vector
48          ;
49          TSXGVC = .-2
50 005532 005574'      .WORD TGJOB#      ; -2. -- Get job #
51 005534 005602'      .WORD TGLDIN      ; -4. -- Get Lead-in character
52 005536 005610'      .WORD TGPRIV      ; -6. -- Determine if this is a privileged job
53 005540 005624'      .WORD TGIOMP      ; -8. -- Determine if I/O page mapping in effect
54 005542 005640'      .WORD TGPROJ      ; -10. -- Job's project number
55 005544 005646'      .WORD TGPROG      ; -12. -- Job's programmer number
56 005546 005654'      .WORD TGLICN      ; -14. -- TSX-Plus site license number
57 005550 005662'      .WORD TGCPRI      ; -16. -- Current execution priority

```

.GVAL

```

58 005552 005670' .WORD TGMPRI ; -18. -- Maximum authorized priority
59 005554 005712' .WORD TGUCLB ; -20. -- Number of data blocks/job in UCL file
60 005556 005720' .WORD TGPRIL ; -22. -- Primary line #
61 005560 005746' .WORD TGSYNM ; -24. -- Name of physical SY device
62 005562 005676' .WORD TGPRLO ; -26. -- Get value of PRILOW
63 005564 005704' .WORD TGPRHI ; -28. -- Get value of PRIHI
64 005566 005736' .WORD TGPRNT ; -30. -- Get parent job number
65 005570 005754' .WORD TGVERS ; -32. -- Get system version number
66 005572 005762' .WORD TGRSUB ; -34. -- Get relative subprocess number
67 000042 MXTSXV = <. -TSXGVC>-2 ; Maximum legal offset value
68 ;
69 ; Get job number
70 ;
71 005574 010100 TGJOBN: MOV R1,RO ; GET JOB INDEX NUMBER
72 005576 006200 ASR RO ; CONVERT TO JOB NUMBER
73 005600 000207 RETURN
74 ;
75 ; Get Lead-in character
76 ;
77 005602 113700 000000G TGLDIN: MOVB VTSLCH,RO ; GET LEAD-IN CHARACTER VALUE
78 005606 000207 RETURN
79 ;
80 ; Determine if this is a privileged job
81 ;
82 005610 032737 000000G 000000G TGPRIV: BIT #PO$SYS,PRIVCO ; ARE WE PRIVILEGED?
83 005616 001401 BEQ 1$ ; BR IF NOT
84 005620 005200 INC RO ; SET PRIVILEGED FLAG
85 005622 000207 1$: RETURN
86 ;
87 ; Determine if I/O page mapping is in effect
88 ;
89 005624 032761 000000G 000000G TGIOMP: BIT #IOMAP,LSW6(R1); IS I/O PAGE MAPPING IN EFFECT?
90 005632 001401 BEQ 1$ ; BR IF NOT
91 005634 005200 INC RO ; SAY MAPPING IS IN EFFECT
92 005636 000207 1$: RETURN
93 ;
94 ; Get Project number
95 ;
96 005640 016100 000000G TGPROJ: MOV LPROJ(R1),RO ; GET PROJECT #
97 005644 000207 RETURN
98 ;
99 ; Get programmer number
100 ;
101 005646 016100 000000G TGPROG: MOV LPROG(R1),RO ; GET PROGRAMMER #
102 005652 000207 RETURN
103 ;
104 ; Get TSX-Plus site license number
105 ;
106 005654 013700 000000G TGLICN: MOV TSXSIT,RO ; GET LICENSE NUMBER
107 005660 000207 RETURN
108 ;
109 ; Get current job execution priority
110 ;
111 005662 116100 000000G TGCPRI: MOVB LPRI(R1),RO ; Get current job priority
112 005666 000207 RETURN
113 ;
114 ; Get maximum authorized job priority

```

.GVAL

```

115 ;
116 005670 113700 000000G TGMPRI: MOVB MXJPRI,RO ;Get maximum authorized execution priority
117 005674 000207 RETURN
118 ;
119 ; Get value of PRILOW
120 ;
121 005676 113700 000000G TGPRLO: MOVB VPRILO,RO
122 005702 000207 RETURN
123 ;
124 ; Get value of PRIHI
125 ;
126 005704 113700 000000G TGPRHI: MOVB VPRIHI,RO
127 005710 000207 RETURN
128 ;
129 ; Get number of blocks per job in TSXUCL data base file
130 ;
131 005712 013700 000000G TGUCLB: MOV UCLBLK,RO ;Get number of blocks per job
132 005716 000207 RETURN
133 ;
134 ; Get number of controlling primary line (0 if we are a primary line)
135 ;
136 005720 016100 000000G TGPRIL: MOV LNPRIM(R1),RO ;Get number of our primary line
137 005724 020001 CMP RO,R1 ;Are we the primary line?
138 005726 001001 BNE 1$ ;Br if not
139 005730 005000 CLR RO ;Return 0 if we are a primary line
140 005732 006200 1$: ASR RO ;Convert from 2*job # to job #
141 005734 000207 RETURN
142 ;
143 ; Get number of parent job
144 ;
145 005736 016100 000000G TGPRNT: MOV LPARNT(R1),RO ;Get parent job index number
146 005742 006200 ASR RO ;Convert to job number
147 005744 000207 RETURN
148 ;
149 ; Get physical name of SY device
150 ;
151 005746 013700 000000G TGSYNM: MOV SYNAME,RO ;Get SY physical device name
152 005752 000207 RETURN
153 ;
154 ; Get system version number
155 ;
156 005754 012700 000000G TGVERS: MOV #TSXVRS,RO ;Get system version number
157 005760 000207 RETURN
158 ;
159 ; Get relative subprocess number
160 ;
161 005762 010546 TGRSUB: MOV R5,-(SP) ;Save R5
162 005764 005000 CLR RO ;Primary and detached are relative 0
163 005766 020127 000000G CMP R1,#LSTD L ;Is this a virtual line?
164 005772 003414 BLE B$ ;Branch if not
165 005774 016105 000000G MOV LNPRIM(R1),R5 ;Get owner line number
166 006000 016505 000000G MOV LSECPT(R5),R5 ;Get pointer to owner's subprocess list
167 006004 005200 1$: INC RO ;Count subprocess number
168 006006 120125 CMPB R1,(R5)+ ;Is this our line?
169 006010 001405 BEQ B$ ;Done if line matches
170 006012 020027 000000G CMP RO,#MAXSEC ;Any more possible?
171 006016 002772 BLT 1$ ;Keep checking if so

```

172 006020 012700 177777
173 006024 012605
174 006026 000207

B#: MOV #-1,R0 ;Whoa! This should never happen!
MOV (SP)+,R5 ;Restore R5
RETURN

.GVAL

```

1
2 ; .PVAL -- Store value into monitor table
3 ;
4 006030 032702 000001 PVAL: BIT #1,R2 ; IS ADDRESS ODD?
5 006034 001071 BNE GVBAD ; BR IF YES -- ERROR
6 006036 020227 000000G CMP R2,#MAXGVL ; IS VALUE IN PROPER RANGE?
7 006042 101066 BHI GVBAD ; BR IF NOT
8 006044 063702 000000G ADD CXTRMN,R2 ; ADD BASE ADDRESS OF JOB'S RMON AREA
9 006050 016237 000002 000000G MOV 2(R2),URO ; RETURN ORIGINAL VALUE IN RO
10 006056 010362 000002 MOV R3,2(R2) ; STORE NEW VALUE
11 006062 000137 000000G JMP EMTXIT ; FINISHED
12 ;
13 ; .PEEK -- Get value from specified memory location
14 ;
15 006066 032702 000001 PEEK: BIT #1,R2 ; IS ADDRESS ODD?
16 006072 001052 BNE GVBAD ; BR IF YES -- ERROR
17 006074 020227 000000G CMP R2,#RMNBAS ; IS ADDRESS WITHIN RMON TABLE?
18 006100 103413 BLO PKLOW ; BR IF TOO LOW
19 006102 020227 000000C CMP R2,#RMNBAS+MAXGVL ; IS IT TOO BIG?
20 006106 101010 BHI PKLOW ; BR IF YES
21 006110 162702 177776G SUB #<RMNBAS-2>,R2 ; MAKE ADDRESS INTO OFFSET
22 006114 063702 000000G ADD CXTRMN,R2 ; ADD BASE ADDRESS OF JOB'S RMON AREA
23 006120 011237 000000G PKGET: MOV (R2),URO ; RETURN VALUE
24 006124 000137 000000G JMP EMTXIT
25 ; Peek into low memory area or I/O page
26 006130 032737 000000G 000000G PKLOW: BIT #PO$MEM,PRIVCO ; Are we authorized to access phys memory?
27 006136 001430 BEQ GVBAD ; BR IF NOT
28 006140 000767 BR PKGET ; GO GET THE VALUE
29 ;
30 ; .POKE -- Store value into specified memory location
31 ;
32 006142 032702 000001 POKE: BIT #1,R2 ; IS ADDRESS ODD?
33 006146 001024 BNE GVBAD ; ERROR IF YES
34 006150 020227 000000G CMP R2,#RMNBAS ; IS ADDRESS WITHIN RMON TABLE?
35 006154 103414 BLO POLOW ; BR IF TOO LOW
36 006156 020227 000000C CMP R2,#RMNBAS+MAXGVL ; IS IT TOO BIG?
37 006162 101011 BHI POLOW ; BR IF YES
38 006164 162702 177776G SUB #<RMNBAS-2>,R2 ; CONVERT TO OFFSET WITHIN TABLE
39 006170 063702 000000G ADD CXTRMN,R2 ; ADD BASE OF JOB'S RMON AREA
40 006174 011237 000000G POVAL: MOV (R2),URO ; GET ORIGINAL VALUE
41 006200 010312 MOV R3,(R2) ; SET NEW VALUE
42 006202 000137 000000G JMP EMTXIT ; FINISHED
43 ; Poke into low memory or I/O page
44 006206 032737 000000G 000000G POLOW: BIT #PO$MEM,PRIVCO ; Are we authorized to access phys memory?
45 006214 001401 BEQ GVBAD ; BR IF NOT
46 006216 000766 BR POVAL ; GO STORE VALUE
47 ;
48 ; Invalid offset specified
49 ;
50 006220 005000 GVBAD: CLR RO ; RETURN ERROR # 0
51 006222 000137 000000G JMP SETERR
52 ;
53 ; Sub-function jump vector
54 ;
55 006226 005404' GVJMP: .WORD DOGVAL ; 0 - .GVAL
56 006230 006066' .WORD PEEK ; 1 - .PEEK
57 006232 006030' .WORD PVAL ; 2 - .PVAL

```

58 006234 006142'

.WORD POKE

;3 - .POKE

.EXIT

```

1          .SBTTL .EXIT
2          ;-----
3          ; Program exit
4          ;
5 006236 005061 000000G 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 006242 032761 000000C 000000G BIT #PASLIN!SCHAIN,LJSW(R1); DOES PROGRAM WANT TO PASS COMMANDS?
10 006250 001422 BEQ 2$ ; BR IF NOT
11          ; Move commands from job chain data area to context area chain data cell.
12 006252 012704 000510 MOV #510,R4 ; ADDRESS OF CELL WITH BYTE COUNT
13 006256 106524 MFPD (R4)+ ; GET COUNT OF # BYTES TO MOVE
14 006260 012602 MOV (SP)+,R2
15 006262 020227 000266 CMP R2,#<1000-512> ; LIMIT TO LARGEST POSSIBLE SIZE
16 006266 101402 BLOS 3$ ; BR IF OK
17 006270 012702 000266 MOV #<1000-512>,R2
18 006274 012703 000010G 3$: MOV #CINDAT+10,R3 ; POINT TO CHAIN DATA CELL IN CONTEXT AREA
19 006300 010223 MOV R2,(R3)+ ; STORE BYTE COUNT
20 006302 001405 BEQ 2$ ; BR IF NULL COMMAND BEING PASSED TO US
21 006304 005202 INC R2 ; ROUND UP
22 006306 006202 ASR R2 ; GET # WORDS TO MOVE
23 006310 106524 1$: MFPD (R4)+ ; GET NEXT WORD OF DATA TO MOVE
24 006312 012623 MOV (SP)+,(R3)+ ; MOVE TO CONTEXT AREA
25 006314 077203 SOB R2,1$ ; MOVE ALL OF COMMANDS
26          ;
27          ; Stop program execution and enter TSKMON
28          ;
29 006316 004737 000000G 2$: CALL STOP ; TERMINATE PROGRAM EXECUTION

```

```

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 006322 013702 000002G  TTEMT:  MOV      EMTBLK+2,R2      ;Get function_value
18 006326 162702 000101      SUB      #'A,R2      ;Convert letter into index value
19 006332 100416              BMI      9$          ;Br if not legal value
20 006334 020227 000000G      CMP      R2,#MAXCC    ;Is the letter ok?
21 006340 103013              BHIS     9$          ;Br if too big
22 006342 006302              ASL      R2          ;Convert to word table index
23 006344 013700 000004G      MOV      EMTBLK+4,R0  ;Get control_value
24 006350 016237 006400' 006362'  MOV      TTFUN(R2),1$  ;Set address of processing routine
25 006356 004737 000000G      CALL     OVRHC       ;Call overlay handler
26 006362 000000      1$:      .WORD  0          ;Store address of routine to call here
27 006364 000137 000000G      JMP      EMTXIT      ;Finished
28         ;
29         ; Error -- Invalid function value
30         ;
31 006370 012700 000001      9$:      MOV      #1,R0      ;Return error code 1
32 006374 000137 000000G      JMP      SETERR

```

```
1  
2 ; -----  
3 ; Vector of addresses of processing routines  
4 TTFUN: .WORD SETRBF ; A  
5 .WORD CMDB ; B  
6 .WORD CMDC ; C  
7 .WORD GTSPAC ; D  
8 .WORD CMDE ; E  
9 .WORD CMDF ; F  
10 .WORD CMDG ; G  
11 .WORD CMDH ; H  
12 .WORD CMDI ; I  
13 .WORD CMDJ ; J  
14 .WORD CMDK ; K  
15 .WORD CMDL ; L  
16 .WORD CDM ; M  
17 .WORD CMDN ; N  
18 .WORD CMDO ; O  
19 .WORD RSSPAC ; P  
20 .WORD SFWAC ; Q  
21 .WORD CMDR ; R  
22 .WORD CMDS ; S  
23 .WORD CMDT ; T  
24 .WORD CMDU ; U  
25 .WORD SFWL ; V  
26 .WORD CMDW ; W  
27 .WORD CMDX ; X  
28 .WORD CMDY ; Y  
29 .WORD CMDZ ; Z
```

