Table of contents

Table of contents

```
                                        .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.
                                ;
000000                                  .CSECT  TSEM2
000000  020550                  TSEM2:  .RAD50  /EM2/            ;Overlay id
        177776                  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
```

```
                        .ENDC
                        CALL    OVRHC           ;CALL THE OVERLAY HANDLER
                        .WORD   ENTADD          ;SPECIFY THE ENTRY POINT
                        .ENDM
                ;
                ;  Macro calls
                ;
                        .MCALL  .PURGE,.TTYOUT
                ;
                ;  Global definitions
                ;
                        .GLOBL  E375MX,EMT376,RT11EX
                        .GLOBL  TSEM2,READ,WRITE,DOEMT
                        .GLOBL  CINFLG,SDCLOS,UACHKW,GETQ
                        .GLOBL  QIO,GETUCH
                        .GLOBL  QFREE,SPLEMT
                        .GLOBL  IOWAIT,CANMKT
                        .GLOBL  SETERR,EMTXIT
                        .GLOBL  BADEMT
                        .GLOBL  KMNEMT
                        .GLOBL  DETEMT,PMEMT
                        .GLOBL  SSEMT
                        .GLOBL  QMSG
                ;
                ;  Global references
                ;
                        .GLOBL  FRECXT,$CFOPN,GTLICN,CLEMT
                        .GLOBL  $VNOTT,SPPRED,$CTRLS,LDMEMT,GETCXT,REDCXT,CXTBUF
                        .GLOBL  $DETCH,RUNDEV,Q.PA6,KPAR6,STTCPL,TSXVRS,VLSEMT
                        .GLOBL  DELETE,LOOKUP,ENTER,RENAME,RTDEV,SFDATE,SFPROT
                        .GLOBL  RTSPND,RTRSUM,ABTIO,XHIOUT,XHIIN,XTERCK,XRDTIM
                        .GLOBL  XHISET,MOUNT,DISMNT,RTEMT,GFINFO,SFTIME,SETPRI
                        .GLOBL  ALCEMT,MONEMT,CPYEMT,TTYIN,TTYOUT,DSTAT,FETCH
                        .GLOBL  CSIGEN,CSISPC,PRINT,CHKUSP,EMTASP,EMTSP,EMTASP
                        .GLOBL  $EMTTR,PO$DBG,NLCHN,DOCOPN,DOOPAP,DOCULK,DORLK
                        .GLOBL  DOTLK,DOULK1,DOSFCK,VMXSF,P2$RLK,VMXWIN,EMTWIN,EMTPLS
                        .GLOBL  MSEND,MSGCK,MSGWT,P2$MSG
                        .GLOBL  PO$OPR,PO$SYS,P2$TRM,PO$NAM
                        .GLOBL  PRIVC2,PO$SND,PO$SPF,PO$BYP
                        .GLOBL  TSXSIT,DS$SFN,$VIRJB,VIMAGE,PO$SPV
                        .GLOBL  LDPDEV,LDSIZE,LDBASE,Q.DEVX,LDFLAG,LD$RON,LDVERS
                        .GLOBL  CQ$LNK,CQ$RTN,CQ$PA5,CQ$JOB,CQ$RO,VPLAS,CFPNT
                        .GLOBL  INTPRI,UPPN,VPAR6,NFRESB,NSPLBL,LACTIV,SFCB,SFCBND
                        .GLOBL  SYTIMH,SYTIML,SYSDAT,SETSPD,VMAXMC,MSGABT,MONABT
                        .GLOBL  PNAME,MAXGVL,NUCHN,ABRTCD,ABRTAD
                        .GLOBL  EMTBLK,LSW2,$CCLRN,EMTPS,PRIVCO
                        .GLOBL  LSW4,LSW5,$INKMN,VALADW,VALADB,$PRGLK
                        .GLOBL  LPROJ,LPROG,IOHALT,EMTADR,MAXLD,LDNAME
                        .GLOBL  SERFLG,UTRPAD,LUNAME,UCLBLK,LPARNT
                        .GLOBL  LSW,$CTRLO,STOP,EXCJOB,CUREMT,DH$$LB
                        .GLOBL  RPAR,RPDR,RDAR,RDDR,ODTBAS,CLRDIR,PVSPBL
                        .GLOBL  CQ$FLG,QF$IOT,DX$EBA,DVFLAG,WRITTT
                        .GLOBL  CS$RON,DS$RON,$INDRN,DS$DIR,CFLAG,SFWRIT
                        .GLOBL  SDMOVE,FAKCMP,TTREAD,DS$WON,DCRD1,INTERR,NUMDCD
                        .GLOBL  DCRD2,Q.CHAN,Q.CSW,Q.JOB,UACHKB,Q.BUFF,Q.JNUM
                        .GLOBL  Q.PAR,LDDEVX,Q.COMP,Q.PA5,CXTBAS,KMNBAS
                        .GLOBL  S$TWFN,S$OTLO,S$QMIO,S$QCCB,S$QSPD,S$TTSC
```

```
                        .GLOBL  S$TTFN,S$HICP,S$RT,S$WFM,S$LOW,LPARBS,VPAR5
                        .GLOBL  LSTHL
                        .GLOBL  SCHED
                        .GLOBL  R$CHN,CHNSIZ,R$XCHN,Q.ICSW,Q.UCSW
                        .GLOBL  $FORM,CQ$CP,CP$STD
                        .GLOBL  RF$WRT,LSW6,$DBGMD
                        .GLOBL  EMTMAP,CHNNUM,LJSW,LSTHL
                        .GLOBL  CLOSE,CLZERR
                        .GLOBL  INDDBL,INDDBS,R$INTC,$INDDW
                        .GLOBL  LSW3,LSLEPH,LSLEPL
                        .GLOBL  PASLIN,SCHAIN,LSPND
                        .GLOBL  CHNADR,EMTCAD,EMTRAD,LSW3
                        .GLOBL  CORUSR,CURCP,SPCPS,$CTRLO,EMTCAS
                        .GLOBL  CXTRMN,UFPTRP,JCDB,$HARD,$DEAD,CFACFL,R$CFST
                        .GLOBL  VTSLCH,LBRKCQ,LBRKCH,SYNAME,IOABFL
                        .GLOBL  MXJPRI,LPRI,LBSPRI,CQ$RNS,CQ$PRI
                        .GLOBL  ABORT,VSWPFL,VPRIDF,VPRILO,VPRIHI
                        .GLOBL  $ODTMD,LSCCA,MXJADR,FPUUSE,CW$FPU
                        .GLOBL  PO$MEM,LSW2S,LIOHLD,LIOCNT
                        .GLOBL  MAXPRI,LITIME,$NOINT
                        .GLOBL  URO,CHKABT,NSPLDV
                        .GLOBL  S$NEDQ,DVSTAT
                        .GLOBL  JSWLOC,SUTOP,TTOPTS
                        .GLOBL  SFSVST,$TTGAG,SB$TXT,SB$END
                        .GLOBL  NMUMB,SNMSHD,QNSPND,SB$LNK,SB$PNT
                        .GLOBL  LNPRIM,LMSGBF,TRNSTR,LNMAP,MAXSEC,LSECPT
                        .GLOBL  EMTERR,LSTATE,S$IOWT,UMODE
                        .GLOBL  CS$ERR,CS$NMX,CS$ENT
                        .GLOBL  CS$EOF,CS$OPN,CS$NMX,DS$NRD
                        .GLOBL  C.CSW,C.USED,Q.WCNT,CINDAT,SFCLS
                        .GLOBL  C.NUMQ,Q.BLKN,Q.FUNC
                        .GLOBL  C.DEVQ,Q.UNIT
                        .GLOBL  C.SBLK,C.LENG
                        .GLOBL  UCHAN,CSIARE,OVLBIT
                        .GLOBL  LBASE,CS$SPL,SFRSST
                        .GLOBL  RMNBAS
                        .GLOBL  S$WSMB
                        .GLOBL  LSTPL,$DILUP
                        .GLOBL  LCONTM,LSW9
                        .GLOBL  S$IOFN,LSW7,S$OTFN
                        .GLOBL  S$TMWT
                        .GLOBL  QHDSPN,NUMDEV
                        .GLOBL  MRKTHD,CQ$HOT,CQ$LOT
                        .GLOBL  LSTDL
                        .GLOBL  R$CHN,R$DATE
                        .GLOBL  LEMTPC,LSW11,$V52EM,$UDSPC
                        .GLOBL  VT52,VT100,HAZEL,ADM3A,LA36,LA120
                        .GLOBL  DIABLO,QUME,LTRMTP,VT2007,VT2008
                        .GLOBL  $FORM
                        .GLOBL  $IOMAP
                        .GLOBL  CONFIG,LNSBLK
                        .GLOBL  JIVLN,JIDLN,JIMLOK,JIPRIV
                        .GLOBL  LSTSL,$KINIT,$MLOCK,LNBLKS,MINTIM
                        .GLOBL  LPRG1,LPRG2,LCPUHI,LCPULO
                        .GLOBL  S$INWT,S$OTWT,S$SPND,S$SFWT,S$MSWT
                        .GLOBL  S$QUSR,S$SFWT,S$SPDB,S$SPCB
                        .GLOBL  S$CPU
```

```
                                        .GLOBL  SETC
                                        .GLOBL  MAXCC
                                        .GLOBL  CSIARE
                                        .GLOBL  RC$BAS
                                        .GLOBL  SETRBF,CMDB,CMDC,GTSPAC,CMDE,CMDF,CMDG,CMDH
                                        .GLOBL  CMDI,CMDJ,CMDK,CMDL,CMDM,CMDN,CMDO,RSSPAC,SFWAC
                                        .GLOBL  CMDR,CMDS,CMDT,CMDU,SFWL,CMDW,CMDX,CMDY,CMDZ
                                        .GLOBL  SWDBLK,SPLBLK,RSFBLK,UCLDAT,INDFIL
                                        .GLOBL  GETDSS,SETDSS,MS$DTR
                                ;  Globals used for system mapped regions
                                        .GLOBL  OVRHC
                                ;
                                ;  EMT error code definitions
                                ;
000000                          EC0     = 0
000001                          EC1     = 1
000002                          EC2     = 2
000003                          EC3     = 3
000004                          EC4     = 4
                                ;
```

```
                                        .SBTTL  Branch vector for processing routines
                                ;-------------------------------------------------------
                                ;
                                ;  DEFINE PROCESSING ROUTINES FOR EACH EMT CODE
                                ;
                                ;  EMT 375 FUNCTIONS
                                ;
000002  000000G        EMTLST: .WORD   DELETE          ;  0  DELETE
000004  000000G                .WORD   LOOKUP          ;  1  LOOKUP
000006  000000G                .WORD   ENTER           ;  2  ENTER
000010  002762'                .WORD   TRPSET          ;  3  SET TRAP
000012  000000G                .WORD   RENAME          ;  4  RENAME
000014  003520'                .WORD   SVSTAT          ;  5  SAVE STATUS
000016  003644'                .WORD   REOPEN          ;  6  REOPEN
000020  000000G                .WORD   CLOSE           ;  7  CLOSE
000022  001674'                .WORD   READ            ; 10  READ(C)(W)
000024  001106'                .WORD   WRITE           ; 11  WRITE(C)(W)
000026  004236'                .WORD   DOWAIT          ; 12  WAIT
000030  000000G                .WORD   BADEMT          ; 13  COPY CHANNEL
000032  000000G                .WORD   RTDEV           ; 14  DEVICE
000034  005332'                .WORD   CDFN            ; 15  DEFINE CHANNELS
000036  000554'                .WORD   EMT16           ; 16  GROUP 16 EMTS
000040  000000G                .WORD   BADEMT          ; 17
000042  003254'                .WORD   GTJB            ; 20  GTJB
000044  004316'                .WORD   GETTIM          ; 21  GET TIME OF DAY
000046  004542'                .WORD   MRKT            ; 22  MARK TIME
000050  004702'                .WORD   CMKT            ; 23  CANCEL MARK TIME
000052  004456'                .WORD   TIMWAT          ; 24  TIMED WAIT
000054  000000G                .WORD   BADEMT          ; 25  SEND DATA
000056  000000G                .WORD   BADEMT          ; 26  RECEIVE DATA
000060  005170'                .WORD   CHSTAT          ; 27  CHANNEL STATUS
000062  003002'                .WORD   SFPA            ; 30  SET FPP EXCEPTION
000064  000000G                .WORD   EMTXIT          ; 31  PROTECT VECTORS
000066  002452'                .WORD   SPFUN           ; 32  SPECIAL DEV FUNCTIONS
000070  000000G                .WORD   EMTXIT          ; 33  SET EXTRA SWAP ADDRESSES
000072  005344'                .WORD   GVAL            ; 34  .GVAL
000074  002736'                .WORD   SCCA            ; 35  .SCCA
000076  000750'                .WORD   EMTPLX          ; 36  .CRRG, .CRAW, .MAP, etc.
000100  000000G                .WORD   BADEMT          ; 37  Multi-terminal support
000102  004370'                .WORD   SDTTM           ; 40  .SDTTM
000104  007202'                .WORD   ESPCPS          ; 41  .SPCPS
000106  000000G                .WORD   SFDATE          ; 42  .SFDAT -- Set file date
000110  000000G                .WORD   SFPROT          ; 43  .FPROT -- Set file protection
000112  000000G                .WORD   EMTXIT          ; 44  UNKNOWN!!!
000114  004142'                .WORD   CLOSZ           ; 45  .CLOSZ
        000046          E375MX  =       <.-EMTLST>/2
                                ;
                                ;  EMT 374 FUNCTIONS
000116  004236'         EMT374: .WORD   DOWAIT          ;  0  WAIT
000120  000000G                .WORD   RTSPND          ;  1  SUSPEND THIS JOB
000122  000000G                .WORD   RTRSUM          ;  2  RESUME THIS JOB
000124  004014'                .WORD   PURGE           ;  3  PURGE A CHANNEL
000126  002410'                .WORD   SERR            ;  4  SET SOFT ERRORS
000130  002432'                .WORD   HERR            ;  5  SET HARD ERRORS
000132  000000G                .WORD   CLOSE           ;  6  CLOSE
000134  000000G                .WORD   EMTXIT          ;  7  TEST & LOCK USR
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                         ;  EMT'S DEFINED BY TSX.
     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   XHISET          ; 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   CAPOPN          ; 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   DETEMT          ; 132 Detached job control emt
     93 000234  011752'                 .WORD   GETBRK          ; 133 Establish break sentinal 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

```
                                ;-----------------------------------------------------------
                                ;  BRANCH VECTOR FOR EMTS IN THE RANGE 340 TO 357
                                ;
000320  000000G      LST16:    .WORD   TTYIN           ; 340  TTYIN
000322  000000G                .WORD   TTYOUT          ; 341  TTYOUT
000324  000000G                .WORD   DSTAT           ; 342  DSTATUS
000326  000000G                .WORD   FETCH           ; 343  FETCH/REL
000330  000000G                .WORD   CSIGEN          ; 344  CSIGEN
000332  000000G                .WORD   CSISPC          ; 345  CSISPC
000334  000000G                .WORD   EMTXIT          ; 346  LOCK USR
000336  000000G                .WORD   EMTXIT          ; 347  RELEASE USR
000340  006236'                .WORD   EXIT            ; 350  EXIT
000342  000000G                .WORD   PRINT           ; 351  PRINT
000344  003420'                .WORD   SRESET          ; 352  SRESET
000346  003046'                .WORD   QSET            ; 353  QSET
000350  003060'                .WORD   SETTOP          ; 354  SETTOP
000352  003370'                .WORD   RCTRLO          ; 355  RCTRLO
000354  007150'                .WORD   ASTXIT          ; 356  Completion routine exit
000356  003412'                .WORD   HRESET          ; 357  HRESET
```