```

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 006464 032737 000000G 000000G SNDMSG: BIT    #PO$SND,PRIVCO ;Are we authorized to send messages?
17 006472 001003          BNE    4$          ;Br if yes
18 006474 005000          CLR    RO          ;Return error code 0
19 006476 000137 000000G          JMP    SETERR
20 006502 032737 000000G 000000G 4$:  BIT    #PO$OPR,PRIVCO ;Does user have OPER privilege?
21 006510 001003          BNE    5$          ;Br if yes
22 006512 042737 000001 000000G          BIC    #1,EMTBLK ;Cannot override gag unless oper privilege
23         ;
24         ; See if a valid job number was specified.
25         ;
26 006520 013703 000002G          5$:  MOV    EMTBLK+2,R3 ;GET NUMBER OF LINE TO RECEIVE MESSAGE
27 006524 006303          ASL    R3          ;CONVERT TO JOB INDEX NUMBER
28 006526 020327 000002          CMP    R3,#2      ;CAN'T BE LESS THAN 2
29 006532 103473          BLO    9$          ;BR IF TOO SMALL
30 006534 020327 000000G          CMP    R3,#LSTSL ;MAKE SURE NOT TOO LARGE
31 006540 101070          BHI    9$          ;BR IF TOO LARGE
32 006542 032763 000000G 000000G          BIT    #$DILUP,LSW(R3) ;IS JOB LOGGED ON?
33 006550 001464          BEQ    9$          ;BR IF NOT
34 006552 032763 000000G 000000G          BIT    #$DETCH,LSW(R3) ;IS JOB A DETACH LINE?
35 006560 001060          BNE    9$          ;BR - DETACH LINES CAN'T RECEIVE
36 006562 032763 000000G 000000G          BIT    #$TTGAG,LSW7(R3) ;IS LINE GAGGED?
37 006570 001414          BEQ    3$          ;BR IF NOT
38 006572 032763 000000G 000000G          BIT    #$INKMN,LSW4(R3) ;IS KMON RUNNING NOW?
39 006600 001010          BNE    3$          ;BR IF YES
40 006602 032737 000001 000000G          BIT    #1,EMTBLK ;DOES HE WANT TO OVERRIDE GAG?
41 006610 001004          BNE    3$          ;BR IF YES
42 006612 012700 000001          MOV    #1,RO      ;RETURN ERROR CODE 1 IF LINE IS GAGGED
43 006616 000137 000000G          JMP    SETERR
44         ;
45         ; Get a free message buffer.
46         ;
47 006622 004737 006730'          3$:  CALL   GTMSBF ;GET A FREE MESSAGE BUFFER
48 006626 103004          BCC    6$          ;CONTINUE IF NO ERROR
49 006630 012700 000002          MOV    #2,RO      ;RETURN ERROR CODE 2 IF NO MESSAGE BUFFERS
50 006634 000137 000000G          JMP    SETERR
51         ;
52         ; Move message from user's buffer to system message buffer.
53         ;
54 006640 013703 000004G          6$:  MOV    EMTBLK+4,R3 ;GET ADDRESS OF USER'S MESSAGE BUFFER
55 006644 010300          MOV    R3,RO      ;VALIDATE ADDRESS
56 006646 004737 000000G          CALL   VALADB
57 006652 010402          MOV    R4,R2      ;GET ADDRESS OF SYSTEM MESSAGE BUFFER BLOCK

```

```

58 006654 062702 000000G      ADD    #SB$TXT,R2      ; POINT TO MESSAGE STORAGE AREA
59 006660 010405              MOV    R4,R5
60 006662 062705 000000G      ADD    #SB$END,R5     ; POINT TO END OF MESSAGE AREA
61 006666 004737 000000G      1$:   CALL   GETUCH    ; GET CHAR FROM USER'S BUFFER
62 006672 110022              MOVB  RO,(R2)+        ; MOVE TO SYSTEM MESSAGE BUFFER
63 006674 001403              BEQ   2$              ; BR IF REACHED END
64 006676 020205              CMP   R2,R5          ; REACHED END OF BUFFER SPACE?
65 006700 103772              BLD  1$              ; BR IF NOT
66 006702 105042              CLRB  -(R2)          ; Store null at end of message
67                               ;
68                               ; Queue the message for the receiving job.
69                               ;
70 006704 013701 000002G      2$:   MOV    EMTBLK+2,R1 ; GET # OF RECEIVING JOB
71 006710 006301              ASL   R1              ; CONVERT TO JOB INDEX #
72 006712 004737 007052'      CALL  QMSG           ; QUEUE MESSAGE FOR JOB
73                               ;
74                               ; Finished
75                               ;
76 006716 000137 000000G      JMP   EMTXIT
77                               ;
78                               ; Invalid line #
79                               ;
80 006722 005000              9$:   CLR   RO          ; RETURN ERROR CODE 0
81 006724 000137 000000G      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 006730   GTMSBF: DISABL          ; ** DISABLE **
10 006736 005737 000000G          TST      NMUMB          ; ARE THERE ANY AVAILABLE MESSAGE BLOCKS?
11 006742 003004                   BGT      3$          ; BR IF SO
12 006744 032737 000002 000000G   BIT      #2,EMTBLK  ; DOES USER WANT TO WAIT?
13 006752 001035                   BNE      7$          ; BR IF NOT
14 006754 013704 000000G   3$:    MOV      SNMSHD,R4  ; GET ADDRESS OF 1ST FREE BLOCK
15 006760 001013                   BNE      1$          ; BR IF OK
16 006762 032737 000002 000000G   BIT      #2,EMTBLK  ; DOES USER WANT TO WAIT?
17 006770 001026                   BNE      7$          ; BR IF NOT
18          ;
19          ; No message buffers available.
20          ; Suspend job until one becomes available.
21          ;
22 006772 012700 000000G   2$:    MOV      #S$WSMB,R0  ; WAIT FOR MESSAGE BUFFER STATE.
23 006776 004737 000000G          CALL     QNSPND        ; WAIT FOR MESSAGE BUFFER ** ENABLE **
24 007002 004737 000000G          CALL     CHKABT        ; WERE WE ABORTED WHILE ASLEEP?
25 007006 000750                   BR       GTMSBF        ; TRY AGAIN
26          ;
27          ; Got a buffer.  Unlink from free chain.
28          ;
29 007010 005337 000000G   1$:    DEC      NMUMB          ; ONE LESS FREE BUFFER
30 007014 016437 000000G 000000G   MOV      SB$LNK(R4),SNMSHD; REMOVE FROM FREE LIST
31 007022                   ENABL          ; ** ENABLE **
32 007030 010400                   MOV      R4,R0        ; INITIALIZE SB$PNT
33 007032 062700 000000G          ADD      #SB$TXT,R0
34 007036 010064 000000G          MOV      R0,SB$PNT(R4)
35 007042 000241                   CLC          ; SAY NO ERRORS
36 007044 000401                   BR       9$
37          ;
38          ; No free message buffers, flag error return
39          ;
40 007046 000261   7$:    SEC
41          ;
42          ; Finished
43          ;
44 007050 000207   9$:    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 007052 010146 QMSG:  MOV      R1, -(SP)
10         ;
11         ; Add message buffer to queue for line
12         ;
13 007054         DISABL          ;** DISABLE **
14 007062 016101 000000G        MOV      LNPRIM(R1),R1 ;OBTAIN THE CURRENT PRIMARY LINE NUMBER
15 007066 016100 000000G        MOV      LMSGBF(R1),R0 ;GET ADDRESS OF HEAD OF MESSAGE LIST FOR JOB
16 007072 001411                BEQ      3$          ;BR IF NO MESSAGES QUEUED FOR JOB
17 007074 005760 000000G        1$:  TST      SB$LNK(R0)    ;IS THIS LAST ENTRY IN LIST?
18 007100 001403                BEQ      2$          ;BR IF YES
19 007102 016000 000000G        MOV      SB$LNK(R0),R0 ;CHAIN TO NEXT QUEUED MESSAGE BLOCK
20 007106 000772                BR       1$
21 007110 010460 000000G        2$:  MOV      R4, SB$LNK(R0) ;ADD OUR MESSAGE TO TAIL OF LIST
22 007114 000402                BR       4$
23 007116 010461 000000G        3$:  MOV      R4, LMSGBF(R1) ;ADD OUR MESSAGE BLOCK TO LIST FOR JOB
24 007122 005064 000000G        4$:  CLR      SB$LNK(R4) ;SAY WE ARE AT END OF LIST
25 007126                ENABL          ;** ENABLE **
26         ;
27         ; Initiate transmission to the line.
28         ;
29 007134 116101 000000G        MOVVB   LNMAP(R1),R1 ;GET VIRTUAL LINE #
30 007140 004777 000000G        CALL    @TRNSTR ;INITIATE TRANSMISSION
31         ;
32         ; Finished
33         ;
34 007144 012601                MOV      (SP)+, R1
35 007146 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 007150  ASTXIT:
6          ;
7          ; Set up information that will sidetrack control in EMTXIT.
8          ;
9 007150  013702  000000G      MOV     EMTCAD,R2      ;Get pointer to top entry on return stack
10 007154  020227  000000G     CMP     R2,#EMTCAS    ;Were we expecting a completion routine exit?
11 007160  001002             BNE     1$           ;Br if yes
12 007162  000137  000000G     JMP     BADEMT       ;Illegal EMT if nothing on cpl rtn return stk
13 007166  012237  000000G     1$:  MOV     (R2)+,EMTRAD ;Set return address for compl routine
14 007172  010237  000000G     MOV     R2,EMTCAD   ;Save new stack pointer
15 007176  000137  000000G     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 007202 105737 000000G  ESPCPS: TSTB   CURCP       ;ARE WE IN A COMPLETION ROUTINE NOW?
8 007206 001003          BNE     1$         ;BR IF YES
9 007210 005000          CLR     RO         ;RETURN ERROR CODE 0 IF NOT
10 007212 000137 000000G  JMP     SETERR
11 007216 005737 000000G  1$:   TST     SPCPS       ; IS ANOTHER .SPCPS PENDING NOW?
12 007222 001404          BEQ     2$         ;BR IF NOT
13 007224 012700 000001  MOV     #1,RO         ;RETURN ERROR CODE 1 IF YES
14 007230 000137 000000G  JMP     SETERR
15 007234 013700 000002G  2$:   MOV     EMTBLK+2,RO ;GET ADDRESS OF USER'S INFORMATION BLOCK
16 007240 004737 000000G  CALL   VALADW        ;VALIDATE THE ADDRESS
17 007244 010037 000000G  MOV     RO,SPCPS      ;SAVE ADDRESS OF INFO BLOCK
18 007250 000137 000000G  JMP     EMTXIT        ;EXIT -- SYSXIT WILL DO REAL PROCESSING

```

```
1                                     .SBTTL  EMT376 -- EMT 376 Processing
2                                     ;-----
3                                     ; Process EMT 376
4                                     ;
5                                     ; Inputs:
6                                     ; R4 = EMT function code.
7                                     ;
8 007254 120427 000376 EMT376: CMPB   R4,#376      ;Is this an EMT 376?
9 007260 001011      BNE    1$              ;Br if not
10 007262 013704 000000G      MOV    EMTADR,R4      ;Get address of EMT instruction
11 007266 106564 000002      MFPD  2(R4)          ;Get value that follows instruction
12 007272 012600      MOV    (SP)+,R0      ;Get code word
13 007274 000300      SWAB   R0              ;Get code byte to low-order
14 007276 120027 000373      CMPB   R0,#373      ;Overlay I/O error?
15 007302 001402      BEQ    2$              ;Br if yes
16 007304 000137 000000G      1$:   JMP    BADEMT      ;Invalid EMT if not
17                                     ;
18                                     ; Overlay I/O error
19                                     ;
20 007310 010437 000000G      2$:   MOV    R4,ABRTAD      ;Set address of abort
21 007314 012737 177750 000000G      MOV    #-30,ABRTCD   ;Set abort code
22 007322 004737 000000G      CALL  STOP          ;Abort the job
```

JBINFO -- Get information about a specific 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 007326 113705 000003G JBINFO: MOVB  EMTBLK+3,R5 ;GET SUB-FUNCTION NUMBER
16 007332 020527 000012      CMP    R5,#MXJIFN ;IS IT TOO BIG?
17 007336 101404          BLOS   1$ ;BR IF OK
18 007340 012700 000001      MOV    #1,R0 ;ERROR CODE 1 IF TOO BIG
19 007344 000137 000000G      JMP    SETERR
20         ;
21         ; Subfunction number is valid.
22         ; Check job number.
23         ;
24 007350 113701 000002G 1$: MOVB  EMTBLK+2,R1 ;GET SPECIFIED JOB NUMBER
25 007354 001404          BEQ    2$ ;ZERO IS INVALID
26 007356 006301          ASL    R1 ;CONVERT TO LINE INDEX NUMBER
27 007360 020127 000000G      CMP    R1,#LSTSL ;IS IT A VALID NUMBER?
28 007364 101404          BLOS   3$ ;BR IF YES
29 007366 012700 000002      2$: MOV    #2,R0 ;ERROR 2 IF INVALID LINE NUMBER
30 007372 000137 000000G      JMP    SETERR
31         ;
32         ; See if line is currently logged on
33         ;
34 007376 032761 000000G 000000G 3$: BIT    %%KINIT,LSW(R1) ;IS THE LINE LOGGED ON NOW?
35 007404 001003          BNE    4$ ;BR IF YES
36 007406 005000          CLR    R0 ;ERROR # 0 IF NOT
37 007410 000137 000000G      JMP    SETERR
38         ;
39         ; Check the address specified for the user's result buffer
40         ;
41 007414 013704 000004G 4$: MOV    EMTBLK+4,R4 ;GET USER'S VIRTUAL ADDRESS
42 007420 010400          MOV    R4,R0
43 007422 004737 000000G      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 007426 006305          ASL    R5 ;CONVERT SUB-FUNCTION # TO WORD INDEX
53 007430 000175 010014'      JMP    @JIJMPX(R5) ;ENTER PROCESSING ROUTINE
54         ;
55         ; #0 -- Get job status flags
56         ;
57 007434 005000          JISTAT: CLR  R0 ;DEVELOP FLAGS IN R0

```

```

58          ; See if job is detached or virtual
59 007436 020127 000000G          CMP      R1,#LSTPL          ; PRIMARY LINE?
60 007442 101410          BLOS     1$                ; BR IF YES
61 007444 020127 000000G          CMP      R1,#LSTD L          ; DETACHED LINE?
62 007450 101403          BLOS     2$                ; BR IF YES
63 007452 052700 000000G          BIS      #JIVLN,RO          ; SET VIRTUAL-LINE FLAG
64 007456 000402          BR       1$
65 007460 052700 000000G          2$:     BIS      #JIDLN,RO          ; SET DETACHED-LINE FLAG
66          ; See if job is locked in memory
67 007464 032761 000000G 000000G 1$:    BIT      #MLOCK,LSW6(R1); IS JOB LOCKED IN MEMORY?
68 007472 001402          BEQ      4$                ; BR IF NOT
69 007474 052700 000000G          BIS      #JIMLOK,RO          ; SET MEMORY-LOCKED FLAG
70          ; See if job has operator privilege
71 007500 010046          4$:     MOV      RO,-(SP)          ; SAVE OUR STATUS CODES
72 007502          OCALL   GETCXT          ; GET EXCLUSIVE ACCESS TO CONTEXT BUFFER
73 007510 103002          BCC     6$                ; BR IF GOT IT
74 007512 004737 000000G          CALL    STOP              ; JOB WAS ABORTED WHILE WAITING
75 007516 010102          6$:     MOV      R1,R2          ; GET INDEX # OF JOB WE ARE COPYING FROM
76 007520 012703 000000G          MOV     #PRIVCO,R3        ; ADDR OF INFO TO GET
77 007524 012700 000002          MOV     #2,RO             ; GET # BYTES TO COPY
78 007530          OCALL   REDCXT          ; GET INFO FROM CONTEXT BLOCK OF OTHER JOB
79 007536 032777 000000G 000000G          BIT     #PO$SYS,@CXTBUF    ; IS THIS A PRIVILEGED JOB?
80 007544 001402          BEQ     5$                ; BR IF NOT
81 007546 052716 000000G          BIS     #JIPRIV,(SP)       ; SET PRIVILEGE FLAG
82 007552          5$:     OCALL   FRECXT          ; FREE THE CONTEXT BUFFER
83 007560 012600          MOV     (SP)+,RO          ; GET STATUS FLAGS
84 007562 000422          BR     JIRET1            ; GO RETURN VALUE
85          ;
86          ; #1 -- Get job's run state
87          ;
88 007564 012700 010042'          JIRUN:  MOV     #JIRNST,RO    ; Point to state table
89 007570 116102 000000G          MOVVB  LSTATE(R1),R2      ; Get job's current execution state
90 007574 120220          1$:     CMPB   R2,(RO)+      ; Search for state in table
91 007576 001406          BEQ     2$                ; Br if found it
92 007600 005200          INC     RO                ; Skip over value
93 007602 020027 010126'          CMP     RO,#JIRNND        ; Checked all entries?
94 007606 103772          BLD     1$                ; Br if yes
95 007610 005000          CLR     RO                ; Don't recognize state, return 0
96 007612 000406          BR     JIRET1            ; Return the value
97 007614 111000          2$:     MOVVB  (RO),RO       ; Get value to return
98 007616 000404          BR     JIRET1            ; Return the value
99          ;
100         ; #2 -- Get amount of memory in use by job
101         ;
102 007620 016100 000000G          JIMEM:  MOV     LNBLKS(R1),RO ; GET # 256-WORD BLOCKS USED BY JOB
103 007624 066100 000000G          ADD     LNSBLK(R1),RO      ; ADD # BLOCKS USED BY PLAS REGIONS
104 007630 010046          JIRET1: MOV     RO,-(SP)     ; STORE VALUE INTO USER'S BUFFER
105 007632 106614          MTPD   (R4)
106 007634 000137 000000G          JMP     EMTXIT            ; FINISHED WITH EMT
107         ;
108         ; #3 -- Get job connect time (minutes)
109         ;
110 007640 013700 000000G          JICONT: MOV     MINTIM,RO    ; GET CURRENT SYSTEM UP-TIME
111 007644 166100 000000G          SUB     LCONTM(R1),RO      ; GET # MINUTES JOB HAS BEEN CONNECTED
112 007650 000767          BR     JIRET1
113         ;
114         ; #4 -- Get base 256-word block # assigned to job area

```

JBINFO -- Get information about a specific job

```

115
116 007652 016100 000000G JIBASE: MOV LBASE(R1),R0 ;GET BASE BLOCK #
117 007656 000764 BR JIRET1
118
119 ; #5 -- Get name of program being run by job
120 ;
121 007660 016102 000000G JIPROG: MOV LPRG1(R1),R2 ;GET FIRST 3 CHARS OF NAME
122 007664 016103 000000G MOV LPRG2(R1),R3 ;GET SECOND 3 CHARS OF NAME
123 007670 000411 BR JIRET2
124
125 ; #6 -- Get project/programmer number
126 ;
127 007672 016102 000000G JIPPN: MOV LPROJ(R1),R2 ;GET PROJECT NUMBER
128 007676 016103 000000G MOV LPROG(R1),R3 ;GET PROGRAMMER NUMBER
129 007702 000404 BR JIRET2
130
131 ; #7 -- Get CPU time used by job
132 ;
133 007704 016102 000000G JICPU: MOV LCPUHI(R1),R2 ;GET HIGH-ORDER CPU TIME
134 007710 016103 000000G MOV LCPULO(R1),R3 ;GET LOW-ORDER CPU TIME
135 007714 010246 JIRET2: MOV R2,-(SP) ;RETURN FIRST WORD OF RESULT
136 007716 106624 MTPD (R4)+
137 007720 010346 MOV R3,-(SP) ;RETURN SECOND WORD OF RESULT
138 007722 106624 MTPD (R4)+
139 007724 000137 000000G JMP EMTXIT ;FINISHED WITH EMT
140
141 ; #8 -- Job execution priority
142 ;
143 007730 116100 000000G JIPRID: MOVB LPRI(R1),R0 ;Get current execution priority for job
144 007734 000735 BR JIRET1 ;Return
145
146 ; #9 -- Job name
147 ;
148 007736 010105 JINAME: MOV R1,R5 ;Copy job index number
149 007740 012702 000000G MOV #6,R2 ;6 words per name
150 007744 070502 MUL R2,R5 ;Name should be 12 bytes long
151 007746 062705 000000G ADD #LUNAME,R5 ;Convert to name table index
152 007752 012546 1$: MOV (R5)+,-(SP) ;Fetch next two bytes of name
153 007754 106624 MTPD (R4)+ ;Move to user's buffer
154 007756 077203 SOB R2,1$ ;Loop through whole name
155 007760 000137 000000G JMP EMTXIT ;Finished with EMT
156
157 ; #10 -- Parent and primary job numbers
158 ;
159 007764 016102 000000G JIPAPR: MOV LNPRIM(R1),R2 ;Get index number of primary line
160 007770 006202 ASR R2 ;Convert to job number
161 007772 016103 000000G MOV LPARNT(R1),R3 ;Get parent job index number
162 007776 006203 ASR R3 ;Convert to job number
163 010000 010246 MOV R2,-(SP) ;Return primary job number
164 010002 106624 MTPD (R4)+
165 010004 010346 MOV R3,-(SP) ;Return parent job number
166 010006 106624 MTPD (R4)+
167 010010 000137 000000G JMP EMTXIT ;FINISHED WITH EMT
168
169 ; Branch table used to enter job information routines
170 ;
171 010014 007434' JIJMPX: .WORD JISTAT ; 0 -- GET JOB STATE FLAGS

```

JBINFO -- Get information about a specific job

172	010016	007564'	.WORD	JIRUN	; 1	-- GET JOB RUN STATE CODE
173	010020	007620'	.WORD	JIMEM	; 2	-- GET AMT OF MEMORY USED BY JOB
174	010022	007640'	.WORD	JICONT	; 3	-- GET CONNECT TIME FOR JOB
175	010024	007652'	.WORD	JIBASE	; 4	-- GET BASE ADDRESS FOR JOB
176	010026	007660'	.WORD	JIPROG	; 5	-- GET NAME OF PROGRAM BEING RUN
177	010030	007672'	.WORD	JIPPN	; 6	-- GET PROJECT/PROGRAMMER #
178	010032	007704'	.WORD	JICPU	; 7	-- GET CPU TIME USED BY JOB
179	010034	007730'	.WORD	JIPRIO	; 8	-- GET JOB EXECUTION PRIORITY
180	010036	007736'	.WORD	JINAME	; 9	-- GET JOB NAME
181	010040	007764'	.WORD	JIPAPR	; 10	-- GET PARENT AND PRIMARY JOB NUMBERS
182		000012	MXJIFN =	<< -JIJMPX>/2>-1		

JBINFO -- Get information about a specific job

1					
2					; Table used to convert TSX scheduler state codes into
3					; externally defined run states.
4					
5	010042	000G	001	JIRNST: .BYTE	S\$TWFN,1 ;Timed wait completion
6	010044	000G	001	.BYTE	S\$OTLO,1 ;Terminal output buffer almost empty
7	010046	000G	001	.BYTE	S\$IOfN,1 ;I/O completion
8	010050	000G	002	.BYTE	S\$CPU,2 ;Normal priority execution
9	010052	000G	003	.BYTE	S\$LOW,3 ;Fixed low priority execution
10	010054	000G	004	.BYTE	S\$INWT,4 ;Waiting for terminal input
11	010056	000G	005	.BYTE	S\$OTWT,5 ;Waiting for terminal output buffer space
12	010060	000G	006	.BYTE	S\$TMWT,6 ;Waiting for timed interval
13	010062	000G	007	.BYTE	S\$SPND,7 ;Doing .SPND
14	010064	000G	010	.BYTE	S\$SFWT,8 ;Shared file wait
15	010066	000G	011	.BYTE	S\$MSWT,9 ;Waiting for message
16	010070	000G	012	.BYTE	S\$QUSR,10 ;Waiting for USR for file operation
17	010072	000G	013	.BYTE	S\$IOWT,11 ;Waiting for I/O operation
18	010074	000G	013	.BYTE	S\$NEDQ,11 ;Waiting for I/O queue element
19	010076	000G	013	.BYTE	S\$QMIO,11 ;Waiting for mapped I/O buffer
20	010100	000G	013	.BYTE	S\$QCCB,11 ;Waiting for cache control block
21	010102	000G	013	.BYTE	S\$QSPD,11 ;Waiting for special device data base
22	010104	000G	013	.BYTE	S\$WSMB,11 ;Waiting for system message buffer
23	010106	000G	014	.BYTE	S\$SPDB,12 ;Waiting for spooled device control block
24	010110	000G	014	.BYTE	S\$SPCB,12 ;Waiting for spooled device control block
25	010112	000G	015	.BYTE	S\$TTSC,13 ;Single char terminal input complete
26	010114	000G	015	.BYTE	S\$TTFN,13 ;Non single char terminal input complete
27	010116	000G	015	.BYTE	S\$OTFN,13 ;Terminal buffer empty
28	010120	000G	015	.BYTE	S\$HICP,13 ;Interactive job execution
29	010122	000G	016	.BYTE	S\$RT,14 ;Fixed high priority
30	010124	000G	017	.BYTE	S\$WFM,15 ;Waiting for memory expansion
31	010126			JIRNND:	;End of table
32				.EVEN	

```

1          .SBTTL  Get terminal type code
2          ;-----
3          ; This EMT returns in R0 a numeric value that indicates the type of
4          ; terminal from which the job is being run.
5          ; The terminal type is declared in the line definition in TSGEN or
6          ; by the SET TERMINAL keyboard command.
7          ;
8 010126 012702 010212' TRMEMT: MOV      #LOTRM,R2      ;POINT TO START OF TERMINAL TYPE TABLE
9 010132 016104 000000G      MOV      LTRMTP(R1),R4    ;GET TERMINAL TYPE FLAGS FOR OUR JOB
10 010136 116100 000000G      MOVVB   LNPRIM(R1),R0    ;Get out primary line number
11 010142 032760 000000G 000000G      BIT      #$V52EM,LSW11(R0);Are we emulating a VT52?
12 010150 001402              BEQ      1$              ;Br if not
13 010152 012704 000000G      MOV      #VT52,R4      ;Say terminal is a VT52
14 010156 020422 1$:          CMP      R4,(R2)+      ;SEARCH FOR MATCHING PATTERN IN TABLE
15 010160 001405              BEQ      2$              ;BR IF FOUND
16 010162 020227 010236'      CMP      R2,#HITRM     ;HAVE WE SEARCHED ALL OF TABLE?
17 010166 101773              BLOS    1$              ;BR IF NOT
18 010170 005002              CLR      R2              ;SAY UNKNOWN TERMINAL TYPE
19 010172 000403              BR      3$              ;
20 010174 162702 010212'      2$:      SUB      #LOTRM,R2    ;CONVERT TABLE OFFSET TO TERMINAL CODE NUMBER
21 010200 006202              ASR      R2              ;
22 010202 010237 000000G      3$:      MOV      R2,URO      ;RETURN TERMINAL TYPE CODE # IN USER'S R0
23 010206 000137 000000G      JMP      EMTXIT        ;FINISHED
24          ;
25          ; Table to translate terminal type flags into code number.
26          ;
27 010212 000000G      LOTRM:  .WORD   VT52      ; 1 - VT52
28 010214 000000G      .WORD   VT100     ; 2 - VT100
29 010216 000000G      .WORD   HAZEL     ; 3 - Hazeltine
30 010220 000000G      .WORD   ADM3A     ; 4 - ADM3A
31 010222 000000G      .WORD   LA36      ; 5 - LA36
32 010224 000000G      .WORD   LA120     ; 6 - LA120
33 010226 000000G      .WORD   DIABLO    ; 7 - Diablo
34 010230 000000G      .WORD   QUME      ; 8 - Qume
35 010232 000000G      .WORD   VT2007    ; 9 - VT200 with 7 bit control codes
36 010234 000000G      .WORD   VT2008    ; 10 - VT200 with 8 bit control codes
37 010236              HITRM:
38          ;
39          .SBTTL  Get Project-Programmer number
40          ;-----
41          ; This emt is used to obtain the current Project-Programmer number
42          ; that a job is running under.
43          ;
44 010236 013700 000002G      GETPPN: MOV      EMTBLK+2,R0    ;GET ADDRESS OF CELL TO RECEIVE PPN
45 010242 004737 000000G      CALL   VALADW      ;VALIDATE THE ADDRESS
46 010246 016146 000000G      MOV      LPROJ(R1),-(SP) ;MOVE PROJECT # TO USER'S BUFFER
47 010252 106620              MTPD   (R0)+
48 010254 016146 000000G      MOV      LPROG(R1),-(SP)
49 010260 106620              MTPD   (R0)+      ;MOVE PROGRAMMER # TO USER'S BUFFER
50 010262 000137 000000G      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 010266 105737 000000G XGTLN: TSTB   EMTBLK   ;WANT CURRENT LINE OR RELATIVE
9 010272 001440          BEQ     1$      ;BR IF CURRENT
10         ;
11         ; THIS IS A REQUEST TO RETURN THE REAL LINE NUMBER OF A SUBPROCESS
12         ;
13 010274 020127 000000G          CMP     R1,#LSTPL   ;ARE WE A PRIMARY LINE?
14 010300 101406          BLOS    4$      ;BR IF OK
15 010302 020127 000000G          CMP     R1,#LSTD   ;ARE WE A VIRTUAL LINE?
16 010306 101003          BHI     4$      ;BR IF OK
17 010310 012700 000002          MOV     #2,R0      ;DETACHED JOB'S DON'T HAVE SUBPROCESSES
18 010314 000425          BR      6$      ;GO COMPLAIN ABOUT IT
19 010316 016101 000000G 4$:   MOV     LNPRIM(R1),R1 ;GET OUR PARENT INDEX
20 010322 013705 000002G          MOV     EMTBLK+2,R5 ;WANTS LINE NUMBER FOR THIS SUBPROCESS
21 010326 001002          BNE    2$      ;BR IF NOT ASKING FOR PARENT
22 010330 010105          MOV     R1,R5      ;COPY PARENT'S PROCESS INDEX
23 010332 000411          BR      3$      ;DON'T NEED SUBPROCESS MAPPING
24 010334 020527 000000G 2$:   CMP     R5,#MAXSEC ;VALID SUBPROCESS NUMBER?
25 010340 101403          BLOS    5$      ;BR IF OK
26 010342 012700 000001          MOV     #1,R0      ;SIGNAL INVALID SUBPROCESS #
27 010346 000410          BR      6$      ;AND GO REPORT ERROR
28 010350 066105 000000G 5$:   ADD     LSECT(R1),R5 ;POINT TO PRIMARY LINE'S SUBPROCESS TABLE
29 010354 114505          MOVB   -(R5),R5   ;GET SPECIFIED LINE'S INDEX (^1 IS OFFSET 0)
30 010356 006205 000000G 3$:   ASR     R5      ;CONVERT INDEX TO LINE #
31 010360 010537 000000G          MOV     R5,URO    ;PASS TO USER
32 010364 001034          BNE    XJNOP     ;IF IN USE, RETURN LINE # TO USER
33 010366 005000          CLR     R0      ;ELSE REPORT LINE NOT ACTIVE
34 010370 000137 000000G 6$:   JMP     SETERR   ;RETURN WITH ERROR
35         ;
36         ; THIS IS A REQUEST TO GET OUR CURRENT LINE NUMBER
37         ;
38 010374 006201 000000G 1$:   ASR     R1      ;CONVERT TO LINE #
39 010376 010137 000000G          MOV     R1,URO    ;RETURN TO USER IN R0
40 010402 000425          BR      XJNOP
41         ;
42         ;-----
43         ; THE .SPLSP EMT RETURNS IN R0 THE NUMBER OF FREE BLOCKS OR
44         ; NUMBER OF BLOCKS IN USE IN THE SPOOL FILE.
45         ;
46 010404 105737 000000G XSPLSP: TSTB   EMTBLK   ;NUMBER OF BLOCKS FREE OF IN USE?
47 010410 001417          BEQ     1$      ;BR IF NUMBER OF FREE BLOCKS
48 010412 113701 000000G          MOVB   NSPLDV,R1 ;GET # OF SPOOLED DEVICES
49 010416 001410          BEQ     2$      ;BR IF NONE
50 010420 013700 000000G          MOV     NSPLBL,R0 ;GET TOTAL BLOCKS IN SPOOL FILE
51 010424 163700 000000G          SUB     NFRESB,R0  ;SUBTRACT THE NUMBER OF FREE BLOCKS
52 010430 070127 000000G          MUL     #PVSPBL,R1 ;MULTIPLY BY # OF BLOCKS FOR EACH SPOOLED DEV
53 010434 160100          SUB     R1,R0      ;AND SUBTRACT FROM BLOCKS IN USE
54 010436 000401          BR      3$      ;
55 010440 005000 000000G 2$:   CLR     R0      ;RETURN ZERO IF NO SPOOLER
56 010442 010037 000000G 3$:   MOV     R0,URO    ;RETURN # OF BLOCKS IN USE TO USER IN R0
57 010446 000403          BR      XJNOP     ;RETURN