```
                                        .SBTTL  EMT entry processing
                                ;-------------------------------------------------------------------------
                                ;  Complete EMT entry processing and branch off to routine to do the
                                ;  actual processing of the EMT.
                                ;
                                ;  At this point the EMT function code, channel number and arguments have
                                ;  been moved to EMTBLK which can be directly addressed from kernel mode
                                ;  within the EMT processing routines.
                                ;
                                ;  If EMT tracing has been requested by SET EMT TRACE,
                                ;  print info about this EMT.
                                ;
000360  113701  000000G       DOEMT:  MOVB    CORUSR,R1       ;GET CURRENT JOB INDEX NUMBER
000364  032761  000000G 000000G       BIT     #$EMTTR,LSW6(R1);IS EMT TRACING WANTED?
000372  001406                       BEQ     2$              ;BR IF NOT
000374  032737  000000G 000000G       BIT     #PO$DBG,PRIVCO  ;Are we authorized to do EMT tracing?
000402  001402                       BEQ     2$              ;Br if not
000404  004737  014010'               CALL    EMTTRC          ;TRACE THE EMT
                                ;
                                ;  Make sure the EMT function code is ok.
                                ;
000410  113705  000001G       2$:     MOVB    EMTBLK+1,R5     ;GET FUNCTION CODE
000414  020527  000147               CMP     R5,#EMTMAX      ;FUNCTION CODE OK?
000420  103402                       BLO     5$              ;BR IF OK
000422  000137  000000G               JMP     BADEMT          ;INVALID EMT FUNCTION CODE
                                ;
                                ;  Make sure channel number is ok.
                                ;
000426  120527  000122       5$:     CMPB    R5,#<140+<RT11EX-100>> ;IS THIS TSX REAL-TIME EMT?
000432  001443                       BEQ     4$              ;BR IF YES -- ALLOW ANY CHANNEL NUMBER
000434  005003                       CLR     R3
000436  153703  000000G              BISB    EMTBLK,R3       ;GET CHANNEL # WITHOUT SIGN EXTENSION
000442  012700  000000G              MOV     #NLCHN,R0       ;MAX # CHANNELS FOR EACH JOB
000446  032737  000000G 000000G       BIT     #UMODE,EMTPS    ;WAS EMT DONE IN USER OR KERNEL MODE?
000454  001406                       BEQ     3$              ;BR IF KERNEL MODE
000456  032761  000000G 000000G       BIT     #$INKMN,LSW4(R1);IS KMON RUNNING NOW?
000464  001002                       BNE     3$              ;BR IF YES -- ALLOW ACCESS TO ALL CHANNELS
000466  012700  000000G              MOV     #NUCHN,R0       ;USER PROGRAM -- RESTRICT TO CHANS 0-15.
000472  020300               3$:     CMP     R3,R0           ;IS THIS CHANNEL # TOO BIG?
000474  103034                       BHIS    CHNERR          ;BR IF YES
                                ;  Save address of CSW for channel.
000476  010337  000000G              MOV     R3,CHNNUM       ;SAVE USER'S CHANNEL NUMBER
000502  013700  000000G              MOV     CXTRMN,R0       ;GET ADDRESS OF SIMULATED RMON
000506  062700  000000G              ADD     #R$CHN,R0       ;POINT TO 1ST CHANNEL AREA
000512  020327  000021               CMP     R3,#21          ;IS THIS CHANNEL IN EXTENDED CHANNEL AREA?
000516  103404                       BLO     1$              ;BR IF NOT
000520  162703  000021               SUB     #21,R3          ;GET CHANNEL # IN EXTENDED AREA
000524  062700  000000C              ADD     #R$XCHN-R$CHN,R0;POINT TO EXTENDED CHANNEL AREA
000530  070327  000000G       1$:     MUL     #CHNSIZ,R3      ;MULTIPLY BY # BYTES PER CHANNEL
000534  060003                       ADD     R0,R3           ;GET ABSOLUTE ADDRESS OF CHANNEL AREA
000536  010337  000000G              MOV     R3,CHNADR       ;SAVE CURRENT CHANNEL ADDRESS
                                ;
                                ;  Enter EMT processing routine.
                                ;  R1 contains user index number.
                                ;  UR0 contains contents of user's R0, put value here to return in R0.
                                ;
000542  105037  000000G       4$:     CLRB    CLZERR          ;Clear .CLOSZ error cell
```

EMT entry processing

```
      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
```

```
                                        .SBTTL  Shared file control EMT's
                                ;-------------------------------------------------------
                                ;  PROCESS THE .CLKOPN EMT WHICH IS USED TO OPEN A CHANNEL
                                ;  TO A SHARED FILE.
                                ;
000576  004737  000702'         CLKOPN: CALL    CHKSF           ;See if shared file support is available
000602  000177  000000G                 JMP     @DOCOPN         ;Do the open
                                ;
                                ;-------------------------------------------------------------------------
                                ;  Open a shared file with access protection.
                                ;
000606  004737  000702'         CAPOPN: CALL    CHKSF           ;See if shared file support is available
000612  000177  000000G                 JMP     @DOOPAP         ;Open the channel
                                ;
                                ;-------------------------------------------------------
                                ;  PROCESS THE .CUNLK EMT WHICH IS USED TO UNLOCK
                                ;  A SHARED FILE CHANNEL.
                                ;
000616  004737  000702'         CUNLK:  CALL    CHKSF           ;See if shared file support is available
000622  000177  000000G                 JMP     @DOCULK         ;Unlock all blocks on channel
                                ;
                                ;-------------------------------------------------
                                ;  PROCESS THE .RLOCK EMT WHICH IS USED TO LOCK A
                                ;  SPECIFIC BLOCK.
                                ;
000626  004737  000702'         RLOCK:  CALL    CHKSF           ;See if shared file support is available
000632  000177  000000G                 JMP     @DORLK          ;Lock the block
                                ;
                                ;-------------------------------------------------
                                ;  PROCESS THE .TRYLK EMT WHICH IS USED TO TRY TO LOCK
                                ;  A SPECIFIC BLOCK.
                                ;
000636  004737  000702'         TRYLK:  CALL    CHKSF           ;See if shared file support is available
000642  000177  000000G                 JMP     @DOTLK          ;Try to lock the block
                                ;
                                ;-----------------------------------------------------------
                                ;  UNLOCK A SPECIFIC SHARED FILE BLOCK.
                                ;
000646  004737  000702'         ULK1BK: CALL    CHKSF           ;See if shared file support is available
000652  000177  000000G                 JMP     @DOULK1         ;Unlock a block
                                ;
                                ;-----------------------------------------------------------
                                ;  CHECK FOR WRITES TO A SHARED FILE BY OTHER USERS.
                                ;
000656  004737  000702'         CSFACT: CALL    CHKSF           ;See if shared file support is available
000662  000177  000000G                 JMP     @DOSFCK         ;Check for file modification
                                ;
                                ;-----------------------------------------------------------
                                ;  SUSPEND A SHARED FILE CHANNEL
                                ;  THIS HAS THE SAME EFFECT ON THE SHARED FILE LOGIC AS DOING
                                ;  A .SAVESTATUS BUT THE CHANNEL IS UNAFFECTED.
                                ;
000666  004737  000702'         SFSPND: CALL    CHKSF           ;See if shared file support is available
000672  004777  000000G                 CALL    @SFSVST         ;TELL SHARED FILE LOGIC
000676  000137  000000G                 JMP     EMTXIT
                                ;
                                ;-------------------------------------------------------------------------
```

```
                                        ;  Determine if shared file support is genned into the system and the
                                        ;  user is authorized to use it.  If not, return error code 0 for the EMT.
                                        ;
000702  005737  000000G          CHKSF:  TST     VMXSF           ;Is shared file support genned in?
000706  001405                          BEQ     10$             ;Br if not
000710  032737  000000G 000000G          BIT     #P2$RLK,PRIVC2  ;Are we privileged to access shared files?
000716  001401                          BEQ     10$             ;Br if not
000720  000207                          RETURN
                                        ;
                                        ;  We cannot use shared file support
                                        ;
000722  005000                  10$:    CLR     R0              ;Return error code 0
000724  000137  000000G                JMP     SETERR
                                        ;-------------------------------------------------------------------------
                                        ;  Display window control EMT's
                                        ;
000730  005737  000000G          CKWIN:  TST     VMXWIN          ;Is window support genned in?
000734  001003                          BNE     1$              ;Br if yes
000736  005000                          CLR     R0              ;Return error code 0
000740  000137  000000G                JMP     SETERR
000744  000137  000000G          1$:     JMP     EMTWIN          ;Enter window control overlay
```

```
                                        .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.
                                ;
 000750  005737  000000G       EMTPLX: TST     VPLAS           ;Was PLAS support generated into system?
 000754  001402                        BEQ     9$              ;Br if not
 000756  000137  000000G               JMP     EMTPLS          ;Enter TSPLAS overlay
                                ;
                                ;  Error: PLAS support not generated into system
                                ;
 000762  012700  000007        9$:     MOV     #7,R0           ;Return error code 7
 000766  005037  000000G               CLR     UR0             ;Return 0 in R0
 000772  000137  000000G               JMP     SETERR
```

```
                                        .SBTTL  Message communication EMT's
                                ;-------------------------------------------------------------------
                                ;  PROCESS THE .MSEND EMT WHICH QUEUES A MESSAGE FOR
                                ;  ANOTHER USER.
                                ;
000776  004737  001060'        XMSEND: CALL    MSGPRV          ;Do we have access to message facility?
001002  012700  000000G                MOV     #EMTBLK,R0      ;Point to EMT arg block
001006                                  OCALL   MSEND           ;SEND THE MESSAGE
001014  000417                          BR      CHKERR          ;CHECK FOR ERRORS
                                ;
                                ;-------------------------------------------------------
                                ;  PROCESS THE .MSGCK EMT WHICH IS USED TO CHECK FOR
                                ;  A PENDING MESSAGE.
                                ;
001016  004737  001060'        XMSGCK: CALL    MSGPRV          ;Do we have access to message facility?
001022  012700  000000G                MOV     #EMTBLK,R0      ;POINT TO EMT ARG BLOCK
001026                                  OCALL   MSGCK           ;CHECK FOR MESSAGE
001034  000407                          BR      CHKERR          ;GO CHECK FOR ERRORS
                                ;
                                ;-------------------------------------------------------
                                ;  PROCESS THE .MSGWT EMT WHICH IS USED TO WAIT FOR
                                ;  A MESSAGE FROM ANOTHER USER.
                                ;
001036  004737  001060'        XMSGWT: CALL    MSGPRV          ;Do we have access to message facility?
001042  012700  000000G                MOV     #EMTBLK,R0      ;POINT TO EMT ARG BLOCK
001046                                  OCALL   MSGWT           ;WAIT FOR THE MESSAGE
001054  000137  000000G        CHKERR: JMP     EMTXIT

                                ;-------------------------------------------------------------------
                                ;  See if the message communication facility is genned into the system
                                ;  and we are authorized to use it.  If not, return EMT error code 0.
                                ;
001060  005737  000000G        MSGPRV: TST     VMAXMC          ;Is message facility genned into system?
001064  001405                          BEQ     10$             ;Br if not
001066  032737  000000G 000000G        BIT     #P2$MSG,PRIVC2  ;Are we authorized to use message facility?
001074  001401                          BEQ     10$             ;Br if not
001076  000207                          RETURN
                                ;
                                ;  We cannot access message facility
                                ;
001100  005000                10$:     CLR     R0              ;Return error code 0
001102  000137  000000G                JMP     SETERR
```

```
                                        .SBTTL  .WRITE
                                ;-------------------------------------------------------------------------------
                                ;  Process a .WRITE EMT.
                                ;
001106  004737  012172'         WRITE:  CALL    CKIOST          ;SEE IF SCHEDULER WANTS TO STOP I/O
001112  013703  000000G                 MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL
001116  032763  000000G 000000G         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
001124  001004                         BNE     6$              ;BR IF YES
001126  012700  000002                 MOV     #2,R0           ;CHANNEL NOT OPEN
001132  000137  000000G                 JMP     SETERR
                                ;
                                ;  Check to make sure the buffer is on a word boundary
                                ;
001136  032737  000001  000004G 6$:     BIT     #1,EMTBLK+4     ;Is buffer address even?
001144  001414                         BEQ     15$             ;Br if yes
001146  016302  000000G                 MOV     C.CSW(R3),R2    ;Get CSW
001152  042702  177701                 BIC     #^C76,R2        ;Get device table index
001156  032762  000000G 000000G         BIT     #DX$EBA,DVFLAG(R2);Does this device require even buffer addr?
001164  001404                         BEQ     15$             ;Br if not
001166  012700  177766                 MOV     #-12,R0         ;Get abort code
001172  000137  000000G                 JMP     SETERR          ;Abort the operation
                                ;
                                ;  Make sure word count is ok
                                ;
001176  005737  000006G         15$:    TST     EMTBLK+6        ;Is word count negative (>32727.)?
001202  001405                         BEQ     17$             ;Br if word count zero
001204  100016                         BPL     18$             ;Branch if valid size (positive)
001206  012700  177766                 MOV     #-12,R0         ;Get abort code
001212  000137  000000G                 JMP     SETERR          ;Abort the operation
                                ;
                                ;  Word count is zero ==> Seek.
                                ;  Treat seek like nop but call completion routine if requested.
                                ;
001216  005037  000000G         17$:    CLR     UR0             ;Say zero words read
001222  023727  000010G 000001         CMP     EMTBLK+10,#1    ;Need to run completion routine?
001230  101402                         BLOS    19$             ;Br if not
001232  004737  000000G                 CALL    FAKCMP          ;Call completion routine
001236  000137  000000G         19$:    JMP     EMTXIT          ;Finished with read
                                ;
                                ;  See if we are writing to TT:
                                ;
001242  004737  012146'         18$:    CALL    CHKTT           ;IS CHANNEL OPENED TO "TT:"?
001246  103402                         BCS     10$             ;BR IF NOT WRITING TO TT
001250  000137  000000G                 JMP     WRITTT          ;ENTER THE TT WRITE ROUTINE IN TSTTY
                                ;
                                ;  See if we are writing to a spooled device.
                                ;
001254  032763  000000G 000000G 10$:    BIT     #CS$SPL,C.CSW(R3);IS THIS A SPOOLED DEVICE?
001262  001171                         BNE     WRTSPL          ;BR IF IT IS A SPOOLED DEVICE
                                ;
                                ;  See if we are attempting to write to a read-only device.
                                ;
001264  016302  000000G                 MOV     C.CSW(R3),R2    ;GET CSW
001270  032702  000000G                 BIT     #CS$RON,R2      ;IS USER AUTHORIZED FOR READ-ONLY ACCESS?
001274  001404                         BEQ     9$              ;BR IF NOT READ-ONLY
001276  012700  177763                 MOV     #-15,R0         ;SET ABORT ERROR CODE
001302  000137  000000G                 JMP     SETERR          ;ABORT EMT
```