```

```

58 010450 013737 000000G 000000G 1$:      MOV      NFRESB,URO      ;RETURN # OF FREE BLOCKS TO USER IN RO
59 010456 000137 000000G          XJNOP:  JMP      EMTXIT
60
61 ;-----
62 ;   SET/RESET ODT CHARACTER ACTIVATION MODE
63 ;
64 010462 105737 000000G          XODTMD: TSTB     EMTBLK      ;SET OR RESET MODE?
65 010466 001407          BEQ      1$          ;BR IF RESET
66 010470 052761 000000G 000000G      BIS      ##ODTMD,LSW4(R1);TURN ON ODT MODE
67 010476 052761 000000G 000000G      BIS      ##DBGMD,LSW6(R1);SAY WE ARE IN DEBUGGER TERMINAL CONTROL MODE
68 010504 000764          BR       XJNOP
69 010506 042761 000000G 000000G 1$:      BIC      ##ODTMD,LSW4(R1);TURN OFF ODT MODE
70 010514 042761 000000G 000000G      BIC      ##DBGMD,LSW6(R1);RESET DEBUGGER TERMINAL MODE
71 010522 000755          BR       XJNOP
72 ;
73 ;-----
74 ;   Check to see if any input activation characters are pending.
75 ;   Return with c-flag set if not.
76 ;
77 010524 005761 000000G          CKINAC: TST      LACTIV(R1)  ;ANY PENDING ACTIVATION CHARS?
78 010530 001002          BNE      1$          ;BR IF YES
79 010532 000137 000000G          JMP      SETC          ;RETURN WITH C-FLAG SET
80 010536 000137 000000G 1$:      JMP      EMTXIT      ;RETURN WITH C-FLAG CLEAR
81 ;
82 ;-----
83 ;   Check to see if we can get site information (license # or site name)
84 ;   These are implemented in TSSPL2 which is only loaded if the
85 ;   spooling system is enabled.
86 ;
87 010542 105737 000000G          SITINF: TSTB     NSPLDV      ;Was spooler loaded?
88 010546 001002          BNE      1$          ;Br if so
89 010550 000137 000000G          JMP      SETC          ;If not, report error immediately
90 010554 000137 000000G 1$:      JMP      GTLICN     ;Jump to spool2 to execute

```

```
1 ;-----  
2 ; Set interactive or non-interactive mode for current program.  
3 ;  
4 010560 005737 000002G INTEM: TST EMTBLK+2 ;Set or reset interactive mode?  
5 010564 001404 BEQ 1$ ;Reset interactive mode  
6 ;  
7 ; Set interactive mode  
8 ;  
9 010566 042761 000000G 000000G BIC ##NOINT,LSW7(R1);Clear non-interactive mode flag  
10 010574 000405 BR 9$  
11 ;  
12 ; Reset interactive mode  
13 ;  
14 010576 052761 000000G 000000G 1$: BIS ##NOINT,LSW7(R1);Run job non-interactively  
15 010604 005061 000000G CLR LITIME(R1) ;Clear interactive timer  
16 ;  
17 ; Finished  
18 ;  
19 010610 000137 000000G 9$: JMP EMTXIT ;Finished
```

```

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 010614   113705   000000G  SYFEMT: MOVB    EMTBLK,R5      ;GET SYSTEM FILE INDEX
10 010620   020527   000004   CMP      R5,#MXSYFN      ;VALID FILE INDEX NUMBER?
11 010624   101403   .        BLOS     1$      ;BRANCH IF OK
12 010626   005000   .        CLR     R0      ;ERROR CODE 0 IF BAD INDEX
13 010630   000137   000000G  .        JMP     SETERR
14          ;
15          ; Convert system file index number to table offset
16          ;
17 010634   006305   .        1$:     ASL     R5      ;CONVERT INDEX TO TABLE OFFSET
18 010636   016504   010672'  .        MOV     SYFTBL(R5),R4 ;GET POINTER TO FILE NAME
19          ;
20          ; Get and validate user's return buffer
21          ;
22 010642   013705   000002G  .        MOV     EMTBLK+2,R5   ;GET USER'S VIRTUAL ADDRESS
23 010646   010500   .        .        MOV     R5,R0
24 010650   004737   000000G  .        .        CALL    VALADW   ;MAKE SURE ADDRESS IS OK
25          ;
26          ; Now return 4 RAD50 words of dev:filspc.ext to user buffer
27          ;
28 010654   012700   000004   .        MOV     #4,R0      ;MOVE 4 WORDS
29 010660   012446   .        2$:     MOV     (R4)+,-(SP) ;GET NEXT RAD50 WORD
30 010662   106625   .        .        MTPD    (R5)+   ;MOVE TO USER BUFFER
31 010664   077003   .        .        SOB     R0,2$
32 010666   000137   000000G  .        .        JMP     EMTXIT   ;FINISHED
33          ;
34          ; System file specification pointers
35          ;
36 010672   000000G  SYFTBL: .WORD   SWDBLK      ; 0 SWAP FILE
37 010674   000000G  .WORD   SPLBLK      ; 1 SPOOL FILE
38 010676   000000G  .WORD   RSFBLK      ; 2 PLAS REGION SWAP FILE
39 010700   000000G  .WORD   UCLDAT      ; 3 USER DEFINED COMMAND (UCL) FILE
40 010702   000000G  .WORD   INDFIL      ; 4 IND TEMP FILE
41          000004   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 010704  EMTSPD:
12         ;
13         ; Get line number and see if it is valid
14         ;
15 010704  013702  000002G          MOV     EMTBLK+2,R2      ;Get line number
16 010710  006302          ASL     R2              ;Convert to line index number
17 010712  001001          BNE    1$              ;Br if line number specified
18 010714  010102          MOV    R1,R2          ;Use current line
19 010716  020227  000000G      1$:    CMP    R2,#LSTSL    ;Primary or virtual line?
20 010722  101002          BHI    10$            ;Br if not
21 010724  016202  000000G      MOV    LNPRIM(R2),R2   ;Map virtual line # to real line #
22 010730  020227  000000G      10$:   CMP    R2,#LSTHL    ;Is this a valid line number?
23 010734  101010          BHI    5$              ;Br if not
24 010736  032762  000000G  000000G  BIT    ##HARD,LSW3(R2) ;Line connected to hardware?
25 010744  001404          BEQ    5$              ;Br if not
26 010746  032762  000000G  000000G  BIT    ##DEAD,LSW3(R2) ;Is line installed?
27 010754  001404          BEQ    4$              ;Br if yes
28 010756  012700  000002      5$:    MOV    #2,RO        ;Return error code 2 for invalid line #
29 010762  000137  000000G      JMP    SETERR
30         ;
31         ; See if we are privileged to perform this operation
32         ;
33 010766  032737  000000G  000000G  4$:    BIT    #P2$TRM,PRIVC2 ;Do we have privilege to change speed?
34 010774  001011          BNE    11$            ;Br if yes
35 010776  105737  000000G      TSTB   EMTBLK        ;EMT to reset XOFF status or line speed EMT?
36 011002  001404          BEQ    12$            ;Br if changing line speed
37 011004  020201          CMP    R2,R1          ;Resetting XOFF for our own line?
38 011006  001404          BEQ    11$            ;Br if yes
39 011010  012700  000001      12$:   MOV    #1,RO        ;Return error code 1
40 011014  000137  000000G      JMP    SETERR
41         ;
42         ; See if this is actually the EMT to reset terminal XOFF status
43         ;
44 011020  010201          11$:   MOV    R2,R1          ;Get line index number to R1
45 011022  123727  000000G  000001  CMPB   EMTBLK,#1     ;Set line speed, reset XOFF or DTR?
46 011030  002416          BLT    2$              ;Br if want to set speed
47 011032  001422          BEQ    20$            ;Br if want to reset XOFF
48         ;
49         ; Want to raise or lower DTR, get current data set status
50         ;
51 011034  004737  000000G      CALL   GETDSS         ;Get data set status into RO
52 011040  042700  000000G      BIC    #MS$DTR,RO    ;Assume we want to lower DTR
53 011044  123727  000000G  000002  CMPB   EMTBLK,#2     ;Should we raise DTR?
54 011052  001002          BNE    13$            ;Br if not
55 011054  052700  000000G      BIS    #MS$DTR,RO    ;Set DTR up bit
56 011060  004737  000000G      13$:   CALL   SETDSS         ;Reset data set status
57 011064  000413          BR     30$            ;And exit

```

```
58 ;  
59 ; Call hardware-dependent routine to set the line speed  
60 ;  
61 011066 013700 0000040 2$: MOV EMTBLK+4,R0 ;Get speed code  
62 011072 004737 0000000 CALL SETSPD ;Set the line speed  
63 011076 000406 BR 30$ ;And exit  
64 ;  
65 ; Reset XOFF status and start transmitter  
66 ;  
67 011100 042761 0000000 0000000 20$: BIC #CTRLS.LSW3(R1);Reset XOFF flag for line  
68 011106 004777 0000000 CALL @TRNSTR ;Try to start output to the line  
69 011112 000400 BR 30$ ;Finished  
70 ;  
71 ; Finished  
72 ;  
73 011114 000137 0000000 30$: JMP EMTXIT ;Finished
```

Set transmit/receive speed for a line

```

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 011120 012702 000400 XGTTAB: MOV #400,R2 ;STORE VALUES HERE
9 011124 012703 000372 MOV #372,R3 ;STORE VECTOR HERE
10 ; TSLICH -- TSX command lead-in character.
11 011130 113746 000000G MOVB VTSLCH,-(SP) ;GET LEAD-IN CHARACTER
12 011134 106623 MTPD (R3)+ ;MOVE TO JOB SPACE
13 ; Name of running program.
14 011136 010246 MOV R2,-(SP) ;SET POINTER TO PROGRAM NAME
15 011140 106623 MTPD (R3)+ ;IN VECTOR
16 011142 012700 000004 MOV #4,R0 ;GET # WORDS TO MOVE
17 011146 012704 000000G MOV #RUNDEV,R4 ;POINT TO NAME OF RUNNING PROGRAM
18 011152 012446 1$: MOV (R4)+,-(SP) ;MOVE NAME TO JOB SPACE
19 011154 106622 MTPD (R2)+
20 011156 077003 SOB R0,1$
21 ;
22 ; Tell user that vector is located at 350.
23 ;
24 011160 012737 000350 000000G MOV #350,UR0 ;RETURN VECTOR ADDRESS IN USER'S R0
25 011166 000137 000000G 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 011172  113705  000000G  TBLEMT: MOV  B   EMTBLK,R5      ;GET SUB-FUNCTION CODE
15 011176  001412                BEQ   1$          ;BR IF IT IS GET-VALUE
16         ;
17         ; Operator privilege is required to modify a table.
18         ;
19 011200  032737  000000G 000000G      BIT   #PO$SYS,PRIVCO ;IS THIS USER PRIVILEGED?
20 011206  001004                BNE   2$          ;BR IF YES
21 011210  012700  000001      MOV   #1,R0        ;RETURN ERROR CODE OF 1 IF NOT
22 011214  000137  000000G      JMP   SETERR
23 011220  013703  000004G  2$:  MOV   EMTBLK+4,R3 ;GET VALUE FOR STORE
24 011224  013702  000002G  1$:  MOV   EMTBLK+2,R2 ;GET TABLE INDEX
25         ;
26         ; If subfunction code = 4 then branch off to special processing routine
27         ;
28 011230  020527  000004      CMP   R5,#4       ;IS THIS A SPECIAL TABLE OPERATION?
29 011234  001006                BNE   11$         ;BR IF NOT
30 011236  006302                ASL   R2          ;GET WORD TABLE INDEX FOR OPERATION
31 011240  020227  000000      CMP   R2,#MXSPTF ;MAKE SURE RANGE IS OK
32 011244  101043                BHI   9$          ;BR IF INVALID
33 011246  000172  011444'      JMP   @SPLTF(R2) ;ENTER PROCESSING ROUTINE
34         ;
35         ; This is not a special table function
36         ;
37 011252  006302  11$:  ASL   R2          ;CONVERT TO WORD TABLE VALUE
38 011254  002007                BGE   4$          ;BR IF POS TABLE
39         ;
40         ; Negative table index
41         ;
42 011256  005402                NEG   R2          ;MAKE TABLE INDEX POSITIVE
43 011260  020227  000006      CMP   R2,#MXNGTX ;IS VALUE TOO BIG?
44 011264  103033                BHIS  9$          ;BR IF TOO BIG
45 011266  016204  011472'      MOV   NEGTAB(R2),R4 ;GET TABLE ADDRESS
46 011272  000406                BR   3$
47         ;
48         ; Positive table index
49         ;
50 011274  020227  000024  4$:  CMP   R2,#MXPSTX ;IS TABLE INDEX TOO BIG?
51 011300  103025                BHIS  9$          ;BR IF TOO BIG
52 011302  016204  011446'      MOV   POSTAB(R2),R4 ;GET TABLE ADDRESS
53 011306  060104                ADD  R1,R4        ;INDEX BY LINE NUMBER
54         ;
55         ; Perform requested operation on table
56         ;
57 011310  120527  000001  3$:  CMPB  R5,#1       ;CHECK SUB-FUNCTION CODE

```

```

58 011314 003006          BGT      5$          ;BR IF BIT SET OR RESET
59 011316 001403          BEQ      6$          ;BR IF STORE VALUE
60                          ; Fetch table value
61 011320 011437 000000G   MOV      @R4,URO      ;RETURN TABLE VALUE IN USER'S RO
62 011324 000411          BR       10$
63                          ; Store value into table.
64 011326 010314 6$:    MOV      R3,@R4      ;STORE INTO TABLE
65 011330 000407          BR       10$
66 011332 020527 000002   5$:    CMP      R5,#2      ;BIT SET OR RESET?
67 011336 001002          BNE      7$          ;BR IF BIT CLEAR
68                          ; Set bits in table
69 011340 050314          BIS      R3,@R4      ;SET BITS
70 011342 000402          BR       10$
71                          ; Clear bits in table
72 011344 040314 7$:    BIC      R3,@R4      ;CLEAR BITS IN TABLE
73 011346 000400          BR       10$
74                          ;
75                          ; Finished
76                          ;
77 011350 000137 000000G 10$:    JMP      EMTXIT
78                          ;
79                          ; Invalid table index
80                          ;
81 011354 005000 9$:    CLR      R0          ;RETURN ERROR CODE 0
82 011356 000137 000000G   JMP      SETERR
83                          ;
84                          ; Special table function routines
85                          ;
86                          ; 1 -- Set maximum execution priority for job
87                          ;
88 011362 005703 SETMXP: TST      R3          ;PRIORITY MUST BE > 0
89 011364 003403          BLE      1$          ;BR IF INVALID
90 011366 020327 000000G   CMP      R3,#MAXPRI   ;MAY NOT EXCEED THIS
91 011372 101402          BLOS    2$          ;BR IF OK
92 011374 113703 1$:    MOVB    VPRIDF,R3      ;IF BAD, SET TO NORMAL PRIORITY
93 011400 110337 000000G 2$:    MOVB    R3,MXJPRI    ;SET MAX PRIORITY FOR JOB
94 011404 120361 000000G   CMPB    R3,LBSPRI(R1) ;COMPARE WITH CURRENT JOB PRIORITY
95 011410 103013          BHIS    3$          ;BR IF CURRENT PRIORITY IS OK
96 011412 110361 000000G   MOVB    R3,LBSPRI(R1) ;SET BASE PRIORITY FOR JOB
97 011416 110361 000000G   MOVB    R3,LPRI(R1)   ;SET CURRENT PRIORITY FOR JOB
98 011422 032761 000000G 000000G BIT    ##VNOTT,LSW(R1) ;Is this a subprocess disconnected from term?
99 011430 001403          BEQ      3$          ;Br if not
100 011432          OCALL    SPPRED      ;Reduce prio of disconnected subprocess
101 011440 000137 000000G 3$:    JMP      EMTXIT      ;FINISHED
102                          ;
103                          ; Jump table for special table functions (subfunction code = 4)
104                          ;
105 011444 011362' SPLTF: .WORD    SETMXP      ; 0 -- Set maximum job priority
106          000000 MXSPTF =    <.-SPLTF>-2
107                          ;
108                          ; Table of table addresses for positive offsets.
109                          ;
110 011446 000000G POSTAB: .WORD    LCONTM      ;+00 -- CONNECT TIME
111 011450 000000G          .WORD    LSW          ;+01 -- LSW
112 011452 000000G          .WORD    LSW2         ;+02 -- LSW2
113 011454 000000G          .WORD    LSW3         ;+03 -- LSW3
114 011456 000000G          .WORD    LSW4         ;+04 -- LSW4

```

```
115 011460 000000G      .WORD   LSW5           ;+05 -- LSW5
116 011462 000000G      .WORD   LSW9           ;+06 -- LSW9
117 011464 000000G      .WORD   LPROJ          ;+07 -- LPROJ
118 011466 000000G      .WORD   LPROG          ;+10 -- LPROG
119 011470 000000G      .WORD   LSW2S          ;+11 -- LSW2S
120                000024  MXPSTX =      .-POSTAB
121                ;
122                ; Table of addresses for negative offsets.
123                ;
124 011472 000000      NEGTAB: .WORD   0           ; -00
125 011474 000000G      .WORD   UPPN           ; -01 -- PROJECT #
126 011476 000002G      .WORD   UPPN+2         ; -02 -- PROGRAMMER NUMBER
127                000006  MXNGTX =      .-NEGTAB
```

```

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 011500 010105 EMUNAM: MOV      R1,R5          ;GET JOB INDEX NUMBER
9 011502 013700      MOV      EMTBLK+2,R0      ;GET ADDRESS OF USER'S BUFFER
10 011506 004737      CALL     VALADW         ;VALIDATE THE ADDRESS
11 011512 123727      CMPB     EMTBLK,#2      ;GET OR SET PROGRAM NAME?
12 011520 002030      BGE      3$           ;BR IF GET/SET PROGRAM NAME
13 011522 012702      MOV      #6,R2         ;6 WORDS PER USER NAME
14 011526 070502      MUL      R2,R5         ;* 12 BYTES PER USER NAME
15 011530 062705      ADD      #LUNAME,R5    ;POINT TO USER NAME TABLE ENTRY
16 011534 105737      TSTB     EMTBLK        ;IS THIS A REQUEST TO GET OR SET USER NAME
17 011540 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 011542 032737      BIT       #PO$NAM,PRIVCO ; IS THIS JOB PRIVILEGED?
23 011550 001004      BNE      2$           ;BR IF YES
24 011552 012700      MOV      #1,R0         ;RETURN ERROR IF NOT
25 011556 000137      JMP      SETERR
26 011562 106520      2$:      MFPD     (R0)+      ;GET NEXT WORD OF USER NAME
27 011564 012625      MOV      (SP)+,(R5)+    ;MOVE TO SYSTEM TABLE
28 011566 077203      SOB      R2,2$        ;LOOP TO MOVE ALL OF NAME
29 011570 000432      BR       9$
30          ;
31          ; This is a request to get the user name.
32          ;
33 011572 012546      1$:      MOV      (R5)+,-(SP) ;STACK A WORD OF THE USER NAME
34 011574 106620      MTPD     (R0)+      ;MOVE TO USER'S BUFFER
35 011576 077203      SOB      R2,1$
36 011600 000426      BR       9$
37          ;
38          ; This is a request to get or set the program name
39          ;
40 011602 003007      3$:      BGT      4$           ;BR IF SETTING
41 011604 016146      MOV      LPRG1(R1),-(SP) ;GET FIRST 3 CHARS OF PRG NAME
42 011610 106620      MTPD     (R0)+      ;MOVE TO USER'S BUFFER
43 011612 016146      MOV      LPRG2(R1),-(SP) ;GET SECOND 3 CHARS OF PRG NAME
44 011616 106620      MTPD     (R0)+      ;MOVE TO USER'S BUFFER
45 011620 000416      BR       9$
46          ;
47          ; This is a request to set the program name
48          ;
49 011622 106520      4$:      MFPD     (R0)+      ;GET FIRST 3 CHARS
50 011624 012605      MOV      (SP)+,R5      ;OF NEW PGM NAME
51 011626 020527      CMP      R5,#175000    ;IS THIS A VALID RAD50 VALUE
52 011632 103013      BHIS     5$           ;BR IF NOT
53 011634 010561      MOV      R5,LPRG1(R1)  ;SET FIRST 3 CHARS OF PRG NAME
54 011640 106520      MFPD     (R0)+      ;GET SECOND 3 CHARS
55 011642 012605      MOV      (SP)+,R5      ;OF NEW PGM NAME
56 011644 020527      CMP      R5,#175000    ;IS THIS A VALID RAD50 VALUE
57 011650 103004      BHIS     5$           ;BR IF YES

```

```
58 011652 010561 0000000      MOV    R5,LPRG2(R1)    ;SET SECOND 3 CHARS OF PROG NAME
59 011656 000137 0000000      9#:   JMP    EMTXIT    ;FINISHED
60
61 011662 012700 0000002      5#:   MOV    #2,R0     ;SIGNAL ERROR CODE
62 011666 000137 0000000      JMP    SETERR
```

```
1 .SBTTL Request operator privilege
2 ;-----
3 ; Request operator privilege.
4 ; This is only legal if user has SYSPRV privilege.
5 ;
6 011672 123727 000000G 000001 REQPRV: CMPB EMTBLK,#1 ;Request to release locked program?
7 011700 001417 BEQ 2$ ;Br if yes
8 011702 032737 000000G 000000G BIT #PO$SPV,PRIVCO ;Does job have SETPRV privilege?
9 011710 001410 BEQ 1$ ;Br if not
10 011712 052737 000000C 000000G BIS #<PO$SPV!PO$SYS!PO$NAM!PO$BYP>,PRIVCO ;ENABLE PRIVILEGES
11 011720 052761 000000G 000000G BIS #$PRGLK,LSW5(R1);Lock program to line
12 011726 000137 000000G JMP EMTXIT
13 011732 005000 1$: CLR RO ;RETURN ERROR CODE OF 0
14 011734 000137 000000G JMP SETERR
15 ;
16 ; Release program lock
17 ;
18 011740 042761 000000G 000000G 2$: BIC #$PRGLK,LSW5(R1);Program no longer locked to line
19 011746 000137 000000G JMP EMTXIT
```

```

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 011752   GETBRK:
7          ;
8          ; See if we are connecting a completion routine to a break character
9          ; or to any input activation condition.
10         ;
11 011752   105737 000000G      TSTB   EMTBLK      ;Connecting to break character?
12 011756   001402          BEQ     3$           ;Br if yes
13 011760   000137 000000G      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 011764   005061 000000G      3$:   CLR     LBRKCH(R1) ;NO BREAK CHARACTER
20 011770   005037 000000G      CLR     URO         ;ASSUME NO BREAK ROUTINE NOW
21 011774   016100 000000G      MOV     LBRKCQ(R1),R0 ;GET ADDRESS OF BREAK COMPLETION QUEUE ENTRY
22 012000   001410          BEQ     1$           ;BR IF NONE
23 012002   016037 000000G 000000G  MOV     CQ$RTN(R0),URO ;RETURN ADDRESS OF CURRENT BREAKPOINT ROUTINE
24 012010   005061 000000G      CLR     LBRKCQ(R1)  ;SAY NO COMPLETION ENTRY
25 012014   010001          MOV     R0,R1       ;GET ADDRESS OF QUEUE ELEMENT TO R1 FOR QFREE
26 012016   004737 000000G      CALL    QFREE       ;RELEASE THE QUEUE ELEMENT
27         ;
28         ; See if we need to establish a completion routine.
29         ;
30 012022   005737 000004G      1$:   TST     EMTBLK+4 ;DID HE SPECIFY THE ADDRESS OF A COMPL RTN?
31 012026   001437          BEQ     2$           ;BR IF NOT
32         ;
33         ; Get a queue element and set up as a completion queue element.
34         ;
35 012030   004737 000000G      CALL    GETQ        ;GET A QUEUE ELEMENT
36 012034   113702 000000G      MOVB   CORUSR,R2    ;GET USER INDEX NUMBER
37 012040   110261 000000G      MOVB   R2,CQ$JOB(R1) ;SET JOB NUMBER IN QUEUE ELEMENT
38 012044   112761 000000G 000000G  MOVB   #S$TWFN,CQ$RNS(R1);SET EXECUTION STATE
39 012052   116261 000000G 000000G  MOVB   LPRI(R2),CQ$PRI(R1);SET EXECUTION PRIORITY
40 012060   112761 000000G 000000G  MOVB   #CP$STD,CQ$CP(R1);Set compl routine class priority
41 012066   013700 000004G      MOV     EMTBLK+4,R0 ;GET ADDRESS OF COMPL ROUTINE
42 012072   004737 000000G      CALL    UACHKW      ;MAKE SURE ADDRESS IS VALID
43 012076   103415          BCS     9$           ;BR IF INVALID
44 012100   010061 000000G      MOV     R0,CQ$RTN(R1) ;SET COMPL ROUTINE ADDRESS IN QUEUE ELEMENT
45 012104   013761 000000G 000000G  MOV     EMTMAP,CQ$PA5(R1);SET EMT ENTRY MAPPING FOR PAR 5
46 012112   010162 000000G      MOV     R1,LBRKCQ(R2) ;SET ADDRESS OF BREAK QUEUE ELEMENT
47         ;
48         ; Remember special user-defined break character.
49         ;
50 012116   113700 000002G      MOVB   EMTBLK+2,R0  ;GET USER-DEFINED BREAK CHARACTER
51 012122   010062 000000G      MOV     R0,LBRKCH(R2) ;SAVE FOR TT INTERRUPT PROCESSING
52         ;
53         ; Finished
54         ;
55 012126   000137 000000G      2$:   JMP     EMTXIT
56         ;
57         ; Error -- Invalid argument address.

```

```
58  
59 012132 004737 000000G ;  
9#: CALL QFREE ;FREE THE QUEUE ELEMENT  
60 012136 012700 177766 MOV #-12,R0 ;SET ERROR CODE  
61 012142 000137 000000G JMP SETERR ;ABORT THE EMT
```

1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13 012146 017700 000000G
 14 012152 002005
 15 012154 032700 000076
 16 012160 001002
 17 012162 000241
 18 012164 000207
 19 012166 000261
 20 012170 000207

```

.SBTTL  CHKTT -- CHECK I/O TO TT DEVICE
-----
;  CHKTT is called to see if the current I/O operation is being directed
;  to device TT or some other device.
;
;  Inputs:
;  CHNADR = Address of current channel block
;
;  Outputs:
;  C-flag cleared if I/O is directed to TT
;  C-flag set if I/O not directed to TT
;
CHKTT:  MOV     @CHNADR,RO      ;GET DEV TABLE INDEX
        BGE    1$             ;BRANCH IF NOT OPEN
        BIT    #76,RO         ;IS THIS TT DEVICE (INDEX # = 0)?
        BNE    1$             ;BR IF NOT TT
        CLC
        RETURN                ;SAY THIS IS TT
1$:     SEC
        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 012172 010146 CKIOST: MOV      R1, -(SP)
9 012174 032737 000000G 000000G BIT      #UMODE,EMTPS ;Was this EMT executed in kernel mode?
10 012202 001434 BEQ      2$ ;Br if yes -- Don't stop system EMT's
11 012204 105737 000000G TSTB    CURCP ;Was EMT executed by a completion routine?
12 012210 001031 BNE     2$ ;Br if yes -- I/O hold flag stops cpl rtns
13 012212 113701 000000G MOVB   CORUSR,R1 ;Get current job index #
14 012216 005761 000000G 1$: TST    LIOHLD(R1) ;Is I/O hold flag set for job?
15 012222 001424 BEQ     2$ ;Br if not
16 012224 DISABL ;** Disable interrupts **
17 012232 105761 000000G TSTB   LIOCNT(R1) ;Is any I/O in progress now?
18 012236 001407 BEQ     3$ ;Br if not
19 012240 012700 000000G MOV    #S$IOWT,R0 ;Get I/O wait state code
20 012244 004737 000000G CALL   QNSPND ;Enqueue in I/O wait state (enable ints)
21 012250 004737 000000G CALL   CHKABT ;See if we were aborted while asleep
22 012254 000760 BR      1$ ;See if we should continue waiting
23 012256 005061 000000G 3$: CLR   LIOHLD(R1) ;Reset I/O hold flag
24 012262 ENABL ;** Enable interrupts **
25 012270 004737 000000G CALL   SCHED ;Call scheduler to probably swap job
26                                     ;
27                                     ; Finished
28                                     ;
29 012274 012601 2$: MOV    (SP)+,R1
30 012276 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 012300 010246 SETQ:  MOV     R2,-(SP)
17 012302 010346      MOV     R3,-(SP)
18 012304 010446      MOV     R4,-(SP)
19         ;
20         ; Set up misc. parameters
21         ;
22 012306 013761 000000G 000000G      MOV     CHNNUM,Q.CHAN(R1);SET USER'S CHANNEL #
23 012314 013703 000000G      MOV     CHNADR,R3      ;GET ADDRESS OF CSW FOR CHANNEL
24 012320 010361 000000G      MOV     R3,Q.UCSW(R1)  ;SET POINTER TO CSW
25 012324 013761 000000G 000000G      MOV     @#KPAR6,Q.PA6(R1);Save context blk mapping to access channel
26 012332 016302 000000G      MOV     C.CSW(R3),R2  ;GET CHANNEL STATUS WORD
27 012336 042702 177701      BIC     #^C76,R2      ;EXTRACT DEVICE INDEX NUMBER
28 012342 110261 000000G      MOV     R2,Q.DEVX(R1) ;SAVE IN QUEUE ELEMENT CELL
29 012346 116361 000000G 000000G      MOV     C.DEVQ(R3),Q.UNIT(R1);SET UNIT NUMBER
30 012354 113702 000000G      MOV     CORUSR,R2    ;GET USER INDEX #
31 012360 006202      ASR     R2      ;Convert to job number
32 012362 110261 000000G      MOV     R2,Q.JOB(R1)  ;Store job # into queue element
33 012366 020227 000037      CMP     R2,#31.     ;Will number fit in old Q.JNUM field?
34 012372 101402      BLOS   12$      ;Br if yes
35 012374 012702 000037      MOV     #31.,R2    ;Signal that Q.JNUM field not valid
36 012400 072227 000003 12$:  ASH     #3,R2      ;Position for Q.JNUM field
37 012404 150261 000000G      BISB   R2,Q.JNUM(R1);STORE JOB #
38         ;
39         ; Copy original channel block into internal channel block that is
40         ; part of I/O queue element.
41         ;
42 012410 010100      MOV     R1,RO      ;Get address of I/O queue block
43 012412 062700 000000G      ADD     #Q.ICSW,RO  ;Get address of CSW within queue block
44 012416 010061 000000G      MOV     RO,Q.CSW(R1);Set pointer to internal CSW
45 012422 010302      MOV     R3,R2      ;Get pointer to original channel block
46 012424 012220      MOV     (R2)+,(RO)+;Copy original channel block to internal one
47 012426 012220      MOV     (R2)+,(RO)+
48 012430 012220      MOV     (R2)+,(RO)+
49 012432 012220      MOV     (R2)+,(RO)+
50 012434 011210      MOV     (R2),(RO)
51         ;
52         ; Set up buffer address
53         ;
54 012436 013702 000006G      MOV     EMTBLK+6,R2 ;GET # WORDS TO BE TRANSFERED
55 012442 001004      BNE     5$         ;BR IF WORD COUNT SPECIFIED
56 012444 123727 000001G 000032      CMPB   EMTBLK+1,#32;IS THIS A .SPFUN REQUEST?
57 012452 001104      BNE     1$         ;BR IF NOT -- THIS MUST BE A SEEK REQUEST