```
     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$:    CLR     URO             ;SAY 0 WORDS WRITTEN
     62 001326  005000                         CLR     R0              ;RETURN ERROR CODE OF 0
     63 001330  000137  000000G                JMP     SETERR
     64                                    ;
     65                                    ;  See if error status is pending in CSW for the channel
     66                                    ;
     67 001334  032763  000000G 000000G 5$:    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$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
     70 001352  012700  000001                 MOV     #1,R0           ;RETURN ERROR CODE OF 1
     71 001356  000137  000000G                JMP     SETERR
     72                                    ;
     73                                    ;  Check if this is an I/O operation by IND
     74                                    ;
     75 001362  032761  000000G 000000G 20$:   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   CLRDIR          ;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     R0,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$:    MOV     R0,C.USED(R3)   ;Remember high-water-mark within file
    113 001520  010061  000000C                MOV     R0,Q.ICSW+C.USED(R1);Store into chan blk in Q element too
    114 001524  005461  000000G          2$:     NEG     Q.WCNT(R1)      ;Negative word count ==> write operation
```

```
                                        ;
                                        ;  If we shortened write request because we hit end of file, return error
                                        ;  code of 0 but go ahead and do the write.
                                        ;
001530  023737  000000G 000006G         CMP     URO,EMTBLK+6    ;DID WE SHORTEN WRITE REQUEST?
001536  001403                          BEQ     7$              ;BR IF NOT
001540  052737  000000G 000000G         BIS     #CFLAG,EMTPS    ;SET C-FLAG IN PS
                                        ;
                                        ;  Set address of write completion routine.
                                        ;
001546  013700  000010G         7$:     MOV     EMTBLK+10,R0    ;GET ADDRESS OF COMPLETION ROUTINE
001552  004737  013000'                  CALL    SETCR           ;STORE ADDR OF COMPL ROUTINE INTO QUEUE ELEM
001556  103431                          BCS     3$              ;BR IF FATAL ERROR
                                        ;
                                        ;  We have built the I/O queue element.
                                        ;  Initiate the I/O operation.
                                        ;
001560  004737  000000G                 CALL    QIO             ;QUEUE AN I/O REQUEST
                                        ;
                                        ;  See if we are writing to a shared file
                                        ;
001564  005737  000000G                 TST     JCDB            ;Does job have any shared files open?
001570  001402                          BEQ     12$             ;Br if not
001572  004777  000000G                 CALL    @SFWRIT         ;Check for writes to shared files
                                        ;
                                        ;  See if we should wait for the I/O operation to finish.
                                        ;
001576  005737  000010G         12$:    TST     EMTBLK+10       ;IS THIS AN .WRITW?
001602  001015                          BNE     1$              ;BR IF NOT
001604  004737  000000G                 CALL    IOWAIT          ;WAIT FOR I/O TO FINISH
                                        ;
                                        ;  See if a hardware error occured on the write
                                        ;
001610  032763  000000G 000000G         BIT     #CS$ERR,C.CSW(R3);DID A WRITE ERROR OCCUR?
001616  001407                          BEQ     1$              ;BR IF NOT
001620  042763  000000G 000000G         BIC     #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
001626  012700  000001                  MOV     #1,R0           ;RETURN ERROR CODE OF 1
001632  000137  000000G                 JMP     SETERR
                                        ;
                                        ;  Finished the EMT.
                                        ;
001636  000137  000000G         1$:     JMP     EMTXIT          ;EXIT FROM EMT
                                        ;
                                        ;  Some fatal error occured which causes us to have to abort the operation.
                                        ;
001642  000137  002374'         3$:     JMP     IOABRT          ;ABORT THE I/O REQUEST
                                        ;
                                        ;  Write to a spooled device.
                                        ;
001646                          WRTSPL: OCALL   SDMOVE          ;WRITE USER'S DATA TO SPOOL FILE
                                        ;
                                        ;  See if we need to queue a completion routine for this write request.
                                        ;
001654  023727  000010G 000001  WRTCPL: CMP     EMTBLK+10,#1    ;WAS A COMPLETION ROUTINE SPECIFIED?
001662  101402                          BLOS    1$              ;BR IF NOT
001664  004737  000000G                 CALL    FAKCMP          ;QUEUE A COMPLETION ROUTINE REQUEST
                                        ;
```

```
    172                                 ;  Finished spooled write
    173                                 ;
    174 001670  000137  000000G         1$:     JMP     EMTXIT
```

```
                                        .SBTTL  .READ
                                ;-------------------------------------------------------------------------
                                ;  Process a .READ EMT.
                                ;
001674  004737  012172'         READ:   CALL    CKIOST          ;SEE IF SCHEDULER WANTS TO STOP I/O
001700  013703  000000G                 MOV     CHNADR,R3       ;GET ADDRESS OF THE CHANNEL
001704  032763  000000G 000000G         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
001712  001004                          BNE     2$              ;BR IF YES
001714  012700  000002                 MOV     #2,R0           ;RETURN ERROR CODE
001720  000137  000000G                 JMP     SETERR
                                ;
                                ;  Check to make sure the buffer is on a word boundary
                                ;
001724  032737  000001  000004G 2$:     BIT     #1,EMTBLK+4     ;Is buffer address even?
001732  001414                          BEQ     15$             ;Br if yes
001734  016302  000000G                 MOV     C.CSW(R3),R2    ;Get CSW
001740  042702  177701                 BIC     #^C76,R2        ;Get device table index
001744  032762  000000G 000000G         BIT     #DX$EBA,DVFLAG(R2);Does this device require even buffer addr?
001752  001404                          BEQ     15$             ;Br if not
001754  012700  177766                 MOV     #-12,R0         ;Get abort code
001760  000137  000000G                 JMP     SETERR          ;Abort the operation
                                ;
                                ;  See if the read request is so large that
                                ;  a handler might interpret it as a write
                                ;
001764  005737  000006G         15$:    TST     EMTBLK+6        ;Is word count negative (>32727.)?
001770  001405                          BEQ     17$             ;Br if word count zero
001772  100016                          BPL     14$             ;Branch if valid size (positive)
001774  012700  177766                 MOV     #-12,R0         ;Get abort code
002000  000137  000000G                 JMP     SETERR          ;Abort the operation
                                ;
                                ;  Word count is zero ==> Seek.
                                ;  Treat seek like nop but call completion routine if requested.
                                ;
002004  005037  000000G         17$:    CLR     UR0             ;Say zero words read
002010  023727  000010G 000001          CMP     EMTBLK+10,#1    ;Need to run completion routine?
002016  101402                          BLOS    19$             ;Br if not
002020  004737  000000G                 CALL    FAKCMP          ;Call completion routine
002024  000137  000000G         19$:    JMP     EMTXIT          ;Finished with read
                                ;
                                ;  See if we are reading from TT:
                                ;
002030  004737  012146'         14$:    CALL    CHKTT           ;IS CHANNEL OPENED TO "TT:"?
002034  103402                          BCS     10$             ;BR IF NOT TT READ
002036  000137  000000G                 JMP     TTREAD          ;TT READ FROM MAPPED TSTTY
                                ;
                                ;  We are reading from some device other than TT.
                                ;
                                ;  See if we are attempting to read from a write-only device.
                                ;
002042  016300  000000G         10$:    MOV     C.CSW(R3),R0    ;GET CSW
                                ;       BIT     #CS$SPL,R0      ;IS THIS A SPOOLED DEVICE?
                                ;       BNE     4$              ;BR IF SPOOLED DEVICE
002046  042700  177701                  BIC     #^C76,R0        ;GET DEVICE TABLE INDEX FOR DEVICE
002052  032760  000000G 000000G         BIT     #DS$WON,DVSTAT(R0);IS THIS A WRITE-ONLY DEVICE?
002060  001405                          BEQ     7$              ;BR IF NOT WRITE-ONLY
002062  005037  000000G         4$:     CLR     UR0             ;SAY 0 WORDS READ
```

```
                                        CLR     R0              ;RETURN ERROR CODE 0
                                        JMP     SETERR
                                ;
                                ;  See if error flag is pending in CSW for this channel
                                ;
                        7$:     BIT     #CS$ERR,C.CSW(R3);DID A HARDWARE ERROR OCCUR ON READ?
                                BEQ     16$             ;BR IF NOT
                                BIC     #CS$ERR,C.CSW(R3);ACKNOWLEDGE ERROR STATUS
                                MOV     #1,R0           ;RETURN ERROR CODE OF 1
                                JMP     SETERR
                                ;
                                ;  See if end-of-file status is pending in CSW for this channel
                                ;
                        16$:    BIT     #CS$EOF,C.CSW(R3);DID DEVICE REPORT END OF FILE?
                                BEQ     18$             ;BR IF NOT
                                BIC     #CS$EOF,C.CSW(R3);ACKNOWLEDGE EOF STATUS
                                CLR     UR0             ;SAY 0 WORDS READ
                                CLR     R0              ;RETURN ERROR CODE OF 0
                                JMP     SETERR
                                ;
                                ;  See if data being read is currently in the data cache
                                ;
                        18$:    TST     JCDB            ;Any shared file channels open for job?
                                BEQ     8$              ;Br if not
                                TST     NUMDCD          ;Are we doing shared-file data caching?
                                BEQ     8$              ;Br if not
                                CALL    @DCRD1          ;SEE IF DATA IS IN DATA CACHE
                                BCS     8$              ;BR IF DATA WAS NOT IN THE CACHE
                                MOV     EMTBLK+6,UR0    ;RETURN WORD COUNT READ IN USER'S R0
                                BR      WRTCPL          ;FINISH UP AND EXIT
                                ;
                                ;  Check for I/O operations by IND
                                ;
                        8$:     BIT     #$INDRN,LSW5(R1);IS IND RUNNING?
                                BEQ     11$             ;BR IF NOT
                                CALL    INDIO           ;MAP I/O OPERATIONS FOR IND
                                ;
                                ;  Build I/O queue element for I/O operation.
                                ;
                        11$:    CALL    GETQ            ;GET A FREE I/O QUEUE ELEMENT
                                ;  R1 now points to a free I/O queue element.
                                CALL    SETQ            ;DO COMMON QUEUE SETUP
                                BCS     IOABRT          ;BR IF FATAL ERROR DETECTED
                                MOV     EMTBLK+10,R0    ;GET COMPLETION ROUTINE ADDRESS
                                CALL    SETCR           ;STORE COMPL RTN ADDR IN I/O QUEUE
                                BCC     3$              ;BR IF ADDRESS OK
                                MOVB    #-12,INTERR     ;SET ERROR CODE
                                BR      IOABRT          ;ABORT OPERATION
                                ;
                                ;  We have build an I/O queue entry for read request.
                                ;  Initiate the I/O operation.
                                ;
                        3$:     CALL    QIO             ;QUEUE THE REQUEST
                                ;
                                ;  See if we need to update data cache
                                ;
                                TST     JCDB            ;Any shared file channels for job?
```

```
    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 occured 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$ERR,C.CSW(R3);ACKNOWLEDGE ERROR STATUS
    131 002330  012700  000001                 MOV     #1,R0           ;RETURN ERROR CODE OF 1
    132 002334  000137  000000G                 JMP     SETERR
    133                                 ;
    134                                 ;  See if the device reported end-of-file.
    135                                 ;
    136 002340  032763  000000G 000000G 6$:     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$EOF,C.CSW(R3);ACKNOWLEDGE EOF STATUS
    139 002356  005037  000000G                 CLR     URO             ;SAY 0 WORDS READ
    140 002362  005000                        CLR     R0              ;RETURN ERROR CODE OF 0
    141 002364  000137  000000G                 JMP     SETERR
    142                                 ;
    143                                 ;  Finished operation.
    144                                 ;
    145 002370  000137  000000G         1$:     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                 MOVB    INTERR,R0       ;GET ERROR CODE
    152 002404  000137  000000G                 JMP     SETERR          ;ERROR RETURN
```

```
                                        .SBTTL  .SERR
                                ;-------------------------------------------------
                                ;  PROCESS THE .SERR EMT
002410  113700  000000G          SERR:   MOVB    SERFLG,R0       ;GET PREVIOUS STATE
002414  010037  000000G                  MOV     R0,URO          ;AND RETURN TO USER
002420  112737  000001  000000G        MOVB    #1,SERFLG       ;REMEMBER USER ISSUED SERR
002426  000137  000000G                  JMP     EMTXIT

                                        .SBTTL  .HERR
                                ;-------------------------------------------------
                                ;  PROCESS THE .HERR EMT
002432  113700  000000G          HERR:   MOVB    SERFLG,R0       ;GET PREVIOUS STATE
002436  010037  000000G                  MOV     R0,URO          ;AND RETURN TO USER
002442  105037  000000G                  CLRB    SERFLG          ;TURN OFF .SERR CONTROL
002446  000137  000000G                  JMP     EMTXIT
```

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

```
                                        ;
                                        ;  Initiate the I/O operation.
                                        ;
002636  004737  000000G                       CALL    QIO             ;QUEUE THE I/O REQUEST
                                        ;
                                        ;  See if we should wait for the I/O operation to finish.
                                        ;
002642  005737  0000126                       TST     EMTBLK+12       ;IS WAIT REQUESTED?
002646  001002                                BNE     13$             ;BR IF NOT
                                        ;
                                        ;  Wait for I/O to finish
                                        ;
002650  004737  000000G                       CALL    IOWAIT          ;WAIT FOR I/O TO FINISH
                                        ;
                                        ;  See if a hardware error or end-of-file occured.
                                        ;
002654  032763  000000G 000000G 13$:    BIT     #CS$ERR,C.CSW(R3);DID A HARDWARE ERROR OCCUR?
002662  001407                          BEQ     6$              ;BR IF NOT
002664  042763  000000G 000000G         BIC     #CS$ERR,C.CSW(R3);CLEAR ERROR FLAG IN CSW
002672  012700  000001                  MOV     #1,R0           ;REPORT ERROR
002676  000137  000000G                 JMP     SETERR
002702  032763  000000G 000000G 6$:     BIT     #CS$EOF,C.CSW(R3);DID WE HIT END-OF-FILE?
002710  001406                          BEQ     5$              ;BR IF NOT
002712  042763  000000G 000000G         BIC     #CS$EOF,C.CSW(R3);CLEAR END-OF-FILE FLAG IN CSW
002720  005000                          CLR     R0              ;RETURN ERROR CODE OF 0
002722  000137  000000G                 JMP     SETERR
                                        ;
                                        ;  Normal completion of .SPFUN
                                        ;
002726  000137  000000G         5$:     JMP     EMTXIT          ;FINISHED EMT
                                        ;
                                        ;  Some fatal error occured.  Abort request.
                                        ;
002732  000137  002374'         9$:     JMP     IOABRT          ;ABORT I/O REQUEST
```