```

SETQ -- Do setup of I/O queue element

```

58 012454 013700 000004G      5$:   MOV     EMTBLK+4,R0      ;BASE ADDRESS OF BUFFER AREA
59 012460 032737 000000G 000000G   BIT     #UMODE,EMTPS    ;WAS EMT DONE IN USER OR KERNEL MODE?
60 012466 001417              BEQ     2$              ;BR IF IN KERNEL MODE (ASSUME ADDRESS IS OK)
61 012470 004737 000000G      CALL    UACHKB          ;MAKE SURE IT IS LEGAL
62 012474 103523              BCS     9$              ;BR IF ILLEGAL
63 012476 123727 000001G 000032   CMPB   EMTBLK+1,#32    ;IS THIS A .SPFUN OPERATION?
64 012504 001410              BEQ     2$              ;BR IF YES -- DON'T CHECK WORDCOUNT VALUE
65 012506 006302              ASL     R2              ;CVT # WORDS TO # BYTES
66 012510 005302              DEC     R2              ;GET ADDRESS OF LAST BYTE TRANSFERED
67 012512 060200              ADD     R2,R0
68 012514 004737 000000G      CALL    UACHKB          ;CHECK HIGH RANGE ADDRESS OF BUFFER
69 012520 103511              BCS     9$              ;BR IF ILLEGAL
70 012522 013700 000004G      MOV     EMTBLK+4,R0    ;GET BACK BASE ADDRESS OF BUFFER
71 012526 004737 013032'      2$:   CALL    CVTUAD          ;CONVERT USER ADDRESS TO PHYSICAL ADDRESS
72 012532 010061 000000G      MOV     R0,Q.BUFF(R1)  ;BUFFER ADDRESS RELATIVE TO Q.PAR
73 012536 010261 000000G      MOV     R2,Q.PAR(R1)  ;RELOCATION BASE ADDRESS
74
75 ; Set up word count and block number for transfer.
76 ;
77 012542 005037 000000G      CLR     URO            ;SAY # WORDS TRANSFERED = 0 (IN CASE OF ERROR)
78 012546 013704 000006G      MOV     EMTBLK+6,R4    ;GET WORD COUNT
79 012552 062704 000377      ADD     #377,R4        ;CONVERT TO BLOCK COUNT
80 012556 105004              CLRB   R4
81 012560 000304              SWAB   R4
82 012562 013700 000002G      MOV     EMTBLK+2,R0    ;GET BLOCK # FOR REQUEST
83 012566 011302              MOV     (R3),R2        ;GET CSW
84 012570 042702 177701      BIC     #^C76,R2       ;GET DEVICE INDEX
85 012574 032762 000000G 000000G   BIT     #DS#DIR,DVSTAT(R2); IS THIS A FILE STRUCTURED DEVICE?
86 012602 001007              BNE     10$            ;BR IF YES
87 012604 060400              ADD     R4,R0          ;COMPUTE HIGH BLOCK NUMBER FOR TRANSFER
88 012606 032713 000000G      BIT     #CS#EOF,(R3)   ;HAS END OF FILE BEEN HIT?
89 012612 001424              BEQ     1$              ;BR IF NOT
90 012614 042713 000000G      BIC     #CS#EOF,(R3)   ;ACKNOWLEDGE EOF
91 012620 000455              BR      7$              ;RETURN EOF STATUS CODE
92 012622 042713 000000G      10$:  BIC     #CS#EOF,(R3)   ;CLEAR END OF FILE FLAG
93 012626 005763 000000G      TST     C.SBLK(R3)     ;STARTING BLOCK FOR FILE = 0?
94 012632 001414              BEQ     1$              ;DIR OP IF YES
95 012634 016302 000000G      MOV     C.LENG(R3),R2  ;GET # BLOCKS ALLOCATED FOR FILE
96 012640 020002              CMP     R0,R2          ;IS THIS A VALID STARTING BLOCK #?
97 012642 103044              BHS     8$              ;BR IF INVALID STARTING BLOCK #
98 012644 060400              ADD     R4,R0          ;COMPUTE ENDING BLOCK NUMBER + 1
99 012646 020002              CMP     R0,R2          ;WILL TRANSFER PASS END OF FILE?
100 012650 101405              BLOS   1$              ;BR IF WITHIN FILE
101 012652 010200              MOV     R2,R0          ;DON'T PASS END OF FILE
102 012654 163702 000002G      SUB     EMTBLK+2,R2    ;CALCULATE # BLOCKS ACTUALLY TO BE TRANSFERRED
103 012660 000302              SWAB   R2              ;GET ACTUAL WORD COUNT FOR TRANSFER
104 012662 000402              BR      3$              ;
105 012664 013702 000006G      1$:   MOV     EMTBLK+6,R2    ;GET REQUEST WORD COUNT
106 012670 010237 000000G      3$:   MOV     R2,URO         ;RETURN ACTUAL WORD COUNT TO USER IN R0
107 012674 010261 000000G      MOV     R2,Q.WCNT(R1)  ;SET WORD COUNT IN Q ELEMENT
108 012700 013702 000002G      MOV     EMTBLK+2,R2    ;GET REQUEST BLOCK #
109 012704 066302 000000G      ADD     C.SBLK(R3),R2  ;ADD BASE BLOCK # OF FILE
110 012710 010261 000000G      MOV     R2,Q.BLKN(R1) ;SET PHYSICAL BLOCK #
111
112 ; See if this is an I/O operation to a logical disk
113 ;
114 012714 126137 000000G 000000G   CMPB   Q.DEVX(R1),LDDEVX; I/O TO LOGICAL DISK?

```

SETQ -- Do setup of I/O queue element

```

115 012722 001006          BNE      11$          ;BR IF NOT
116 012724 010046          MOV      RO,-(SP)        ;SAVE HIGH BLOCK NUMBER
117 012726          OCALL   LDIO          ;ALTER QUEUE ELEMENT FOR LOGICAL DISK I/O
118 012734 012600          MOV      (SP)+,RO        ;RESTORE HIGH BLOCK NUMBER
119 012736 103406          BCS      8$              ;BR IF NOT VALID LOGICAL DISK
120 012740 000241          11$:    CLC          ;SIGNAL SUCCESS ON RETURN
121 012742 000407          BR       4$
122          ; Invalid buffer address
123 012744 112737 177766 000000G 9$:    MOVB   #-12,INTERR    ;RETURN ERROR CODE 12
124 012752 000402          BR       6$
125          ; Invalid block number (return EOF status)
126 012754          8$:
127          ;      BIS      #CS$EOF,(R3)    ;SET END-OF-FILE FLAG IN CSW
128 012754 105037 000000G 7$:    CLR    INTERR          ;RETURN ERROR CODE OF 0
129 012760 000261          6$:    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 012762 016361 000000G 000000G 4$:    MOV      C.CSW(R3),Q.ICSW(R1);Copy CSW into Q element
136          ;
137          ; Finished
138          ;
139 012770 012604          MOV      (SP)+,R4
140 012772 012603          MOV      (SP)+,R3
141 012774 012602          MOV      (SP)+,R2
142 012776 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          ;   RO = 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 013000 020027 000001 SETCR:  CMP     RO,#1          ;COMPL RTN ADDR OF 1 ==> .READW/.WRITW
14 013004 101403          BLOS   2$          ;BR IF YES
15 013006 004737 000000G CALL   UACHKW       ;VALIDATE ADDRESS OF COMPLETION ROUTINE
16 013012 103406          BCS   1$          ;BR IF ILLEGAL
17 013014 010061 000000G 2$:   MOV    RO,Q.COMP(R1) ;STORE ADDRESS INTO QUEUE ELEMENT
18 013020 013761 000000G 000000G MOV    EMTMAP,Q.PA5(R1);SET EMT ENTRY PAR 5 MAPPING VALUE
19 013026 000241          CLC          ;SIGNAL GOOD RETURN
20 013030 000207          1$:   RETURN

```

```

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 013032 010146 CVTUAD: MOV     R1,-(SP)
33 013034 010346          MOV     R3,-(SP)
34 013036 113701 000000G          MOVB   CORUSR,R1          ;GET JOB INDEX NUMBER
35         ;
36         ; Determine if the EMT was done in user or kernel mode.
37         ;
38 013042 032737 000000G 000000G          BIT     #UMODE,EMTPS          ;WAS EMT DONE IN USER OR KERNEL MODE?
39 013050 001455          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 013052 032761 000000G 000000G          BIT     ##INKMN,LSW4(R1); IS KMON RUNNING?
45 013060 001405          BEQ     8$          ;BR IF NOT
46 013062 020027 000000G          CMP     R0,#CXTBAS          ;IS KMON DOING I/O TO JOB CONTEXT AREA?
47 013066 103051          BHIS   5$          ;BR IF YES
48 013070 162700 000000G          SUB     #KMNBAS,R0          ;SUBTRACT KMON ADDRESS BIAS
49 013074 010003          B$:   MOV     R0,R3          ;GET VIRTUAL ADDRESS
50 013076 005002          CLR     R2          ;CLEAR FOR SHIFT
51 013100 073227 0000003          ASHC   #3,R2          ;GET PAR VALUE FROM ADDRESS
52 013104 006302          ASL     R2          ;CONVERT TO WORD INDEX
53 013106 032761 000000G 000000G          BIT     ##UDSPC,LSW11(R1); Has user enabled I/D split?
54 013114 001006          BNE     4$          ;Check D-space if so
55 013116 005762 000000G          TST     RPDR(R2)          ;IS THIS I-space PAGE SPECIALLY MAPPED?
56 013122 001413          BEQ     6$          ;BR IF NOT
57 013124 016202 000000G          MOV     RPAR(R2),R2          ;GET BASE 64-BYTE BLOCK # FOR THIS PAGE

```

CVTUAD -- Convert user address to physical address

```

58 013130 000405          BR      10$      ;Go clean up virtual address
59 013132 005762 000000G  4$:    TST      RDDR(R2)    ;Is this D-space PAR specially mapped?
60 013136 001405          BEQ      6$      ;Br if not, use root memory
61 013140 016202 000000G  MOV      RDAR(R2),R2    ;If address in special D-space, get PAR
62 013144 042700 160000  10$:   BIC      #160000,R0    ;CLEAR PAR # FROM VIRTUAL ADDRESS
63 013150 000402          BR      7$      ;
64 013152 016102 000000G  6$:    MOV      LPARBS(R1),R2 ;GET BASE 64-BYTE BLOCK # FOR JOB
65 013156 010003 7$:    MOV      R0,R3      ;GET VIRTUAL ADDRESS
66 013160 000241          CLC                ;SHIFT RIGHT 6-BITS WITHOUT SIGN EXTENSION
67 013162 006003          ROR      R3
68 013164 072327 177773  ASH      #-5,R3
69 013170 060302          ADD      R3,R2      ;GET 64-BYTE BLOCK # OF PHYSICAL ADDRESS
70 013172 042700 177700  BIC      #^C77,R0    ;GET BYTE-WITHIN-BLOCK FROM VIRTUAL ADDRESS
71 013176 062700 000000G  ADD      #VPAR6,R0   ;MAP INTO PAR6 REGION
72 013202 000440          BR      9$      ;
73                          ;
74                          ; EMT was done in kernel mode.
75                          ; Determine which region to map to.
76                          ;
77 013204 020027 000000G  1$:    CMP      R0,#CXTBAS  ;IS ADDRESS IN JOB CONTEXT BLOCK?
78 013210 103405          BLD      2$      ;BR IF NOT
79                          ;
80                          ; Map to Job context block.
81                          ;
82 013212 016102 000000G  5$:    MOV      LBASE(R1),R2 ;GET BASE 256-WORD BLOCK # OF CONTEXT BLOCK
83 013216 072227 000003  ASH      #3,R2      ;CONVERT TO 32-WORD BLOCK #
84                          ; Note: Virtual address in R0 is already in PAR 6 range (140000-157777).
85 013222 000430          BR      9$      ;
86                          ;
87                          ; See if accessing the system mapped region.
88                          ;
89 013224 020027 000000G  2$:    CMP      R0,#VPAR5  ;IS ADDRESS IN MAPPED SYSTEM REGION?
90 013230 103405          BLD      12$     ;BR IF NOT
91                          ;
92                          ; Map to the buffer using the current contents of EMTMAP which indicates
93                          ; the mapped value for kernel PAR 5 when the EMT was done.
94                          ;
95 013232 013702 000000G  MOV      EMTMAP,R2    ;Get mapping value that PAR 5 had on entry
96 013236 062700 020000  ADD      #20000,R0    ;INCREASE VIRTUAL ADDRESS
97 013242 000420          BR      9$      ;
98                          ;
99                          ; Not in job context area.
100                          ; See if accessing base of job region.
101                          ;
102 013244 005700 12$:   TST      R0          ;VIRTUAL ADDRESS = 0?
103 013246 001005          BNE      3$      ;BR IF NOT
104                          ;
105                          ; Map to base of job space.
106                          ;
107 013250 016102 000000G  MOV      LPARBS(R1),R2 ;GET PAR OF JOB BASE
108 013254 062700 000000G  ADD      #VPAR6,R0    ;PUT VIRTUAL ADDRESS IN PAGE 6 REGION
109 013260 000411          BR      9$      ;
110                          ;
111                          ; Address is already physical.
112                          ;
113 013262 010002 3$:    MOV      R0,R2      ;GET ACTUAL ADDRESS
114 013264 000241          CLC                ;CONVERT TO PAGE #

```

```
115 013266 006002          ROR      R2
116 013270 072227 177773   ASH      #-5,R2
117 013274 042700 177700   BIC      #^C77,R0      ;GET BYTE-IN-PAGE TO R0
118 013300 062700 000000G   ADD      #VPA6,R0      ;MAP THROUGH PAR 6
119
120          ; Finished
121          ;
122 013304 012603   9#:    MOV      (SP)+,R3
123 013306 012601   MOV      (SP)+,R1
124 013310 000207   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 013312 010446 LDIO:  MOV     R4, -(SP)
14         ;
15         ; Change logical device number and unit number into physical
16         ; device number and unit number.
17         ;
18 013314 116104 000000G      MOVVB  Q.UNIT(R1),R4  ;GET LOGICAL DISK UNIT NUMBER
19 013320 042704 177770      BIC   #^C7,R4      ;GET UNIT # ONLY
20 013324 006304             ASL   R4             ;CONVERT TO WORD TABLE INDEX
21 013326 016400 000000G      MOV   LDPDEV(R4),RO  ;GET PHYSICAL UNIT # AND DEVICE #
22 013332 001456             BEQ   2$              ;BR IF THIS LOGICAL UNIT NOT IN USE
23 013334 110061 000000G      MOVVB  RO,Q.DEVX(R1) ;SET PHYSICAL DEVICE INDEX NUMBER
24 013340 142761 000007 000000G BICB  #7,Q.UNIT(R1) ;CLEAR OLD UNIT #
25 013346 000300             SWAB  RO             ;GET PHYSICAL UNIT # TO LOW BYTE
26 013350 150061 000000G      BISB  RO,Q.UNIT(R1) ;SET PHYSICAL UNIT #
27         ;
28         ; See if this is a .SPFUN to get device size or LD unit tables
29         ;
30 013354 116100 000000G      MOVVB  Q.FUNC(R1),RO  ;IS THIS A .SPFUN REQUEST?
31 013360 002023             BGE   1$              ;BR IF NOT
32 013362 105061 000000G      CLRB  Q.DEVX(R1)    ;TELL QIO NOT TO DO ANY ACTUAL I/O
33 013366 120027 000372      CMPB  RO,#372        ;IS THIS FUNCTION TO GET DEVICE TABLE?
34 013372 001006             BNE   5$              ;BR IF NOT
35 013374 005737 000006G      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 013400 000240             BPL   4$              ;IGNORE IT IF YES
40 013402 004737 013476'      NOP                    ;Reserve room to patch BR back in if needed
41 013406 000426             CALL  LDRST          ;PASS DEVICE TABLE INFO TO USER
42 013410 120027 000373      BR    4$              ;FINISHED
43 013414 001023             5$:  CMPB  RO,#373        ;IS THIS FUNCTION TO GET DEVICE SIZE?
44 013416 016446 000000G      BNE   4$              ;BR IF NOT -- IGNORE ALL OTHER SPECIAL FUNCTS
45 013422 106677 000004G      MOV   LDSIZE(R4),-(SP);GET LOGICAL DISK SIZE
46 013426 000416             MTPD  @EMTBLK+4     ;MOVE INTO USER'S BUFFER
47         ;
48         ; BR    4$              ;FINISHED
49         ;
50         ; Check to make sure that the transfer is completely within the
51         ; area of the logical disk.
52         ;
53 013430 016100 000000G      1$:  MOV   Q.WCNT(R1),RO  ;GET WORD COUNT FOR TRANSFER
54 013434 062700 000377      ADD   #377,RO      ;CONVERT TO BLOCK COUNT
55 013440 105000             CLRB  RO             ;
56 013442 000300             SWAB  RO             ;
57 013444 066100 000000G      ADD   Q.BLKN(R1),RO  ;ADD TRANSFER STARTING BLOCK NUMBER
58 013450 020064 000000G      CMP   RO,LDSIZE(R4) ;IS TRANSFER WITHIN LOGICAL DISK?
59 013454 101005             BHI   2$              ;BR IF NOT

```

```
58 ;  
59 ; Bias logical block number by base block of logical disk file  
60 ;  
61 013456 066461 0000000 0000000 ADD LDBASE(R4),Q.BLKN(R1) ;BIAS BLOCK NUMBER  
62 ;  
63 ; Transfer is ready to go  
64 ;  
65 013464 000241 4#: CLC ;SIGNAL SUCCESS ON RETURN  
66 013466 000401 BR 3#  
67 ;  
68 ; End of file hit by transfer  
69 ;  
70 013470 000261 2#: SEC ;SIGNAL ERROR ON RETURN  
71 ;  
72 ; Finished  
73 ;  
74 013472 012604 3#: MOV (SP)+,R4  
75 013474 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 013476 010246 LDRST: MOV      R2,-(SP)
20 013500 010346      MOV      R3,-(SP)
21 013502 010446      MOV      R4,-(SP)
22         ;
23         ; Pass flag table info to user
24         ;
25 013504 013702 000004G      MOV      EMTBLK+4,R2      ;Get address of user's buffer
26 013510 123727 000000G 000001  CMPB    LDVERS,#1      ;LD version format 5.3& earlier?
27 013516 001406          BEQ     11$           ;Skip first two words if old format
28 013520 012746 045640      MOV      #^RLD ,-(SP)   ;Pass LD device name to user
29 013524 106622          MTPD    (R2)+
30 013526 012746 000000G      MOV      #MAXLD,-(SP)  ;Pass number of LD units to user
31 013532 106622          MTPD    (R2)+
32 013534 005003          11$:   CLR     R3           ;Init LD index
33 013536 016300 000000G      1$:   MOV     LDPDEV(R3),R0  ;Get real disk device # and unit #
34 013542 001417          BEQ     2$           ;Br if unit is not allocated
35 013544 052700 100000      BIS     #100000,R0     ;Set unit-allocated flag in status
36 013550 032763 000000G 000000G  BIT     #LD#RON,LDFLAG(R3); Is the unit read-only?
37 013556 001411          BEQ     2$           ;Br if not read-only
38 013560 123727 000000G 000001  CMPB    LDVERS,#1      ;LD version format 5.3& earlier?
39 013566 001403          BEQ     22$          ;Use old mask for old format
40 013570 052700 040000      BIS     #040000,R0     ;Set read-only status flag (5.4&later)
41 013574 000402          BR     2$
42 013576 052700 020000      22$:  BIS     #020000,R0     ;Set read-only status flag (5.3&earlier)
43 013602 010046          2$:   MOV     R0,-(SP)     ;Pass status flags to user
44 013604 106622          MTPD    (R2)+
45 013606 062703 000002      ADD     #2,R3         ;Increment unit index
46 013612 020327 000000C      CMP     R3,#MAXLD*2   ;Done all units?
47 013616 103747          BLD    1$           ;Br if more to do
48         ;
49         ; Offset info table
50         ;
51 013620 012704 000000G      MOV     #LDBASE,R4    ;Point to table with base values for disks
52 013624 012700 000000G      MOV     #MAXLD,R0     ;Get number of units
53 013630 012446          3$:   MOV     (R4)+,-(SP)  ;Pass offset info to user
54 013632 106622          MTPD    (R2)+
55 013634 077003          SOB    R0,3$
56         ;
57         ; Size info table

```

```
58 ;
59 013636 012704 000000G      MOV      #LDSIZE,R4      ;Point to table with size info
60 013642 012700 000000G      MOV      #MAXLD,R0      ;Get # LD units
61 013646 012446      4$:    MOV      (R4)+,-(SP)    ;Pass size info to user
62 013650 106622      MTPD     (R2)+
63 013652 077003      SOB      R0,4$
64 ;
65 ; File name table
66 ;
67 013654 012704 000000G      MOV      #LDNAME,R4     ;Point to table with file name info
68 013660 012700 000000C      MOV      #MAXLD*4,R0    ;Get # words to move
69 013664 012446      5$:    MOV      (R4)+,-(SP)    ;Pass info to user
70 013666 106622      MTPD     (R2)+
71 013670 077003      SOB      R0,5$
72 ;
73 ; Finished
74 ;
75 013672 012604      MOV      (SP)+,R4
76 013674 012603      MOV      (SP)+,R3
77 013676 012602      MOV      (SP)+,R2
78 013700 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         ; CHNUM = 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 013702 010546 INDIO:  MOV      R5, -(SP)
22         ;
23         ; Determine if this I/O is directed to the IND data segment overlay
24         ;
25 013704 023727 000000G 000017          CMP      CHNUM, #17      ; IS I/O BEING DONE TO IND. SAV FILE?
26 013712 001034          BNE      9#                ; BR IF NOT
27 013714 023737 000002G 000000G      CMP      EMTBLK+2, INDDBL ; ARE WE ACCESSING DATA SEGMENT OVERLAY?
28 013722 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 013724 032761 000000G 000000G      BIT      %%INDDW, LSW6(R1); HAS DATA SEGMENT BEEN WRITTEN TO TEMP FILE?
34 013732 001007          BNE      1#                ; BR IF YES
35 013734 123727 000001G 000010      CMPB    EMTBLK+1, #10    ; IS THIS A READ OR WRITE REQUEST?
36 013742 001420          BEQ      9#                ; BR IF READ -- GO TO SAV FILE TILL 1ST WRITE
37 013744 052761 000000G 000000G      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 013752 010105          1#:  MOV      R1, R5                ; GET JOB #
43 013754 006205          ASR      R5                    ; DIVIDE BY 2
44 013756 005305          DEC      R5                    ; MAKE RELATIVE TO 0
45 013760 070537 000000G          MUL      INDDBS, R5                ; TIMES # OF BLOCKS IN THE OVERLAY SEGMENT
46 013764 010537 000002G          MOV      R5, EMTBLK+2          ; REPLACE BLOCK # FOR I/O OPERATION
47 013770 013703 000000G          MOV      CXTRMN, R3              ; GET ADDRESS OF JOB'S SIMULATED RMON
48 013774 062703 000000G          ADD      #R#INTC, R3            ; POINT TO CHANNEL BLOCK FOR INDTMP FILE
49 014000 010337 000000G          MOV      R3, CHNADR            ; DO I/O TO INDTMP FILE (SET NEW CSW ADDRESS)
50         ;
51         ; Finished
52         ;
53 014004 012605          9#:  MOV      (SP)+, R5
54 014006 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 014010 010246 EMTTRC: MOV     R2,-(SP)
8 014012 010346      MOV     R3,-(SP)
9 014014 010446      MOV     R4,-(SP)
10 014016 010546      MOV     R5,-(SP)
11          ;
12          ; Determine if we should trace this EMT
13          ;
14 014020 032761 000000G 000000G      BIT     $$INKMN,LSW4(R1); IS KMON RUNNING?
15 014026 001101      BNE     9$          ;BR IF YES -- DON'T TRACE KMON
16 014030 032761 000000G 000000G      BIT     $$CCLRN,LSW5(R1); IS CCL RUNNING?
17 014036 001075      BNE     9$          ;BR IF YES -- DON'T TRACE CCL
18 014040 032737 000000G 000000G      BIT     #UMODE,EMTPS      ;WAS EMT EXECUTED IN USER MODE?
19 014046 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 014050 012700 000015      MOV     #15,R0          ;PRINT CR
25 014054      .TTYOUT
26 014060 012700 000012      MOV     #12,R0          ;PRINT LF
27 014064      .TTYOUT
28          ; Print EMT address
29 014070 013705 000000G      MOV     EMTADR,R5      ;GET EMT ADDRESS
30 014074 012703 000005      MOV     #5,R3          ;PRINT 16-BIT VALUE
31 014100 004737 014244'      CALL    PRTOCT
32          ; Print EMT instruction low-order byte
33 014104 113705 000000G      MOV     CUREMT,R5      ;GET INSTRUCTION ARGUMENT BYTE
34 014110 012703 000003      MOV     #3,R3          ;PRINT 8-BITS
35 014114 004737 014244'      CALL    PRTOCT          ;PRINT IT
36          ; Print function code
37 014120 113705 000001G      MOV     EMTBLK+1,R5    ;GET FUNCTION CODE
38 014124 123727 000000G 000374      CMP     CUREMT,#374    ;IS THIS AN EMT 374?
39 014132 001003      BNE     3$          ;BR IF NOT
40 014134 162705 000046      SUB     #E375MX,R5     ;GET 374 FUNCTION CODE
41 014140 000405      BR     2$          ;GO PRINT IT
42 014142 020527 000062      3$:  CMP     R5,#RT11EX    ;IS THIS A TSX EMT?
43 014146 103402      BLO     2$          ;BR IF NOT
44 014150 062705 000016      ADD     #<100-RT11EX>,R5; GET ORIGINAL FUNCTION CODE VALUE
45 014154 004737 014244'      2$:  CALL    PRTOCT
46          ; Print channel number
47 014160 113705 000000G      MOV     EMTBLK,R5      ;GET CHANNEL #
48 014164 004737 014244'      CALL    PRTOCT          ;PRINT IT
49          ; Print EMT argument values
50 014170 012704 000002G      MOV     #EMTBLK+2,R4   ;POINT TO ARGUMENT BLOCK
51 014174 012702 000005      MOV     #5,R2          ;PRINT 5 ARGUMENT VALUES
52 014200 010203      MOV     R2,R3          ;PRINT 16-BITS EACH
53 014202 012405      1$:  MOV     (R4)+,R5      ;GET ARGUMENT VALUE
54 014204 004737 014244'      CALL    PRTOCT          ;PRINT IT
55 014210 077204      SOB     R2,1$          ;LOOP IF MORE TO PRINT
56          ; End of line
57 014212 012700 000015      MOV     #15,R0          ;PRINT CR

```

```
58 014216 . TTYOUT
59 014222 012700 000012 MOV #12, R0 ; PRINT LF
60 014226 . TTYOUT
61 ;
62 ; Finished
63 ;
64 014232 012605 9#: MOV (SP)+, R5
65 014234 012604 MOV (SP)+, R4
66 014236 012603 MOV (SP)+, R3
67 014240 012602 MOV (SP)+, R2
68 014242 000207 RETURN
```

```

1
2
3
4
5
6
7
8
9 014244 010346
10 014246 010446
11 014250 010546
12
13
14
15 014252 020327 000005
16 014256 001007
17 014260 012700 000030
18 014264 006105
19 014266 006100
20 014270
21 014274 000404
22
23
24
25 014276 042705 177400
26 014302 072527 000007
27
28
29
30 014306 012704 000006
31 014312 073427 000003
32 014316 010400
33 014320
34 014324 077310
35
36
37
38 014326 112700 000040
39 014332
40 014336
41
42
43
44 014342 012605
45 014344 012604
46 014346 012603
47 014350 000207
48 000001
  
```

```

.SBTTL PRTOCT -- Print an octal value
-----
; PRTOCT is called to print an octal value on the user's terminal.
;
; Inputs:
; R5 = Value to be printed.
; R3 = 8/16-bit flag: 3==>8-bit value, 5==>16-bit value
;
PRTOCT: MOV R3, -(SP)
        MOV R4, -(SP)
        MOV R5, -(SP)
;
; If this is a 16-bit value, print 1st digit now
;
        CMP R3, #5 ; IS THIS A 16-BIT VALUE?
        BNE 1$ ; BR IF NOT
        MOV #30, R0 ; GET 60 SHIFTED
        ROL R5 ; GET 1ST BIT OF VALUE
        ROL R0 ; INTO R0
        .TTYOUT ; PRINT IT
        BR 2$
;
; This is an 8-bit value, left justify the 8 bit quantity
;
1$: BIC #^C<377>, R5 ; CLEAR ALL BUT 8 BITS
    ASH #7, R5 ; LEFT JUSTIFY THE VALUE
;
; Begin loop to print the value
;
2$: MOV #6, R4 ; WILL BECOME 60 WHEN SHIFTED
    ASHC #3, R4 ; MOVE 3 BITS FROM R5 TO R4
    MOV R4, R0 ; GET TO R0 FOR TTYOUT
    .TTYOUT ; SEND THE CHARACTER
    SOB R3, 2$ ; LOOP IF MORE DIGITS TO PRINT
;
; Put in 2 spaces following value
;
        MOVB #' , R0 ; GET SPACE
        .TTYOUT ; SEND IT
        .TTYOUT ; " "
;
; Finished
;
        MOV (SP)+, R5
        MOV (SP)+, R4
        MOV (SP)+, R3
        RETURN
        .END
  
```

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 9458 Words (37 Pages)
 Size of core pool: 18176 Words (71 Pages)