```
                                        .SBTTL  .SCCA
                                ;-------------------------------------------------
                                ;  PROCESS THE .SCCA EMT
                                ;
002736  016137  000000G 000000G SCCA:   MOV     LSCCA(R1),URO   ;RETURN OLD SCCA RTN ADDRESS TO USER IN RO
002744  013700  000002G                 MOV     EMTBLK+2,R0     ;GET ADDRESS OF SCCA ROUTINE
002750  004737  000000G                 CALL    VALADW          ;VALIDATE THE ADDRESS
002754  010061  000000G                 MOV     R0,LSCCA(R1)    ;SAVE ADDRESS OF SCCA ROUTINE FOR JOB
002760  000406                          BR      JNOP

                                        .SBTTL  .TRPSET
                                ;-------------------------------------------------
                                ;  PROCESS THE .TRPSET EMT
                                ;
002762  013700  000002G         TRPSET: MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S TRAP ROUTINE
002766  004737  000000G                 CALL    VALADW          ;VALIDATE THE ADDRESS
002772  010037  000000G                 MOV     R0,UTRPAD       ;SET TRAP ROUTINE ADDRESS FOR JOB
002776  000137  000000G         JNOP:   JMP     EMTXIT

                                        .SBTTL  .SFPA
                                ;-------------------------------------------------
                                ;  PROCESS THE .SFPA EMT
                                ;
003002  013700  000002G         SFPA:   MOV     EMTBLK+2,R0     ;GET ADDRESS OF TRAP ROUTINE
003006  004737  000000G                 CALL    VALADB          ;VALIDATE IT (NOTE: ADDRESS MAY = 1)
003012  010037  000000G                 MOV     R0,UFPTRP       ;SET ADDRESS OF TRAP ROUTINE
003016  005700                          TST     R0              ;IS USER GOING TO USE FPU UNIT?
003020  001407                          BEQ     2$              ;BR IF NOT
003022  032737  000000G 000000G        BIT     #CW$FPU,CONFIG  ;DOES SYSTEM HAVE A FPU UNIT?
003030  001403                          BEQ     2$              ;BR IF NOT
003032  110137  000000G                 MOVB    R1,FPUUSE       ;REMEMBER USER IS ACCESSING FPU
003036  000402                          BR      3$
003040  105037  000000G         2$:     CLRB    FPUUSE          ;SAY FPU NOT IN USE
003044  000754                  3$:     BR      JNOP

                                        .SBTTL  .QSET
                                ;-------------------------------------------------------------------
                                ;  Process the .QSET EMT.
                                ;  This is essentially a NOP under TSX-Plus.
                                ;  The address beyond the end of the queue buffer is returned in R0.
                                ;
003046  013737  000004G 000000G QSET:   MOV     EMTBLK+4,UR0    ;Return address of Q buffer in R0
003054  000137  000000G                 JMP     EMTXIT          ;Finished with EMT

                                        .SBTTL  .SETTOP
                                ;-------------------------------------------------------------------
                                ;  PROCESS THE .SETTOP EMT
                                ;
003060  013700  000000G         SETTOP: MOV     UR0,R0          ;GET USER'S REQUESTED TOP ADDRESS
003064  005200                  SETTP1: INC     R0              ;ROUND ADDRESS UP TO WORD IF IT'S ODD
003066  001001                          BNE     2$              ;BR IF DID NOT OVERFLOW
003070  005300                          DEC     R0              ;IF OVERFLOWED, FORCE BACK TO 177777
003072  042700  000001         2$:     BIC     #1,R0           ;FORCE ADDRESS TO BE EVEN
003076  020037  000000G                 CMP     R0,ODTBAS       ;DOES REQUEST EXCEED JOB'S LIMIT?
003102  103404                          BLO     1$              ;BR IF NOT
003104  013700  000000G                 MOV     ODTBAS,R0       ;GET MAX AMT JOB IS ALLOWED TO USE
003110  162700  000002                  SUB     #2,R0           ;POINT TO WORD BELOW TOP LIMIT
```

```
     58 003114  010037  000000G          1$:     MOV     R0,URO          ;RETURN TOP ADDRESS IN USER'S R0
     59 003120  010046                           MOV     R0,-(SP)        ;AND IN LOCATION 50
     60 003122  106637  000050                   MTPD    @#50
     61 003126  000137  000000G                  JMP     EMTXIT          ;FINISHED
     62                                     ;
     63                                     ;-----------------------------------------------------------------------
     64                                     ;  The TSX-Plus SETSIZ emt is used to change the memory space allocated
     65                                     ;  to a job.
     66                                     ;
     67                                     ;  The form of the EMT argument block is:
     68                                     ;
     69                                     ;       .BYTE   0,141
     70                                     ;       .WORD   .top-addr       ;Requested top address for job
     71                                     ;
     72 003132  013700  000002G          SETSIZ: MOV     EMTBLK+2,R0     ;GET REQUESTED TOP OF JOB ADDRESS
     73 003136  105737  000000G                  TSTB    VSWPFL          ;IS THIS A NON-SWAPPING SYSTEM?
     74 003142  001750                           BEQ     SETTP1          ;IF NON-SWAPPING THEN CAN'T CHANGE SIZE
     75 003144  120137  000000G                  CMPB    R1,EXCJOB       ;DOES JOB HAVE EXCLUSIVE ACCESS TO SYSTEM?
     76 003150  001745                           BEQ     SETTP1          ;IF YES THEN CAN'T CHANGE SIZE
     77 003152  013702  000000G                  MOV     MXJADR,R2       ;GET ADDRESS ABOVE TOP OF AVAILABLE SPACE
     78 003156  162702  000002                   SUB     #2,R2           ;GET ADDRESS OF TOP AVAILABLE WORD
     79 003162  020002                           CMP     R0,R2           ;IS REQUEST ABOVE TOP ALLOWED?
     80 003164  101401                           BLOS    1$              ;BR IF NOT
     81 003166  010200                           MOV     R2,R0           ;CONSTRAIN TO TOP ALLOWED ADDRESS
     82 003170  010037  000000G          1$:     MOV     R0,URO          ;RETURN TOP ADDRESS TO USER IN R0
     83 003174  010046                           MOV     R0,-(SP)        ;AND PUT IN USER'S LOCATION 50
     84 003176  106637  000050                   MTPD    @#50
     85 003202  020027  160000                   CMP     R0,#160000      ;IS JOB NOW USING MORE THAN 56KB?
     86 003206  103412                           BLO     2$              ;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,R0           ;GET ADDRESS ABOVE TOP WORD WANTED
     92 003240  010037  000000G                  MOV     R0,ODTBAS       ;CONSTRAIN .SETTOP TO THIS
     93 003244  004737  000000G                  CALL    SUTOP           ;DO MEMORY ALLOCATION FOR JOB
     94 003250  000137  000000G                  JMP     EMTXIT          ;FINISHED
```

```
                                        .SBTTL  .GTJB
                                ;-------------------------------------------------------
                                ;  PROCESS THE .GTJB EMT
                                ;
003254  013700  000002G         GTJB:   MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S BUFFER
003260  004737  000000G                 CALL    VALADW          ;VALIDATE BUFFER ADDRESS
003264  010146                         MOV     R1,-(SP)        ;RETURN JOB #
003266  006216                         ASR     (SP)
003270  106620                         MTPD    (R0)+           ;[WORD 1]
003272  013746  000000G                 MOV     ODTBAS,-(SP)    ;HIGH MEMORY LIMIT
003276  106620                         MTPD    (R0)+           ;[WORD 2]
003300  005046                         CLR     -(SP)           ;LOW MEMORY
003302  106620                         MTPD    (R0)+           ;[WORD 3]
003304  012746  000000C                 MOV     #RMNBAS+R$CHN-2,-(SP);START OF JOB CHANNEL AREA
003310  106620                         MTPD    (R0)+           ;[WORD 4]
003312  005046                         CLR     -(SP)           ;ADDRESS OF JOB'S IMPURE AREA
003314  106620                         MTPD    (R0)+           ;[WORD 5]
003316  010146                         MOV     R1,-(SP)        ;JOB'S TERMINAL NUMBER
003320  006216                         ASR     (SP)
003322  106620                         MTPD    (R0)+           ;[WORD 6]
003324  013746  000000G                 MOV     ODTBAS,-(SP)    ;VIRTUAL HIGH LIMIT
003330  106620                         MTPD    (R0)+           ;[WORD 7]
003332  000137  000000G         GTJJ:   JMP     EMTXIT
```

```
                                        .SBTTL  .CHAIN
                                ;-------------------------------------------------------
                                ;  PROCESS A .CHAIN REQUEST.
                                ;
003336  105237  000000G         DOCIN:  INCB    CINFLG          ;SAY .CHAIN IN PROGRESS
                                ;
                                ;  Move chain data to job context area.
                                ;
003342  012702  000500                  MOV     #500,R2         ;POINT TO START OF CHAIN DATA AREA
003346  012703  000000G                 MOV     #CINDAT,R3      ;POINT TO CHAIN DATA SAVE AREA IN CONTEXT BLK
003352  012704  000140                  MOV     #<<1000-500>/2>,R4 ;GET # WORDS TO MOVE
003356  106522                  1$:     MFPD    (R2)+           ;GET A WORD FROM USER'S CHAIN DATA AREA
003360  012623                          MOV     (SP)+,(R3)+     ;MOVE TO CONTEXT BLOCK AREA
003362  077403                          SOB     R4,1$
003364  004737  000000G                 CALL    STOP            ;EXIT TO KMON TO DO .CHAIN

                                        .SBTTL  .RCTRLO
                                ;-------------------------------------------------------
                                ;  PROCESS THE .RCTRLO EMT
                                ;
003370  032761  000000G 000000G RCTRLO: BIT     #$CFOPN,LSW4(R1);Is input coming from a command file?
        003376                  RCTOCF == .-TSEM2
003376  001003                          BNE     1$              ;Br if yes -- Don't reset ctrl-O
003400  042761  000000G 000000G         BIC     #$CTRLO,LSW3(R1);Reset Ctrl-O flag
003406  000137  000000G         1$:     JMP     EMTXIT
```

```
                                        .SBTTL  .HRESET
                                ;-------------------------------------------------------------
                                ;  Process the .HRESET EMT
                                ;  (Halt all I/O for job and then purge all I/O channels)
                                ;
003412  004737  000000G       HRESET: CALL    IOHALT          ;HALT ALL I/O FOR JOB
003416  000400                        BR      SRESET          ;NOW DO .SRESET
                                        .SBTTL  .SRESET
                                ;-------------------------------------------------------------
                                ;  PROCESS THE .SRESET AND .HRESET EMTS.
                                ;
003420                          SRESET:
                                ;
                                ;  Reset .SPND/.RSUM counter for job
                                ;
003420  005061  000000G               CLR     LSPND(R1)       ;RESET .SPND COUNT FOR JOB
                                ;
                                ;  Cancel any pending .MRKT requests
                                ;
003424  004737  005062'               CALL    CANMKT          ;CANCEL PENDING MARK-TIME REQUESTS
                                ;
                                ;  Cancel any job monitoring requests
                                ;
003430                                OCALL   MONABT          ;Cancel any job monitoring
                                ;
                                ;  Cancel any message-read completion routine requests
                                ;
003436  005737  000000G               TST     VMAXMC          ;Is message support genned in?
003442  001403                        BEQ     4$              ;Br if not
003444                                OCALL   MSGABT          ;Abort any message-read cpl routines
                                ;
                                ;  Purge I/O channels
                                ;
003452  012702  177777G       4$:     MOV     #NUCHN-1,R2     ;GET HIGHEST USER CHANNEL #
003456  020227  000017         1$:     CMP     R2,#17          ;IS THIS CHANNEL 17?
003462  001004                        BNE     2$              ;BR IF NOT
003464  032761  000000G 000000G       BIT     #OVLBIT,LJSW(R1);IS THE PROGRAM OVERLAYED?
003472  001004                        BNE     3$              ;IF IT IS, DON'T CLOSE CHANNEL 17
003474                          2$:     .PURGE  R2              ;PURGE EACH ONE
003504  005302                  3$:     DEC     R2
003506  002363                        BGE     1$
                                ;
                                ;  Undo .CDFN
                                ;
003510  005037  000000G               CLR     UCHAN           ;RESET USER CHANNEL DEFINITIONS
003514  000137  000000G               JMP     EMTXIT
```

```
                                        .SBTTL  .SAVESTATUS
                                ;-------------------------------------------------------------------
                                ;  Save channel status.
                                ;
003520  013703  000000G          SVSTAT: MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
003524  032763  000000G 000000G         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
003532  001417                          BEQ     4$              ;BR IF NOT OPEN NOW
                                ;  Make sure file was opened with a lookup.
003534  032763  000000G 000000G         BIT     #CS$ENT,C.CSW(R3);OPENED WITH LOOKUP OR ENTER?
003542  001404                          BEQ     2$              ;BR IF LOOKUP
003544  012700  000001                  MOV     #1,R0           ;RETURN ERROR CODE 1
003550  000137  000000G                 JMP     SETERR
                                ;
                                ;  Wait for any I/O activity on the channel to finish
                                ;
003554  004737  000000G          2$:    CALL    IOWAIT          ;WAIT FOR I/O TO FINISH
                                ;
                                ;  Tell TSX shared file logic that savestatus is being done
                                ;
003560  005737  000000G                 TST     JCDB            ;Any shared file channels open for job?
003564  001402                          BEQ     4$              ;Br if not
003566  004777  000000G                 CALL    @SFSVST         ;TELL SHARED FILE SYSTEM
                                ;
                                ;  Move channel block to user's area
                                ;
003572  010301                  4$:    MOV     R3,R1           ;SAVE ADDRESS OF CHANNEL AREA
003574  013700  000002G                 MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S SAVE AREA
003600  004737  000000G                 CALL    VALADW          ;VALIDATE THE ADDRESS
003604  012704  000005                  MOV     #5,R4           ;GET # WORDS TO MOVE
003610  012346                  3$:    MOV     (R3)+,-(SP)     ;MOVE DATA TO USER'S AREA
003612  106620                          MTPD    (R0)+
003614  077403                          SOB     R4,3$
                                ;
                                ;  Mark channel as closed
                                ;
003616  032761  000000G 000000G         BIT     #CS$OPN,C.CSW(R1);IS THE CHANNEL OPEN?
003624  001404                          BEQ     5$              ;BR IF CHANNEL IS NOT OPEN
003626  005061  000000G                 CLR     C.CSW(R1)       ;PURGE THE CHANNEL
003632  000137  000000G                 JMP     EMTXIT          ;FINISHED
003636  005000                  5$:    CLR     R0              ;RETURN ERROR CODE OF 0 IF CHANNEL CLOSED
003640  000137  000000G                 JMP     SETERR
```

```
                                        .SBTTL  .REOPEN
                                ;-------------------------------------------------------------------------
                                ;  Reopen a channel using saved status information.
                                ;
003644  013703  000000G         REOPEN: MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
                                ;
                                ;  Make sure channel is not now open.
                                ;
003650  032763  000000G 000000G         BIT     #CS$OPN,C.CSW(R3);IS CHANNEL OPEN NOW?
003656  001403                         BEQ     1$              ;BR IF NOT
003660  005000                         CLR     R0              ;RETURN ERROR CODE OF 0
003662  000137  000000G                JMP     SETERR
                                ;
                                ;  Channel is closed.  Restore status.
                                ;
003666  013700  000002G         1$:     MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S SAVE STATUS BLOCK
003672  004737  000000G                 CALL    VALADW          ;VALIDATE THE ADDRESS
003676  012704  000005                 MOV     #5,R4           ;GET # WORDS TO MOVE
003702  106520                 3$:     MFPD    (R0)+           ;GET A WORD FROM USER'S AREA
003704  012623                         MOV     (SP)+,(R3)+     ;MOVE INTO CHANNEL BLOCK
003706  077403                         SOB     R4,3$           ;MOVE ALL
                                ;
                                ;  Check consistency of savestatus information that we restored.
                                ;
003710  013703  000000G                 MOV     CHNADR,R3       ;POINT TO CHANNEL BLOCK
003714  016300  000000G                 MOV     C.CSW(R3),R0    ;GET CHANNEL STATUS WORD (CSW)
003720  032700  000000G                 BIT     #CS$OPN,R0      ;CHANNEL SHOULD BE OPEN NOW
003724  001425                         BEQ     9$              ;ERROR IF NOT
003726  032700  000000G                 BIT     #CS$ENT,R0      ;CHANNEL SHOULD HAVE BEEN OPENED VIA LOOKUP
003732  001022                         BNE     9$              ;ERROR IF NOT
003734  105063  000000G                 CLRB    C.NUMQ(R3)      ;THERE SHOULD BE NO ACTIVE I/O OPERATIONS
003740  126327  000000G 000007         CMPB    C.DEVQ(R3),#7   ;CHECK VALIDITY OF UNIT NUMBER
003746  101014                         BHI     9$              ;ERROR IF TOO BIG
003750  042700  000000C                 BIC     #^C<CS$NMX>,R0  ;GET DEVICE INDEX NUMBER
003754  020037  000000G                 CMP     R0,NUMDEV       ;MAKE SURE IT IS A VALID NUMBER
003760  101007                         BHI     9$              ;BR IF TOO BIG
                                ;
                                ;  Tell shared file system about the reopen.
                                ;
003762  005737  000000G                 TST     JCDB            ;Does job have any shared file channels?
003766  001402                         BEQ     4$              ;Br if not
003770  004777  000000G                 CALL    @SFRSST         ;WE MAY BE REOPENING A SHARED FILE
                                ;
                                ;  Finished
                                ;
003774  000137  000000G         4$:     JMP     EMTXIT
                                ;
                                ;  Error -- Savestatus info we restored was invalid.
                                ;           Return error code of 1.
                                ;
004000  005063  000000G         9$:     CLR     C.CSW(R3)       ;PURGE THE CHANNEL
004004  012700  000001                 MOV     #1,R0           ;RETURN ERROR CODE 1
004010  000137  000000G                 JMP     SETERR
```

```
                                        .SBTTL  .PURGE
                                ;-------------------------------------------------------------------------
                                ;  Purge a channel.
                                ;
004014  013703  000000G         PURGE:  MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
004020  032763  000000G 000000G        BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN NOW?
004026  001434                        BEQ     1$              ;BR IF NOT OPEN
004030  004737  000000G                CALL    IOWAIT          ;WAIT FOR I/O ON CHANNEL TO FINISH
004034  113701  000000G                MOVB    CHNNUM,R1       ;GET CHANNEL NUMBER
004040  105737  000000G                TSTB    NSPLDV          ;Are there any spooled devices?
004044  001403                        BEQ     2$              ;BR IF NOT
004046                                OCALL   SDCLOS          ;SEE IF THIS IS A SPOOLED DEVICE
004054  005737  000000G         2$:     TST     JCDB            ;Does job have any shared file channels?
004060  001402                        BEQ     4$              ;Br if not
004062  004777  000000G                CALL    @SFCLS          ;SEE IF IT IS A SHARED FILE
                                ;
                                ;  If we are purging a channel opened to a mag tape and I/O abort
                                ;  entry point code is turned off, then treat the purge like a close.
                                ;
004066  016300  000000G         4$:     MOV     C.CSW(R3),R0    ;GET CHANNEL STATUS WORD
004072  042700  000000C                BIC     #^C<CS$NMX>,R0  ;MASK OUT ALL BUT DEVICE INDEX NUMBER
004076  032760  000000G 000000G        BIT     #DS$NRD,DVSTAT(R0);IS DEVICE A MAGNETIC TAPE?
004104  001405                        BEQ     1$              ;BR IF NOT
004106  005737  000000G                TST     IOABFL          ;IS IO ABORT IN EFFECT?
004112  001002                        BNE     1$              ;BR IF YES
004114  000137  000000G                JMP     CLOSE           ;TREAT PURGE ON MAG TAPE LIKE CLOSE
004120  005063  000000G         1$:     CLR     C.CSW(R3)       ;MARK CHANNEL AS CLOSED
                                ;
                                ;  See if we need to report an error from .CLOSZ
                                ;
004124  113700  000000G                MOVB    CLZERR,R0       ;Do we need to report .CLOSZ error?
004130  001402                        BEQ     3$              ;Br if not
004132  000137  000000G                JMP     SETERR          ;And report to user
                                ;
                                ;  Succesful completion
                                ;
004136  000137  000000G         3$:     JMP     EMTXIT          ;If no errors, done
```

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

```
                                        .SBTTL  .WAIT
                                ;-------------------------------------------------------------------------------
                                ;  .WAIT -- Wait for all I/O to finish on a channel.
                                ;
004236  013703  000000G         DOWAIT: MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL
                                ;
                                ;  Make sure channel is open
                                ;
004242  032763  000000G 000000G         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
004250  001003                         BNE     1$              ;BR IF YES
004252  005000                         CLR     R0              ;RETURN ERROR CODE OF 0 IF NOT OPEN
004254  000137  000000G               JMP     SETERR
                                ;
                                ;  Wait for all I/O to finish
                                ;
004260  004737  000000G         1$:     CALL    IOWAIT          ;WAIT FOR I/O TO FINISH
                                ;
                                ;  Check for hardware errors
                                ;
004264  032763  000000G 000000G         BIT     #CS$ERR,C.CSW(R3);DID A HARD ERROR OCCUR?
004272  001407                         BEQ     9$              ;BR IF NOT
004274  042763  000000G 000000G         BIC     #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
004302  012700  000001                 MOV     #1,R0           ;Return error code 1
004306  000137  000000G               JMP     SETERR
                                ;
                                ;  Finished
                                ;
004312  000137  000000G         9$:     JMP     EMTXIT          ;RETURN
```

```
                                        .SBTTL  .GTIM
                                ;-------------------------------------------------------------------
                                ;  Get time of day.
                                ;
004316  013700  000002G        GETTIM: MOV     EMTBLK+2,R0     ;GET ADDR OF USER'S TIME BUFFER
004322  004737  000000G                CALL    VALADW          ;VALIDATE THE ADDRESS
004326  013746  000000G                MOV     SYTIMH,-(SP)    ;RETURN HIGH-ORDER TIME
004332  106620                        MTPD    (R0)+
004334  013746  000000G                MOV     SYTIML,-(SP)    ;RETURN LOW-ORDER TIME
004340  106610                        MTPD    (R0)
004342  000403                        BR      UPDATE          ;GO UPDATE DATE CELL

                                        .SBTTL  .DATE
                                ;-------------------------------------------------------------------
                                ;  Get current date (return to user in R0).
                                ;
004344  013737  000000G 000000G DATE:   MOV     SYSDAT,URO      ;RETURN DATE TO USER IN R0
                                ;  Move current date value to cell in user's simulated monitor vector table.
004352  013702  000000G        UPDATE: MOV     CXTRMN,R2       ;GET BASE ADDRESS OF JOB VECTOR
004356  013762  000000G 000000G        MOV     SYSDAT,R$DATE(R2);SET DATE VALUE
004364  000137  000000G                JMP     EMTXIT

                                        .SBTTL  .SDTTM
                                ;-------------------------------------------------------------------
                                ;  .SDTTM -- Set system date and time.
                                ;
004370  032737  000000G 000000G SDTTM:  BIT     #P0$OPR,PRIVCO  ;IS THIS A PRIVILEGED USER RUNNING?
004376  001003                        BNE     1$              ;BR IF YES
004400  005000                        CLR     R0              ;RETURN ERROR CODE 0
004402  000137  000000G                JMP     SETERR
                                ;
                                ;  Set date
                                ;
004406  013702  000002G        1$:     MOV     EMTBLK+2,R2     ;GET ADDRESS OF DATE/TIME VALUE BLOCK
004412  010200                        MOV     R2,R0           ;VALIDATE THE ADDRESS
004414  004737  000000G                CALL    VALADW
004420  106522                        MFPD    (R2)+           ;GET NEW DATE VALUE
004422  012600                        MOV     (SP)+,R0
004424  002402                        BLT     2$              ;IF NEGATIVE THEN DON'T CHANGE DATE
004426  010037  000000G                MOV     R0,SYSDAT       ;SET NEW SYSTEM DATE
                                ;
                                ;  Set time
                                ;
004432  106522                2$:     MFPD    (R2)+           ;GET HIGH-ORDER TIME VALUE
004434  012600                        MOV     (SP)+,R0
004436  002405                        BLT     3$              ;IF NEGATIVE, DON'T CHANGE TIME VALUE
004440  010037  000000G                MOV     R0,SYTIMH       ;SET NEW HIGH-ORDER TIME VALUE
004444  106522                        MFPD    (R2)+           ;GET NEW LOW-ORDER TIME VALUE
004446  012637  000000G                MOV     (SP)+,SYTIML    ;SET LOW-ORDER TIME VALUE
                                ;
                                ;  Finished
                                ;
004452  000137  000000G        3$:     JMP     EMTXIT
```

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

```
                        .SBTTL  .MRKT
                ;-----------------------------------------------------------------------
                ;  Start a mark-time request.
                ;
004542          MRKT:
                ;
                ;  Get an I/O queue entry.
                ;
004542  004737  000000G         CALL    GETQ            ;GET A FREE I/O QUEUE ELEMENT
                ;
                ;  Set up the queue element as a completion queue element.
                ;  (R1 = Address of queue element)
                ;
004546  113761  000000G 000000G        MOVB    CORUSR,CQ$JOB(R1);SET JOB # IN QUEUE ELEMENT
                ;  Set time value.
004554  013700  000002G         MOV     EMTBLK+2,R0     ;GET ADDRESS OF TIME VALUE ARG BLOCK
004560  004737  000000G         CALL    UACHKW          ;MAKE SURE ADDRESS IS LEGAL
004564  103440                  BCS     9$              ;BR IF INVALID ADDRESS
004566  106520                  MFPD    (R0)+           ;GET HIGH-ORDER TIME VALUE
004570  012661  000000G         MOV     (SP)+,CQ$HOT(R1);SET HIGH-ORDER TIME VALUE
004574  106510                  MFPD    (R0)            ;GET LOW-ORDER TIME VALUE
004576  012661  000000G         MOV     (SP)+,CQ$LOT(R1)
                ;  Set address of completion routine
004602  013700  000004G         MOV     EMTBLK+4,R0     ;GET ADDRESS OF COMPLETION ROUTINE
004606  004737  000000G         CALL    UACHKW          ;MAKE SURE ADDRESS IS LEGAL
004612  103425                  BCS     9$              ;BR IF NOT LEGAL
004614  010061  000000G         MOV     R0,CQ$RTN(R1)   ;SET ADDRESS OF COMPLETION ROUTINE
                ;  Set mapping for kernel PAR 5 that was in effect when EMT was executed.
004620  013761  000000G 000000G        MOV     EMTMAP,CQ$PA5(R1);SET EMT ENTRY MAPPING FOR PAR 5
                ;  Set ID number
004626  013761  000006G 000000G        MOV     EMTBLK+6,CQ$R0(R1);SET COMPLETION ID NUMBER
                ;
                ;  Add queue entry to mark-time list.
                ;
004634                          DISABL                  ;** DISABLE **
004642  013761  000000G 000000G        MOV     MRKTHD,CQ$LNK(R1);ADD TO FRONT OF LIST
004650  010137  000000G         MOV     R1,MRKTHD
004654                          ENABL                   ;** ENABLE **
                ;
                ;  Finished
                ;
004662  000137  000000G         JMP     EMTXIT
                ;
                ;  Error -- Invalid address in MRKT argument list.
                ;
004666  004737  000000G 9$:     CALL    QFREE           ;RELEASE THE QUEUE ELEMENT
004672  012700  177766          MOV     #-12,R0         ;RETURN ERROR CODE
004676  000137  000000G         JMP     SETERR          ;ABORT EMT
```

```
                                .SBTTL  .CMKT
                        ;-------------------------------------------------------------------------
                        ;  Cancel mark-time request.
                        ;
004702  010103          CMKT:   MOV     R1,R3           ;COPY JOB INDEX #
004704  013702  000002G         MOV     EMTBLK+2,R2     ;GET ID VALUE
004710  001460                  BEQ     1$              ;BR IF WE SHOULD CANCEL ALL REQUESTS FOR JOB
                        ;
                        ;  Cancel a specifiec mark-time request for this job.
                        ;
004712  012704  000000C         MOV     #MRKTHD-CQ$LNK,R4;FAKE UP POINTER TO LIST HEAD
004716                          DISABL                  ;** DISABLE **
004724  016401  000000G 4$:     MOV     CQ$LNK(R4),R1   ;GET ADDRESS OF NEXT MARK-TIME QUEUE ENTRY
004730  001442                  BEQ     5$              ;BR IF REACHED END OF LIST
004732  120361  000000G         CMPB    R3,CQ$JOB(R1)   ;DOES THIS ENTRY BELONG TO OUR JOB?
004736  001007                  BNE     2$              ;BR IF NOT
004740  132761  000000G 000000G BITB    #QF$IOT,CQ$FLG(R1);IS THIS A .TIMIO ENTRY?
004746  001003                  BNE     2$              ;BR IF YES
004750  020261  000000G         CMP     R2,CQ$RO(R1)    ;DO ID NUMBERS MATCH?
004754  001402                  BEQ     3$              ;BR IF YES -- FOUND ENTRY TO DELETE
004756  010104          2$:     MOV     R1,R4           ;CHAIN FORWARD TO NEXT ENTRY IN LIST
004760  000761                  BR      4$
                        ;
                        ;  Found specified queue entry -- delete from list.
                        ;
004762  016164  000000G 000000G 3$:     MOV     CQ$LNK(R1),CQ$LNK(R4);RELINK LIST AROUND US
004770                          ENABL                   ;** ENABLE **
                        ;
                        ;  See if we need to return remaining time.
                        ;
004776  013700  000004G         MOV     EMTBLK+4,R0     ;DOES USER WANT REMAINING TIME?
005002  001411                  BEQ     6$              ;BR IF NOT
005004  004737  000000G         CALL    UACHKW          ;MAKE SURE RECIEVING AREA IS LEGAL
005010  103406                  BCS     6$              ;BR IF NOT LEGAL
005012  016146  000000G         MOV     CQ$HOT(R1),-(SP);RETURN HIGH-ORDER TIME
005016  106620                  MTPD    (R0)+
005020  016146  000000G         MOV     CQ$LOT(R1),-(SP);RETURN LOW-ORDER TIME
005024  106610                  MTPD    (R0)
                        ;
                        ;  Return queue element to free list.
                        ;
005026  004737  000000G 6$:     CALL    QFREE           ;FREE THE Q ELEMENT
                        ;
                        ;  Finished
                        ;
005032  000137  000000G         JMP     EMTXIT
                        ;
                        ;  Could not find specified queue element -- return error code 0.
                        ;
005036                  5$:     ENABL                   ;** ENABLE **
005044  005000                  CLR     R0              ;RETURN ERROR CODE 0
005046  000137  000000G         JMP     SETERR
                        ;
                        ;  Cancel all mark-time requests for this job.
                        ;
005052  004737  005062' 1$:     CALL    CANMKT          ;CANCEL ALL MARK-TIME REQUESTS FOR JOB
005056  000137  000000G         JMP     EMTXIT          ;FINISHED
```