Operating system: RT-11

Elapsed time: 00:01:11.49
,LP: TSEM2=DK: TSEM2/C/N: SYM

\$CCLRN	1-103	58-16											
\$CFOPN	1-84	14-21											
\$CTRLD	1-107	1-127	14-24										
\$CTRLS	1-85	44-67											
\$DBGMD	1-120	41-67	41-70										
\$DEAD	1-128	44-26											
\$DETCH	1-86	32-34											
\$DILUP	1-152	32-32											
\$EMTTR	1-92	4-14											
\$FORM	1-119	1-163											
\$HARD	1-128	44-24											
\$INDDW	1-123	57-33	57-37										
\$INDRN	1-110	8-75	9-91										
\$INKMN	1-104	4-36	32-38	54-44	58-14								
\$IOMAP	1-164	27-89											
\$KINIT	1-167	38-34											
\$MLOCK	1-167	38-67											
\$NDINT	1-134	42-9	42-14										
\$ODTMD	1-132	41-66	41-69										
\$PRGLK	1-104	48-11	48-18										
\$TTGAG	1-138	32-36											
\$UDSPC	1-160	54-53											
\$V52EM	1-160	40-11											
\$VIRJB	1-97	12-90											
\$VNOTT	1-85	46-98											
... V1	15-40	58-25	58-27	58-58	58-60	59-20	59-33	59-39	59-40				
... V2	15-40	15-40#											
ABORT	1-131												
ABRTAD	1-102	37-20*											
ABRTCD	1-102	37-21*											
ABTIO	1-88	2-60											
ADM3A	1-161	40-30											
ALCEMT	1-90	2-112											
ASTXIT	3-18	35-5#											
BADEMT	1-76	2-19	2-23	2-29	2-30	2-39	2-58	4-25	35-12	37-16			
C. CSW	1-144	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	
	19-15	20-9	20-20	20-22*	26-9	26-11	26-23	26-39	26-46	52-26	52-135		
C. DEVQ	1-146	17-32	26-36	52-29									
C. LENG	1-147	19-23	26-29	52-95									
C. NUMQ	1-145	17-31*											
C. SBLK	1-147	8-83	8-108	26-26	52-93	52-109							
C. USED	1-144	8-104	8-112*	8-113*	19-25*	26-32							
CANMKT	1-74	15-21	24-56	25-9#									
CAPOPN	2-87	5-12#											
CDFN	2-21	26-63#											
CFACFL	1-128	27-29											
CFLAG	1-110	8-121											
CFPNT	1-99	27-27											
CHKABT	1-135	22-14	33-24	51-21									
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	50-13#										
CHKUSP	1-91												
CHNADR	1-126	4-51*	8-6	9-6	11-6	16-5	17-5	17-25	18-5	19-5	20-5	26-5	

	50-13	52-23	57-49*											
CHNERR	4-40	4-71#												
CHNNUM	1-121	4-42*	18-9	52-22	57-25									
CHNSIZ	1-118	4-49												
CHSTAT	2-31	26-5#												
CINDAT	1-144	14-10	29-18											
CINFLG	1-71	14-5*												
CKINAC	2-85	41-77#												
CKIOST	8-5	9-5	11-5	51-8#										
CKWIN	2-115	5-75#												
CLEMT	1-84	2-111												
CLKOPN	2-66	5-6#												
CLOSE	1-122	2-15	2-55	18-26	19-28									
CLOSZ	2-45	19-5#												
CLRDIR	1-108	8-91												
CLZERR	1-122	4-57*	18-31	19-17*	19-27*									
CMDB	1-176	31-5												
CMDC	1-176	31-6												
CMDE	1-176	31-8												
CMDF	1-176	31-9												
CMDG	1-176	31-10												
CMDH	1-176	31-11												
CMDI	1-177	31-12												
CMDJ	1-177	31-13												
CMDK	1-177	31-14												
CMDL	1-177	31-15												
CMDM	1-177	31-16												
CMDN	1-177	31-17												
CMDO	1-177	31-18												
CMDR	1-178	31-21												
CMDS	1-178	31-22												
CMDT	1-178	31-23												
CMDU	1-178	31-24												
CMDW	1-178	31-26												
CMDX	1-178	31-27												
CMDY	1-178	31-28												
CMDZ	1-178	31-29												
CMKT	2-27	24-5#												
CONFIG	1-165	12-29												
CORUSR	1-127	4-13	23-14	25-15	49-36	51-13	52-30	54-34						
CP\$STD	1-119	49-40												
CPYEMT	1-90	2-114												
CQ\$CP	1-119	49-40*												
CQ\$FLG	1-109	24-17	25-17											
CQ\$HOT	1-157	23-20*	24-35											
CQ\$JOB	1-99	23-14*	24-15	25-15	49-37*									
CQ\$LNK	1-99	23-36*	24-11	24-13	24-26	24-26*	25-11	25-13	25-19	25-19*				
CQ\$LOT	1-157	23-22*	24-37											
CQ\$PA5	1-99	23-29*	49-45*											
CQ\$PRI	1-130	49-39*												
CQ\$RO	1-99	23-31*	24-19											
CQ\$RNS	1-130	49-38*												
CQ\$RTN	1-99	23-27*	49-23	49-44*										
CS\$ENT	1-142	8-106	16-9	17-29	19-15									
CS\$EOF	1-143	9-71	9-73	9-136	9-138	11-79	11-81	52-88	52-90	52-92				
CS\$ERR	1-142	8-67	8-69	8-148	8-150	9-63	9-65	9-128	9-130	11-74	11-76	20-20		

GVBAD	28-5	28-7	28-16	28-27	28-33	28-45	28-50#			
GVJMP	27-13	28-55#								
HAZEL	1-161	40-29								
HERR	2-54	10-12#								
HITRM	40-16	40-37#								
HRESET	3-19	15-6#								
INDDBL	1-123	57-27								
INDDBS	1-123	57-45								
INDFIL	1-179	43-40								
INDIO	8-77	9-93	57-21#							
INTEMT	2-109	42-4#								
INTERR	1-111	9-104*	9-151	52-123*	52-128*					
INTPRI	1-100	23-38	24-27	24-50	25-20	25-28	33-31	34-25	51-24	
IOABFL	1-129	18-24								
IOABRT	8-160	9-100	9-105	9-150#	11-91					
IOHALT	1-105	15-6								
IOWAIT	1-74	8-144	9-124	11-70	16-16	18-8	20-16			
JBINFO	2-102	38-15#								
JCDB	1-128	8-136	9-80	9-114	16-20	17-40	18-13			
JIBASE	38-116#	38-175								
JICONT	38-110#	38-174								
JICPU	38-133#	38-178								
JIDLN	1-166	38-65								
JIJMPX	38-53	38-171#	38-182							
JIMEM	38-102#	38-173								
JIMLOK	1-166	38-69								
JINAME	38-148#	38-180								
JIPAPR	38-159#	38-181								
JIPPN	38-127#	38-177								
JIPRIO	38-143#	38-179								
JIPRIV	1-166	38-81								
JIPROG	38-121#	38-176								
JIRET1	38-84	38-96	38-98	38-104#	38-112	38-117	38-144			
JIRET2	38-123	38-129	38-135#							
JIRNND	38-93	39-31#								
JIRNST	38-88	39-5#								
JIRUN	38-88#	38-172								
JISTAT	38-57#	38-171								
JIVLN	1-166	38-63								
JNOP	12-9	12-18#	12-34							
JSWLOC	1-137	12-89*								
KMNBAS	1-113	54-48								
KMNEMT	1-77	2-88								
KPAR6	1-86	52-25								
LA120	1-161	40-32								
LA36	1-161	40-31								
LACTIV	1-100	41-77								
LBASE	1-149	38-116	54-82							
LBRKCH	1-129	49-19*	49-51*							
LBRKCQ	1-129	49-21	49-24*	49-46*						
LBSPRI	1-130	46-94	46-96*							
LCONTM	1-153	38-111	46-110							
LCPUHI	1-168	38-133								
LCPULO	1-168	38-134								
LD#RON	1-98	56-36								
LDBASE	1-98	55-61	56-51							

LDDEVX	1-113	11-34	52-114						
LDFLAG	1-98	56-36							
LDIO	52-117	55-13#							
LDMEMT	1-85	2-117							
LDNAME	1-105	56-67							
LDPDEV	1-98	55-21	56-33						
LDRST	55-40	56-19#							
LDSIZE	1-98	55-44	55-56	56-59					
LDVERS	1-98	56-26	56-38						
LEMTPC	1-160	29-5*							
LIOCNT	1-133	51-17							
LIOHLD	1-133	51-14	51-23*						
LITIME	1-134	42-15*							
LJSW	1-121	12-87*	12-88	15-38	29-9				
LMSGBF	1-140	34-15	34-23*						
LNBLKS	1-167	38-102							
LNMAP	1-140	34-29							
LNPRIM	1-140	27-136	27-165	34-14	38-159	40-10	41-19	44-21	
LNSBLK	1-165	38-103							
LOOKUP	1-87	2-9							
LOTRM	40-8	40-20	40-27#						
LPARBS	1-115	54-64	54-107						
LPARNT	1-106	27-145	38-161						
LPRG1	1-168	38-121	47-41	47-53*					
LPRG2	1-168	38-122	47-43	47-58*					
LPRI	1-130	27-111	38-143	46-97*	49-39				
LPROG	1-105	27-101	38-128	40-48	46-118				
LPROJ	1-105	27-96	38-127	40-46	46-117				
LSCCA	1-132	12-5	12-8*						
LSECPT	1-140	27-166	41-28						
LSLEPH	1-124	22-8*	22-15						
LSLEPL	1-124	22-10*	22-18						
LSPND	1-125	15-17*							
LST16	3-4#	4-66							
LSTATE	1-141	38-89							
LSTDL	1-158	27-163	38-61	41-15					
LSTHL	1-116	1-121	44-22						
LSTPL	1-152	38-59	41-13						
LSTSL	1-167	32-30	38-27	44-19					
LSW	1-107	32-32	32-34	38-34	46-98	46-111			
LSW11	1-160	40-11	54-53						
LSW2	1-103	46-112							
LSW2S	1-133	46-119							
LSW3	1-124	1-126	14-24*	44-24	44-26	44-67*	46-113		
LSW4	1-104	4-36	14-21	32-38	41-66*	41-69*	46-114	54-44	58-14
LSW5	1-104	8-75	9-91	46-115	48-11*	48-18*	58-16		
LSW6	1-120	4-14	27-89	38-67	41-67*	41-70*	57-33	57-37*	
LSW7	1-154	32-36	42-9*	42-14*					
LSW9	1-153	12-90*	46-116						
LTRMTP	1-162	40-9							
LUNAME	1-106	38-151	47-15						
MAXCC	1-173	30-20							
MAXGVL	1-102	27-21	28-6	28-19	28-36				
MAXLD	1-105	56-30	56-46	56-52	56-60	56-68			
MAXPRI	1-134	46-90							
MAXSEC	1-140	27-170	41-24						

PRIVCO	1-103 48-8	4-16 48-10*	11-32	21-27	27-82	28-26	28-44	32-16	32-20	38-76	46-19	47-22
PRIVC2	1-96	5-63	7-35	44-33								
PRTOCT	58-31	58-35	58-45	58-48	58-54	59-9#						
PS	1-24# 34-25*	23-35* 51-16*	23-38* 51-24*	24-12*	24-27*	24-50*	25-12*	25-20*	25-28*	33-9*	33-31*	34-13*
PURGE	2-52	18-5#	19-18									
PVAL	28-4#	28-57										
PVSPBL	1-108	41-52										
Q. BLKN	1-145	52-110*	55-55	55-61*								
Q. BUFF	1-112	52-72*										
Q. CHAN	1-112	52-22*										
Q. COMP	1-113	53-17*										
Q. CSW	1-112	52-44*										
Q. DEVX	1-98	52-28*	52-114	55-23*	55-32*							
Q. FUNC	1-145	11-46*	55-30									
Q. ICSW	1-118	8-113*	52-43	52-135*								
Q. JNUM	1-112	52-37*										
Q. JOB	1-112	52-32*										
Q. PA5	1-113	53-18*										
Q. PA6	1-86	52-25*										
Q. PAR	1-113	52-73*										
Q. UCSW	1-118	52-24*										
Q. UNIT	1-146	52-29*	55-18	55-24*	55-26*							
Q. WCNT	1-144	8-102	8-114*	52-107*	55-51							
QF\$IOT	1-109	24-17	25-17									
QFREE	1-73	9-150	23-46	24-42	25-21	49-26	49-59					
QHDSFN	1-156	22-21										
QIO	1-72	8-132	9-110	11-61								
QMSG	1-80	32-72	34-9#									
QNSPND	1-139	33-23	51-20									
QSET	3-15	12-42#										
QUME	1-162	40-34										
R\$CFST	1-128	27-26*	27-29*									
R\$CHN	1-118	1-159	4-44	4-48	13-14							
R\$DATE	1-159	21-20*										
R\$INTC	1-123	57-48										
R\$XCHN	1-118	4-48										
RC\$BAS	1-175											
RCTOCF	14-22#											
RCTRLO	3-17	14-21#										
RDAR	1-108	54-61										
RDDR	1-108	54-59										
READ	1-70	2-16	9-5#									
REDCXT	1-85	38-78										
RENAME	1-87	2-12										
REOPEN	2-14	17-5#										
REQPRV	2-91	48-6#										
RF\$WRT	1-120											
RLOCK	2-68	5-26#										
RMNBAS	1-150	13-14	28-17	28-19	28-21	28-34	28-36	28-38				
RPAR	1-108	54-57										
RPDR	1-108	54-55										
RSFBLK	1-179	43-38										
RSSPAC	1-177	31-19										
RT11EX	1-69	2-62#	4-29	58-42	58-44							

SETSIZ	2-99	12-72#			
SETSPD	1-101	44-62			
SETTOP	3-16	12-49#			
SETTP1	12-50#	12-74	12-76		
SFCB	1-100				
SFCBND	1-100				
SFCLS	1-144	18-15			
SFDATE	1-87	2-42			
SFPA	2-32	12-24#			
SFPROT	1-87	2-43			
SFRSST	1-149	17-42			
SFSPND	2-84	5-53#			
SFSVST	1-138	5-54	16-22		
SFTIME	1-89	2-104			
SFWAC	1-177	31-20			
SFWL	1-178	31-25			
SFWRIT	1-110	8-138			
SITINF	2-86	41-87#			
SNDMSG	2-89	32-16#			
SNMSHD	1-139	33-14	33-30*		
SPCPS	1-127	36-11	36-17*		
SPFUN	2-34	11-5#			
SPLBLK	1-179	43-37			
SPLEMT	1-73	2-107			
SPLTF	46-33	46-105#	46-106		
SPPRED	1-85	46-100			
SRESET	3-14	15-7	15-13#		
SSEMT	1-79	2-101			
STOP	1-107	14-15	29-29	37-22	38-74
STTCPL	1-86	49-13			
SUTOP	1-137	12-93			
SVSTAT	2-13	16-5#			
SWDBLK	1-179	43-36			
SYFEMT	2-118	43-9#			
SYFTBL	43-18	43-36#	43-41		
SYNAME	1-129	27-151			
SYSDAT	1-101	21-17	21-20	21-40*	
SYSHLT	1-42				
SYTIMH	1-101	21-7	21-47*		
SYTIML	1-101	21-9	21-49*		
TBLEMT	2-90	46-14#			
TGCPRI	27-57	27-111#			
TGIOMP	27-53	27-89#			
TGJOBN	27-50	27-71#			
TGLDIN	27-51	27-77#			
TGLICN	27-56	27-106#			
TGMPRI	27-58	27-116#			
TGPRHI	27-63	27-126#			
TGPRIL	27-60	27-136#			
TGPRIV	27-52	27-82#			
TGPRLO	27-62	27-121#			
TGPRNT	27-64	27-145#			
TGPROG	27-55	27-101#			
TGPROJ	27-54	27-96#			
TGRSUB	27-66	27-161#			
TGSYNM	27-61	27-151#			

WRTSPL	8-49	8-164#				
XGTLN	2-74	41-8#				
XGTTAB	2-76	45-8#				
XHIIN	1-88	2-79				
XHIOUT	1-88	2-78				
XHISSET	1-89	2-82				
XJNOP	41-32	41-40	41-57	41-59#	41-68	41-71
XMSEND	2-70	7-6#				
XMSGCK	2-71	7-15#				
XMSGWT	2-72	7-24#				
XODTMD	2-75	41-64#				
XRDTIM	1-88	2-81				
XSPLSP	2-73	41-46#				
XTERCK	1-88	2-80				