```
                                ;-------------------------------------------------------------------------------
                                ;  CANMKT is called to cancel all mark-time requests for the current job.
                                ;  It is called when a .CMKT EMT is done with an id of 0 or when a job
                                ;  exits.
                                ;
                                ;  Inputs:
                                ;   MRKTHD = Head of mark-time queue chain.
                                ;
005062  010146                  CANMKT: MOV     R1,-(SP)
005064  010446                          MOV     R4,-(SP)
005066  012704  000000C         1$:     MOV     #MRKTHD-CQ$LNK,R4;DUMMY POINTER TO LIST HEAD
005072                                  DISABL                  ;** DISABLE **
005100  016401  0000000         2$:     MOV     CQ$LNK(R4),R1   ;GET NEXT ENTRY IN LIST
005104  001423                          BEQ     4$              ;BR IF REACHED END OF LIST
005106  123761  0000000 0000000         CMPB    CORUSR,CQ$JOB(R1);IS THIS ENTRY FOR THIS JOB?
005114  001015                          BNE     3$              ;BR IF NOT
005116  132761  0000000 0000000         BITB    #QF$IOT,CQ$FLG(R1);IS THIS A HANDLER .TIMIO ENTRY?
005124  001011                          BNE     3$              ;BR IF YES
005126  016164  0000000 0000000         MOV     CQ$LNK(R1),CQ$LNK(R4);RELINK LIST AROUND US
005134                                  ENABL                   ;** ENABLE **
005142  004737  0000000                 CALL    QFREE           ;RETURN QUEUE ELEMENT TO FREE LIST
005146  000747                          BR      1$
005150  010104                  3$:     MOV     R1,R4           ;LINK TO NEXT ELEMENT IN LIST
005152  000752                          BR      2$
                                ;
                                ;  Finished
                                ;
005154                          4$:     ENABL                   ;** ENABLE **
005162  012604                          MOV     (SP)+,R4
005164  012601                          MOV     (SP)+,R1
005166  000207                          RETURN
```

```
                                        .SBTTL  .CSTAT
                                ;-------------------------------------------------------------------------
                                ;  Return channel status
                                ;
005170  013703  000000G         CHSTAT: MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
                                ;
                                ;  Make sure channel is open or at least has device index
                                ;
005174  032763  000000G 000000G        BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
005202  001004                        BNE     1$              ;BR IF YES
005204  032763  000000G 000000G        BIT     #CS$NMX,C.CSW(R3) ;IS THERE ANY DEVICE INFORMATION?
005212  001444                        BEQ     7$              ;IMMEDIATE ERROR EXIT IF NOT
                                ;
                                ;  Channel is open.  Return status.
                                ;  Also if there is device information available, but then return error.
                                ;
005214  013704  000002G         1$:     MOV     EMTBLK+2,R4     ;GET ADDRESS OF USER'S STATUS BLOCK
005220  010400                        MOV     R4,R0           ;VALIDATE ADDRESS
005222  004737  000000G               CALL    VALADW
                                ;  Point R0 to user's channel status block passed in EMT argument.
005226  010437  000000G               MOV     R4,UR0          ;Point R0 to channel status block
                                ;  Return CSW
005232  016346  000000G               MOV     C.CSW(R3),-(SP)
005236  106624                        MTPD    (R4)+
                                ;  Return starting block number of file
005240  016346  000000G               MOV     C.SBLK(R3),-(SP)
005244  106624                        MTPD    (R4)+
                                ;  Return allocated length of file.
005246  016346  000000G               MOV     C.LENG(R3),-(SP)
005252  106624                        MTPD    (R4)+
                                ;  Return highest block number written so far
005254  016346  000000G               MOV     C.USED(R3),-(SP)
005260  106624                        MTPD    (R4)+
                                ;  Return device unit number.
005262  005046                        CLR     -(SP)           ;ENSURE HIGH BYTE IS CLEAR
005264  116316  000000G               MOVB    C.DEVQ(R3),(SP) ;MOVE UNIT NUMBER INTO LOW BYTE
005270  106624                        MTPD    (R4)+
                                ;  Return Rad50 device name.
005272  016301  000000G               MOV     C.CSW(R3),R1    ;GET DEVICE TABLE #
005276  042701  177701                BIC     #^C76,R1        ;GET DEV # ONLY
005302  016146  000000G               MOV     PNAME(R1),-(SP) ;GET RAD50 DEVICE NAME
005306  106624                        MTPD    (R4)+
                                ;
                                ;  Normal or error return?
                                ;
005310  032763  000000G 000000G        BIT     #CS$OPN,C.CSW(R3) ;IS CHANNEL OPEN?
005316  001402                        BEQ     7$              ;ERROR RETURN IF NOT
                                ;
                                ;  Normal exit
                                ;
005320  000137  000000G               JMP     EMTXIT
                                ;
                                ;  Error return
                                ;
005324  005000                7$:     CLR     R0              ;RETURN ERROR CODE OF 0
005326  000137  000000G               JMP     SETERR
                                
```

```
                                        .SBTTL  .CDFN
                                ;-------------------------------------------------------------------------------
                                ;  Define channels.
                                ;
005332  013737  0000020 0000000 CDFN:   MOV     EMTBLK+2,UCHAN  ;SAVE ADDRESS OF USER CHANNEL SPACE
005340  000137  0000000                 JMP     EMTXIT
```

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

```
                                        .WORD   TGMPRI          ;-18. -- Maximum authorized priority
                                        .WORD   TGUCLB          ;-20. -- Number of data blocks/job in UCL file
                                        .WORD   TGPRIL          ;-22. -- Primary line #
                                        .WORD   TGSYNM          ;-24. -- Name of physical SY device
                                        .WORD   TGPRLO          ;-26. -- Get value of PRILOW
                                        .WORD   TGPRHI          ;-28. -- Get value of PRIHI
                                        .WORD   TGPRNT          ;-30. -- Get parent job number
                                        .WORD   TGVERS          ;-32. -- Get system version number
                                        .WORD   TGRSUB          ;-34. -- Get relative subprocess number
000042                          MXTSXV  =       <.-TSXGVC>-2    ;Maximum legal offset value
                                ;
                                ;  Get job number
                                ;
005574  010100                  TGJOBN: MOV     R1,R0           ;GET JOB INDEX NUMBER
005576  006200                          ASR     R0              ;CONVERT TO JOB NUMBER
005600  000207                          RETURN
                                ;
                                ;  Get Lead-in character
                                ;
005602  113700  000000G         TGLDIN: MOVB    VTSLCH,R0       ;GET LEAD-IN CHARACTER VALUE
005606  000207                          RETURN
                                ;
                                ;  Determine if this is a privileged job
                                ;
005610  032737  000000G 000000G TGPRIV: BIT     #PO$SYS,PRIVCO  ;ARE WE PRIVILEGED?
005616  001401                          BEQ     1$              ;BR IF NOT
005620  005200                          INC     R0              ;SET PRIVILEGED FLAG
005622  000207                  1$:     RETURN
                                ;
                                ;  Determine if I/O page mapping is in effect
                                ;
005624  032761  000000G 000000G TGIOMP: BIT     #$IOMAP,LSW6(R1);IS I/O PAGE MAPPING IN EFFECT?
005632  001401                          BEQ     1$              ;BR IF NOT
005634  005200                          INC     R0              ;SAY MAPPING IS IN EFFECT
005636  000207                  1$:     RETURN
                                ;
                                ;  Get Project number
                                ;
005640  016100  000000G         TGPROJ: MOV     LPROJ(R1),R0    ;GET PROJECT #
005644  000207                          RETURN
                                ;
                                ;  Get programmer number
                                ;
005646  016100  000000G         TGPROG: MOV     LPROG(R1),R0    ;GET PROGRAMMER #
005652  000207                          RETURN
                                ;
                                ;  Get TSX-Plus site license number
                                ;
005654  013700  000000G         TGLICN: MOV     TSXSIT,R0       ;GET LICENSE NUMBER
005660  000207                          RETURN
                                ;
                                ;  Get current job execution priority
                                ;
005662  116100  000000G         TGCPRI: MOVB    LPRI(R1),R0     ;Get current job priority
005666  000207                          RETURN
                                ;
                                ;  Get maximum authorized job priority
```

```
                                ;
005670  113700  000000G  TGMPRI: MOVB    MXJPRI,R0       ;Get maximum authorized execution priority
005674  000207                  RETURN
                                ;
                                ;  Get value of PRILOW
                                ;
005676  113700  000000G  TGPRLO: MOVB    VPRILO,R0
005702  000207                  RETURN
                                ;
                                ;  Get value of PRIHI
                                ;
005704  113700  000000G  TGPRHI: MOVB    VPRIHI,R0
005710  000207                  RETURN
                                ;
                                ;  Get number of blocks per job in TSXUCL data base file
                                ;
005712  013700  000000G  TGUCLB: MOV     UCLBLK,R0       ;Get number of blocks per job
005716  000207                  RETURN
                                ;
                                ;  Get number of controlling primary line (0 if we are a primary line)
                                ;
005720  016100  000000G  TGPRIL: MOV     LNPRIM(R1),R0   ;Get number of our primary line
005724  020001                  CMP     R0,R1           ;Are we the primary line?
005726  001001                  BNE     1$              ;Br if not
005730  005000                  CLR     R0              ;Return 0 if we are a primary line
005732  006200          1$:     ASR     R0              ;Convert from 2*job # to job #
005734  000207                  RETURN
                                ;
                                ;  Get number of parent job
                                ;
005736  016100  000000G  TGPRNT: MOV     LPARNT(R1),R0   ;Get parent job index number
005742  006200                  ASR     R0              ;Convert to job number
005744  000207                  RETURN
                                ;
                                ;  Get physical name of SY device
                                ;
005746  013700  000000G  TGSYNM: MOV     SYNAME,R0       ;Get SY physical device name
005752  000207                  RETURN
                                ;
                                ;  Get system version number
                                ;
005754  012700  000000G  TGVERS: MOV     #TSXVRS,R0      ;Get system version number
005760  000207                  RETURN
                                ;
                                ;  Get relative subprocess number
                                ;
005762  010546          TGRSUB: MOV     R5,-(SP)        ;Save R5
005764  005000                  CLR     R0              ;Primary and detached are relative 0
005766  020127  000000G          CMP     R1,#LSTDL       ;Is this a virtual line?
005772  003414                  BLE     8$              ;Branch if not
005774  016105  000000G          MOV     LNPRIM(R1),R5   ;Get owner line number
006000  016505  000000G          MOV     LSECPT(R5),R5   ;Get pointer to owner's subprocess list
006004  005200          1$:     INC     R0              ;Count subprocess number
006006  120125                  CMPB    R1,(R5)+        ;Is this our line?
006010  001405                  BEQ     8$              ;Done if line matches
006012  020027  000000G          CMP     R0,#MAXSEC      ;Any more possible?
006016  002772                  BLT     1$              ;Keep checking if so
```

```
   172 006020  012700  177777              MOV     #-1,R0          ;Whoa! This should never happen!
   173 006024  012605              8$:     MOV     (SP)+,R5        ;Restore R5
   174 006026  000207                      RETURN
```

```
                                        ;
                                        ;   .PVAL  --  Store value into monitor table
                                        ;
006030  032702  000001          PVAL:   BIT     #1,R2           ;IS ADDRESS ODD?
006034  001071                          BNE     GVBAD           ;BR IF YES -- ERROR
006036  020227  000000G                 CMP     R2,#MAXGVL      ;IS VALUE IN PROPER RANGE?
006042  101066                          BHI     GVBAD           ;BR IF NOT
006044  063702  000000G                 ADD     CXTRMN,R2       ;ADD BASE ADDRESS OF JOB'S RMON AREA
006050  016237  000002  000000G         MOV     2(R2),UR0       ;RETURN ORIGINAL VALUE IN R0
006056  010362  000002                  MOV     R3,2(R2)        ;STORE NEW VALUE
006062  000137  000000G                 JMP     EMTXIT          ;FINISHED
                                        ;
                                        ;   .PEEK -- Get value from specified memory location
                                        ;
006066  032702  000001          PEEK:   BIT     #1,R2           ;IS ADDRESS ODD?
006072  001052                          BNE     GVBAD           ;BR IF YES -- ERROR
006074  020227  000000G                 CMP     R2,#RMNBAS      ;IS ADDRESS WITHIN RMON TABLE?
006100  103413                          BLO     PKLOW           ;BR IF TOO LOW
006102  020227  000000C                 CMP     R2,#RMNBAS+MAXGVL ;IS IT TOO BIG?
006106  101010                          BHI     PKLOW           ;BR IF YES
006110  162702  177776G                 SUB     #<RMNBAS-2>,R2  ;MAKE ADDRESS INTO OFFSET
006114  063702  000000G                 ADD     CXTRMN,R2       ;ADD BASE ADDRESS OF JOB'S RMON AREA
006120  011237  000000G          PKGET:  MOV     (R2),UR0        ;RETURN VALUE
006124  000137  000000G                 JMP     EMTXIT
                                        ;  Peek into low memory area or I/O page
006130  032737  000000G 000000G PKLOW:  BIT     #PO$MEM,PRIVCO  ;Are we authorized to access phys memory?
006136  001430                          BEQ     GVBAD           ;BR IF NOT
006140  000767                          BR      PKGET           ;GO GET THE VALUE
                                        ;
                                        ;   .POKE -- Store value into specified memory location
                                        ;
006142  032702  000001          POKE:   BIT     #1,R2           ;IS ADDRESS ODD?
006146  001024                          BNE     GVBAD           ;ERROR IF YES
006150  020227  000000G                 CMP     R2,#RMNBAS      ;IS ADDRESS WITHIN RMON TABLE?
006154  103414                          BLO     POLOW           ;BR IF TOO LOW
006156  020227  000000C                 CMP     R2,#RMNBAS+MAXGVL ;IS IT TOO BIG?
006162  101011                          BHI     POLOW           ;BR IF YES
006164  162702  177776G                 SUB     #<RMNBAS-2>,R2  ;CONVERT TO OFFSET WITHIN TABLE
006170  063702  000000G                 ADD     CXTRMN,R2       ;ADD BASE OF JOB'S RMON AREA
006174  011237  000000G          POVAL:  MOV     (R2),UR0        ;GET ORIGINAL VALUE
006200  010312                          MOV     R3,(R2)         ;SET NEW VALUE
006202  000137  000000G                 JMP     EMTXIT          ;FINISHED
                                        ;  Poke into low memory or I/O page
006206  032737  000000G 000000G POLOW:  BIT     #PO$MEM,PRIVCO  ;Are we authorized to access phys memory?
006214  001401                          BEQ     GVBAD           ;BR IF NOT
006216  000766                          BR      POVAL           ;GO STORE VALUE
                                        ;
                                        ;   Invalid offset specified
                                        ;
006220  005000                  GVBAD:  CLR     R0              ;RETURN ERROR # 0
006222  000137  000000G                 JMP     SETERR
                                        ;
                                        ;   Sub-function jump vector
                                        ;
006226  005404'                 GVJMP:  .WORD   DOGVAL          ;0 - .GVAL
006230  006066'                         .WORD   PEEK            ;1 - .PEEK
006232  006030'                         .WORD   PVAL            ;2 - .PVAL
```

```
     58 006234  006142'                          .WORD   POKE            ;3 - .POKE
```

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

```
                                        .SBTTL  TTEMT  -- Terminal control EMT
                                ;-------------------------------------------------------------------
                                ;  The terminal control EMT is used to perform all of the terminal
                                ;  control functions that can also be performed by use of the
                                ;  lead-in type terminal functions.
                                ;  The form of the EMT argument block is:
                                ;
                                ;       .BYTE   0,152
                                ;       .WORD   function_value
                                ;       .WORD   control_value
                                ;
                                ;  Where "function_value" corresponds to the value of the ascii character
                                ;  that would be used with the lead-in sequence, and "control_value"
                                ;  corresponds to the letter that would be sent as an argument to some
                                ;  of the control functions.
                                ;
006322  013702  000002G       TTEMT:  MOV     EMTBLK+2,R2     ;Get function_value
006326  162702  000101                SUB     #'A,R2          ;Convert letter into index value
006332  100416                BMI     9$              ;Br if not legal value
006334  020227  000000G       CMP     R2,#MAXCC       ;Is the letter ok?
006340  103013                BHIS    9$              ;Br if too big
006342  006302                ASL     R2              ;Convert to word table index
006344  013700  000004G       MOV     EMTBLK+4,R0     ;Get control_value
006350  016237  006400' 006362'        MOV     TTFUN(R2),1$    ;Set address of processing routine
006356  004737  000000G       CALL    OVRHC           ;Call overlay handler
006362  000000                1$:     .WORD   0               ;Store address of routine to call here
006364  000137  000000G               JMP     EMTXIT          ;Finished
                                ;
                                ;  Error -- Invalid function value
                                ;
006370  012700  000001        9$:     MOV     #1,R0           ;Return error code 1
006374  000137  000000G               JMP     SETERR
```

```
                                ;-------------------------------------------------------------------
                                ;  Vector of addresses of processing routines
                                ;
006400  000000G  TTFUN:  .WORD   SETRBF          ;A
006402  000000G          .WORD   CMDB           ;B
006404  000000G          .WORD   CMDC           ;C
006406  000000G          .WORD   GTSPAC         ;D
006410  000000G          .WORD   CMDE           ;E
006412  000000G          .WORD   CMDF           ;F
006414  000000G          .WORD   CMDG           ;G
006416  000000G          .WORD   CMDH           ;H
006420  000000G          .WORD   CMDI           ;I
006422  000000G          .WORD   CMDJ           ;J
006424  000000G          .WORD   CMDK           ;K
006426  000000G          .WORD   CMDL           ;L
006430  000000G          .WORD   CMDM           ;M
006432  000000G          .WORD   CMDN           ;N
006434  000000G          .WORD   CMDO           ;O
006436  000000G          .WORD   RSSPAC         ;P
006440  000000G          .WORD   SFWAC          ;Q
006442  000000G          .WORD   CMDR           ;R
006444  000000G          .WORD   CMDS           ;S
006446  000000G          .WORD   CMDT           ;T
006450  000000G          .WORD   CMDU           ;U
006452  000000G          .WORD   SFWL           ;V
006454  000000G          .WORD   CMDW           ;W
006456  000000G          .WORD   CMDX           ;X
006460  000000G          .WORD   CMDY           ;Y
006462  000000G          .WORD   CMDZ           ;Z
```

```
                                        .SBTTL  SNDMSG -- Send a message to a line
                                ;-------------------------------------------------------------------------
                                ;  Send an asciz message to a line.
                                ;
                                ;  EMT argument block format:
                                ;
                                ;       .BYTE   Sub-function,127
                                ;       .WORD   Line-number
                                ;       .WORD   Buffer-address
                                ;
                                ;  Where Sub-function is:
                                ;            No-gag-override    Gag-override
                                ;  Hang             0                  1
                                ;  Error            2                  3
                                ;
006464  032737  000000G 000000G SNDMSG: BIT     #PO$SND,PRIVCO  ;Are we authorized to send messages?
006472  001003                          BNE     4$              ;Br if yes
006474  005000                          CLR     R0              ;Return error code 0
006476  000137  000000G                 JMP     SETERR
006502  032737  000000G 000000G 4$:     BIT     #PO$OPR,PRIVCO  ;Does user have OPER privilege?
006510  001003                          BNE     5$              ;Br if yes
006512  042737  000001  000000G         BIC     #1,EMTBLK       ;Cannot override gag unless oper privilege
                                ;
                                ;  See if a valid job number was specified.
                                ;
006520  013703  000002G         5$:     MOV     EMTBLK+2,R3     ;GET NUMBER OF LINE TO RECEIVE MESSAGE
006524  006303                          ASL     R3              ;CONVERT TO JOB INDEX NUMBER
006526  020327  000002                  CMP     R3,#2           ;CAN'T BE LESS THAN 2
006532  103473                          BLO     9$              ;BR IF TOO SMALL
006534  020327  000000G                 CMP     R3,#LSTSL       ;MAKE SURE NOT TOO LARGE
006540  101070                          BHI     9$              ;BR IF TOO LARGE
006542  032763  000000G 000000G         BIT     #$DILUP,LSW(R3) ;IS JOB LOGGED ON?
006550  001464                          BEQ     9$              ;BR IF NOT
006552  032763  000000G 000000G         BIT     #$DETCH,LSW(R3) ;IS JOB A DETACH LINE?
006560  001060                          BNE     9$              ;BR - DETACH LINES CAN'T RECEIVE
006562  032763  000000G 000000G         BIT     #$TTGAG,LSW7(R3);IS LINE GAGGED?
006570  001414                          BEQ     3$              ;BR IF NOT
006572  032763  000000G 000000G         BIT     #$INKMN,LSW4(R3);IS KMON RUNNING NOW?
006600  001010                          BNE     3$              ;BR IF YES
006602  032737  000001  000000G         BIT     #1,EMTBLK       ;DOES HE WANT TO OVERRIDE GAG?
006610  001004                          BNE     3$              ;BR IF YES
006612  012700  000001                  MOV     #1,R0           ;RETURN ERROR CODE 1 IF LINE IS GAGGED
006616  000137  000000G                 JMP     SETERR
                                ;
                                ;  Get a free message buffer.
                                ;
006622  004737  006730'         3$:     CALL    GTMSBF          ;GET A FREE MESSAGE BUFFER
006626  103004                          BCC     6$              ;CONTINUE IF NO ERROR
006630  012700  000002                  MOV     #2,R0           ;RETURN ERROR CODE 2 IF NO MESSAGE BUFFERS
006634  000137  000000G                 JMP     SETERR
                                ;
                                ;  Move message from user's buffer to system message buffer.
                                ;
006640  013703  000004G         6$:     MOV     EMTBLK+4,R3     ;GET ADDRESS OF USER'S MESSAGE BUFFER
006644  010300                          MOV     R3,R0           ;VALIDATE ADDRESS
006646  004737  000000G                 CALL    VALADB
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$:   CALL    GETUCH          ;GET CHAR FROM USER'S BUFFER
     62 006672  110022                         MOVB    R0,(R2)+        ;MOVE TO SYSTEM MESSAGE BUFFER
     63 006674  001403                         BEQ     2$              ;BR IF REACHED END
     64 006676  020205                         CMP     R2,R5           ;REACHED END OF BUFFER SPACE?
     65 006700  103772                         BLO     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     R0              ;RETURN ERROR CODE 0
     81 006724  000137  000000G                JMP     SETERR
```

```
                                        .SBTTL  GTMSBF -- Get free system message buffer
                                ;-------------------------------------------------------------------------
                                ;  GTMSBF is called to get a free system message buffer block.
                                ;  If none are available, the job is suspended until one becomes free.
                                ;
                                ;  Outputs:
                                ;   R4 = Address of message block acquired.
                                ;
006730                          GTMSBF: DISABL                  ;** DISABLE **
006736  005737  000000G                TST     NMUMB           ;ARE THERE ANY AVAILABLE MESSAGE BLOCKS?
006742  003004                        BGT     3$              ;BR IF SO
006744  032737  000002  000000G       BIT     #2,EMTBLK       ;DOES USER WANT TO WAIT?
006752  001035                        BNE     7$              ;BR IF NOT
006754  013704  000000G        3$:     MOV     SNMSHD,R4       ;GET ADDRESS OF 1ST FREE BLOCK
006760  001013                        BNE     1$              ;BR IF OK
006762  032737  000002  000000G       BIT     #2,EMTBLK       ;DOES USER WANT TO WAIT?
006770  001026                        BNE     7$              ;BR IF NOT
                                ;
                                ;  No message buffers available.
                                ;  Suspend job until one becomes available.
                                ;
006772  012700  000000G        2$:     MOV     #S$WSMB,R0      ;WAIT FOR MESSAGE BUFFER STATE.
006776  004737  000000G                CALL    QNSPND          ;WAIT FOR MESSAGE BUFFER ** ENABLE **
007002  004737  000000G                CALL    CHKABT          ;WERE WE ABORTED WHILE ASLEEP?
007006  000750                        BR      GTMSBF          ;TRY AGAIN
                                ;
                                ;  Got a buffer.  Unlink from free chain.
                                ;
007010  005337  000000G        1$:     DEC     NMUMB           ;ONE LESS FREE BUFFER
007014  016437  000000G 000000G        MOV     SB$LNK(R4),SNMSHD;REMOVE FROM FREE LIST
007022                                  ENABL                   ;** ENABLE **
007030  010400                        MOV     R4,R0           ;INITIALIZE SB$PNT
007032  062700  000000G                ADD     #SB$TXT,R0
007036  010064  000000G                MOV     R0,SB$PNT(R4)
007042  000241                        CLC                     ;SAY NO ERRORS
007044  000401                        BR      9$
                                ;
                                ;  No free message buffers, flag error return
                                ;
007046  000261                 7$:     SEC
                                ;
                                ;  Finished
                                ;
007050  000207                 9$:     RETURN
```

```
                                        .SBTTL  QMSG   -- Queue a message for a job
                                ;-------------------------------------------------------------------------
                                ;  QMSG is called to queue a system message for a job.
                                ;
                                ;  Inputs:
                                ;   R1 = Job number that is to receive message.
                                ;   R4 = Address of system message buffer.
                                ;
007052  010146                  QMSG:   MOV     R1,-(SP)
                                ;
                                ;  Add message buffer to queue for line
                                ;
007054                                  DISABL                  ;** DISABLE **
007062  016101  0000000G                MOV     LNPRIM(R1),R1   ;OBTAIN THE CURRENT PRIMARY LINE NUMBER
007066  016100  0000000G                MOV     LMSGBF(R1),R0   ;GET ADDRESS OF HEAD OF MESSAGE LIST FOR JOB
007072  001411                          BEQ     3$              ;BR IF NO MESSAGES QUEUED FOR JOB
007074  005760  0000000G          1$:   TST     SB$LNK(R0)      ;IS THIS LAST ENTRY IN LIST?
007100  001403                          BEQ     2$              ;BR IF YES
007102  016000  0000000G                MOV     SB$LNK(R0),R0   ;CHAIN TO NEXT QUEUED MESSAGE BLOCK
007106  000772                          BR      1$
007110  010460  0000000G          2$:   MOV     R4,SB$LNK(R0)   ;ADD OUR MESSAGE TO TAIL OF LIST
007114  000402                          BR      4$
007116  010461  0000000G          3$:   MOV     R4,LMSGBF(R1)   ;ADD OUR MESSAGE BLOCK TO LIST FOR JOB
007122  005064  0000000G          4$:   CLR     SB$LNK(R4)      ;SAY WE ARE AT END OF LIST
007126                                  ENABL                   ;** ENABLE **
                                ;
                                ;  Initiate transmission to the line.
                                ;
007134  116101  0000000G                MOVB    LNMAP(R1),R1    ;GET VIRTUAL LINE #
007140  004777  0000000G                CALL    @TRNSTR         ;INITIATE TRANSMISSION
                                ;
                                ;  Finished
                                ;
007144  012601                          MOV     (SP)+,R1
007146  000207                          RETURN
```

```
      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
```

```
                                        .SBTTL  .SPCPS -- Alter exit address from a completion routine
                                ;-----------------------------------------------------------------------
                                ;  The .SPCPS EMT is used to set a address that is used to control the
                                ;  mainline code execution on exit from a completion routine.
                                ;  The real work to accomplish this is done in SYSXIT.
                                ;
007202  105737  000000G       ESPCPS: TSTB    CURCP           ;ARE WE IN A COMPLETION ROUTINE NOW?
007206  001003                        BNE     1$              ;BR IF YES
007210  005000                        CLR     R0              ;RETURN ERROR CODE 0 IF NOT
007212  000137  000000G               JMP     SETERR
007216  005737  000000G       1$:     TST     SPCPS           ;IS ANOTHER .SPCPS PENDING NOW?
007222  001404                        BEQ     2$              ;BR IF NOT
007224  012700  000001                MOV     #1,R0           ;RETURN ERROR CODE 1 IF YES
007230  000137  000000G               JMP     SETERR
007234  013700  000002G       2$:     MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S INFORMATION BLOCK
007240  004737  000000G               CALL    VALADW          ;VALIDATE THE ADDRESS
007244  010037  000000G               MOV     R0,SPCPS        ;SAVE ADDRESS OF INFO BLOCK
007250  000137  000000G               JMP     EMTXIT          ;EXIT -- SYSXIT WILL DO REAL PROCESSING
```

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

```
                                        .SBTTL  JBINFO -- Get information about a specific job
                                ;-------------------------------------------------------------------------
                                ;  The JBINFO EMT is used to obtain information about a specific job.
                                ;  The form of the EMT argument block is:
                                ;  +--------+--------+
                                ;  |   144  |    0   |
                                ;  +--------+--------+
                                ;  |Sub Fun.| Job #  |
                                ;  +--------+--------+
                                ;  | Result buf addr |
                                ;  +-----------------+
                                ;
                                ;  Get Subfunction number and check its range
                                ;
007326  113705  000003G         JBINFO: MOVB    EMTBLK+3,R5     ;GET SUB-FUNCTION NUMBER
007332  020527  000012                 CMP     R5,#MXJIFN      ;IS IT TOO BIG?
007336  101404                         BLOS    1$              ;BR IF OK
007340  012700  000001                 MOV     #1,R0           ;ERROR CODE 1 IF TOO BIG
007344  000137  0000000G               JMP     SETERR
                                ;
                                ;  Subfunction number is valid.
                                ;  Check job number.
                                ;
007350  113701  000002G         1$:     MOVB    EMTBLK+2,R1     ;GET SPECIFIED JOB NUMBER
007354  001404                         BEQ     2$              ;ZERO IS INVALID
007356  006301                         ASL     R1              ;CONVERT TO LINE INDEX NUMBER
007360  020127  0000000G               CMP     R1,#LSTSL       ;IS IT A VALID NUMBER?
007364  101404                         BLOS    3$              ;BR IF YES
007366  012700  000002          2$:     MOV     #2,R0           ;ERROR 2 IF INVALID LINE NUMBER
007372  000137  0000000G               JMP     SETERR
                                ;
                                ;  See if line is currently logged on
                                ;
007376  032761  0000000G 0000000G 3$:   BIT     #$KINIT,LSW(R1) ;IS THE LINE LOGGED ON NOW?
007404  001003                         BNE     4$              ;BR IF YES
007406  005000                         CLR     R0              ;ERROR # 0 IF NOT
007410  000137  0000000G               JMP     SETERR
                                ;
                                ;  Check the address specified for the user's result buffer
                                ;
007414  013704  000004G         4$:     MOV     EMTBLK+4,R4     ;GET USER'S VIRTUAL ADDRESS
007420  010400                         MOV     R4,R0
007422  004737  0000000G               CALL    VALADW          ;MAKE SURE ADDRESS IS REASONABLE
                                ;
                                ;  Everything looks good.
                                ;  Branch off to processing routine based on sub-function number.
                                ;  At this point the following registers are set up:
                                ;   R1 = Line index number for job about which information is desired.
                                ;   R4 = Address of user's buffer where result is to be placed.
                                ;   R5 = Sub function index number.
                                ;
007426  006305                         ASL     R5              ;CONVERT SUB-FUNCTION # TO WORD INDEX
007430  000175  010014'               JMP     @JIJMPX(R5)     ;ENTER PROCESSING ROUTINE
                                ;
                                ;  #0 -- Get job status flags
                                ;
007434  005000                 JISTAT: CLR     R0              ;DEVELOP FLAGS IN R0
```

```
                                        ;  See if job is detached or virtual
007436  020127  000000G                       CMP     R1,#LSTPL       ;PRIMARY LINE?
007442  101410                               BLOS    1$              ;BR IF YES
007444  020127  000000G                       CMP     R1,#LSTDL       ;DETACHED LINE?
007450  101403                               BLOS    2$              ;BR IF YES
007452  052700  000000G                       BIS     #JIVLN,R0       ;SET VIRTUAL-LINE FLAG
007456  000402                               BR      1$
007460  052700  000000G               2$:     BIS     #JIDLN,R0       ;SET DETACHED-LINE FLAG
                                        ;  See if job is locked in memory
007464  032761  000000G 000000G       1$:     BIT     #$MLOCK,LSW6(R1);IS JOB LOCKED IN MEMORY?
007472  001402                               BEQ     4$              ;BR IF NOT
007474  052700  000000G                       BIS     #JIMLOK,R0      ;SET MEMORY-LOCKED FLAG
                                        ;  See if job has operator privilege
007500  010046                       4$:     MOV     R0,-(SP)        ;SAVE OUR STATUS CODES
007502                                       OCALL   GETCXT          ;GET EXCLUSIVE ACCESS TO CONTEXT BUFFER
007510  103002                               BCC     6$              ;BR IF GOT IT
007512  004737  000000G                       CALL    STOP            ;JOB WAS ABORTED WHILE WAITING
007516  010102                       6$:     MOV     R1,R2           ;GET INDEX # OF JOB WE ARE COPYING FROM
007520  012703  000000G                       MOV     #PRIVCO,R3      ;ADDR OF INFO TO GET
007524  012700  000002                       MOV     #2,R0           ;GET # BYTES TO COPY
007530                                       OCALL   REDCXT          ;GET INFO FROM CONTEXT BLOCK OF OTHER JOB
007536  032777  000000G 000000G              BIT     #P0$SYS,@CXTBUF ;IS THIS A PRIVILEGED JOB?
007544  001402                               BEQ     5$              ;BR IF NOT
007546  052716  000000G                       BIS     #JIPRIV,(SP)    ;SET PRIVILEGE FLAG
007552                                       OCALL   FRECXT          ;FREE THE CONTEXT BUFFER
007560  012600                               MOV     (SP)+,R0        ;GET STATUS FLAGS
007562  000422                               BR      JIRET1          ;GO RETURN VALUE
                                        ;
                                        ;  #1 -- Get job's run state
                                        ;
007564  012700  010042'          JIRUN:  MOV     #JIRNST,R0      ;Point to state table
007570  116102  000000G                  MOVB    LSTATE(R1),R2   ;Get job's current execution state
007574  120220                   1$:     CMPB    R2,(R0)+        ;Search for state in table
007576  001406                           BEQ     2$              ;Br if found it
007600  005200                           INC     R0              ;Skip over value
007602  020027  010126'                  CMP     R0,#JIRNND      ;Checked all entries?
007606  103772                           BLO     1$              ;Br if yes
007610  005000                           CLR     R0              ;Don't recognize state, return 0
007612  000406                           BR      JIRET1          ;Return the value
007614  111000                   2$:     MOVB    (R0),R0         ;Get value to return
007616  000404                           BR      JIRET1          ;Return the value
                                        ;
                                        ;  #2 -- Get amount of memory in use by job
                                        ;
007620  016100  000000G          JIMEM:  MOV     LNBLKS(R1),R0   ;GET # 256-WORD BLOCKS USED BY JOB
007624  066100  000000G                  ADD     LNSBLK(R1),R0   ;ADD # BLOCKS USED BY PLAS REGIONS
007630  010046                   JIRET1: MOV     R0,-(SP)        ;STORE VALUE INTO USER'S BUFFER
007632  106614                           MTPD    (R4)
007634  000137  000000G                  JMP     EMTXIT          ;FINISHED WITH EMT
                                        ;
                                        ;  #3 -- Get job connect time (minutes)
                                        ;
007640  013700  000000G          JICONT: MOV     MINTIM,R0       ;GET CURRENT SYSTEM UP-TIME
007644  166100  000000G                  SUB     LCONTM(R1),R0   ;GET # MINUTES JOB HAS BEEN CONNECTED
007650  000767                           BR      JIRET1
                                        ;
                                        ;  #4 -- Get base 256-word block # assigned to job area
```

```
                                        ;
007652  016100  000000G         JIBASE: MOV     LBASE(R1),R0    ;GET BASE BLOCK #
007656  000764                         BR      JIRET1
                                        ;
                                        ;  #5 -- Get name of program being run by job
                                        ;
007660  016102  000000G         JIPROG: MOV     LPRG1(R1),R2    ;GET FIRST 3 CHARS OF NAME
007664  016103  000000G                 MOV     LPRG2(R1),R3    ;GET SECOND 3 CHARS OF NAME
007670  000411                         BR      JIRET2
                                        ;
                                        ;  #6 -- Get project/programmer number
                                        ;
007672  016102  000000G         JIPPN:  MOV     LPROJ(R1),R2    ;GET PROJECT NUMBER
007676  016103  000000G                 MOV     LPROG(R1),R3    ;GET PROGRAMMER NUMBER
007702  000404                         BR      JIRET2
                                        ;
                                        ;  #7 -- Get CPU time used by job
                                        ;
007704  016102  000000G         JICPU:  MOV     LCPUHI(R1),R2   ;GET HIGH-ORDER CPU TIME
007710  016103  000000G                 MOV     LCPULO(R1),R3   ;GET LOW-ORDER CPU TIME
007714  010246                 JIRET2: MOV     R2,-(SP)        ;RETURN FIRST WORD OF RESULT
007716  106624                         MTPD    (R4)+
007720  010346                         MOV     R3,-(SP)        ;RETURN SECOND WORD OF RESULT
007722  106624                         MTPD    (R4)+
007724  000137  000000G               JMP     EMTXIT          ;FINISHED WITH EMT
                                        ;
                                        ;  #8 -- Job execution priority
                                        ;
007730  116100  000000G         JIPRIO: MOVB    LPRI(R1),R0     ;Get current execution priority for job
007734  000735                         BR      JIRET1          ;Return
                                        ;
                                        ;  #9 -- Job name
                                        ;
007736  010105                 JINAME: MOV     R1,R5           ;Copy job index number
007740  012702  000006                 MOV     #6,R2           ;6 words per name
007744  070502                         MUL     R2,R5           ;Name should be 12 bytes long
007746  062705  000000G                 ADD     #LUNAME,R5      ;Convert to name table index
007752  012546                 1$:     MOV     (R5)+,-(SP)     ;Fetch next two bytes of name
007754  106624                         MTPD    (R4)+           ;Move to user's buffer
007756  077203                         SOB     R2,1$           ;Loop through whole name
007760  000137  000000G               JMP     EMTXIT          ;Finished with EMT
                                        ;
                                        ;  #10 -- Parent and primary job numbers
                                        ;
007764  016102  000000G         JIPAPR: MOV     LNPRIM(R1),R2   ;Get index number of primary line
007770  006202                         ASR     R2              ;Convert to job number
007772  016103  000000G                 MOV     LPARNT(R1),R3   ;Get parent job index number
007776  006203                         ASR     R3              ;Convert to job number
010000  010246                         MOV     R2,-(SP)        ;Return primary job number
010002  106624                         MTPD    (R4)+
010004  010346                         MOV     R3,-(SP)        ;Return parent job number
010006  106624                         MTPD    (R4)+
010010  000137  000000G               JMP     EMTXIT          ;FINISHED WITH EMT
                                        ;
                                        ;  Branch table used to enter job information routines
                                        ;
010014  007434'                 JIJMPX: .WORD   JISTAT          ; 0  -- GET JOB STATE FLAGS
```

```
    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
```

```
 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
```

```
                                        .SBTTL  Get terminal type code
                                ;-------------------------------------------------------------------------
                                ;  This EMT returns in R0 a numeric value that indicates the type of
                                ;  terminal from which the job is being run.
                                ;  The terminal type is declared in the line definition in TSGEN or
                                ;  by the SET TERMINAL keyboard command.
                                ;
010126  012702  010212'         TRMEMT: MOV     #LOTRM,R2       ;POINT TO START OF TERMINAL TYPE TABLE
010132  016104  000000G                 MOV     LTRMTP(R1),R4   ;GET TERMINAL TYPE FLAGS FOR OUR JOB
010136  116100  000000G                 MOVB    LNPRIM(R1),R0   ;Get out primary line number
010142  032760  000000G 000000G         BIT     #$V52EM,LSW11(R0);Are we emulating a VT52?
010150  001402                         BEQ     1$              ;Br if not
010152  012704  000000G                 MOV     #VT52,R4        ;Say terminal is a VT52
010156  020422                 1$:     CMP     R4,(R2)+        ;SEARCH FOR MATCHING PATTERN IN TABLE
010160  001405                         BEQ     2$              ;BR IF FOUND
010162  020227  010236'                 CMP     R2,#HITRM       ;HAVE WE SEARCHED ALL OF TABLE?
010166  101773                         BLOS    1$              ;BR IF NOT
010170  005002                         CLR     R2              ;SAY UNKNOWN TERMINAL TYPE
010172  000403                         BR      3$
010174  162702  010212'         2$:     SUB     #LOTRM,R2       ;CONVERT TABLE OFFSET TO TERMINAL CODE NUMBER
010200  006202                         ASR     R2
010202  010237  000000G         3$:     MOV     R2,@R0          ;RETURN TERMINAL TYPE CODE # IN USER'S R0
010206  000137  000000G                 JMP     EMTXIT          ;FINISHED
                                ;
                                ;  Table to translate terminal type flags into code number.
                                ;
010212  000000G                 LOTRM:  .WORD   VT52            ;  1 - VT52
010214  000000G                         .WORD   VT100           ;  2 - VT100
010216  000000G                         .WORD   HAZEL           ;  3 - Hazeltine
010220  000000G                         .WORD   ADM3A           ;  4 - ADM3A
010222  000000G                         .WORD   LA36            ;  5 - LA36
010224  000000G                         .WORD   LA120           ;  6 - LA120
010226  000000G                         .WORD   DIABLO          ;  7 - Diablo
010230  000000G                         .WORD   QUME            ;  8 - Qume
010232  000000G                         .WORD   VT2007          ;  9 - VT200 with 7 bit control codes
010234  000000G                         .WORD   VT2008          ; 10 - VT200 with 8 bit control codes
010236                          HITRM:
                                
                                        .SBTTL  Get Project-Programmer number
                                ;-------------------------------------------------------------------------
                                ;  This emt is used to obtain the current Project-Programmer number
                                ;  that a job is running under.
                                ;
010236  013700  000002G         GETPPN: MOV     EMTBLK+2,R0     ;GET ADDRESS OF CELL TO RECEIVE PPN
010242  004737  000000G                 CALL    VALADW          ;VALIDATE THE ADDRESS
010246  016146  000000G                 MOV     LPROJ(R1),-(SP) ;MOVE PROJECT # TO USER'S BUFFER
010252  106620                         MTPD    (R0)+
010254  016146  000000G                 MOV     LPROG(R1),-(SP)
010260  106620                         MTPD    (R0)+           ;MOVE PROGRAMMER # TO USER'S BUFFER
010262  000137  000000G                 JMP     EMTXIT          ;FINISHED
```

```
                                        .SBTTL  MISC. TSX EMT'S
                                ;
                                ;-------------------------------------------------------------
                                ;  THE .LNUM EMT RETURNS IN R0 THE NUMBER OF THE TSX
                                ;  LINE WHICH THE USER IS CONNECTED TO, OR THE LINE NUMBER
                                ;  OF THE SPECIFIED SUBPROCESS
                                ;
010266  105737  0000006        XGTLN:  TSTB    EMTBLK          ;WANT CURRENT LINE OR RELATIVE
010272  001440                          BEQ     1$              ;BR IF CURRENT
                                ;
                                ;  THIS IS A REQUEST TO RETURN THE REAL LINE NUMBER OF A SUBPROCESS
                                ;
010274  020127  0000006                 CMP     R1,#LSTPL       ;ARE WE A PRIMARY LINE?
010300  101406                          BLOS    4$              ;BR IF OK
010302  020127  0000006                 CMP     R1,#LSTDL       ;ARE WE A VIRTUAL LINE?
010306  101003                          BHI     4$              ;BR IF OK
010310  012700  000002                  MOV     #2,R0           ;DETACHED JOB'S DON'T HAVE SUBPROCESSES
010314  000425                          BR      6$              ;GO COMPLAIN ABOUT IT
010316  016101  0000006        4$:     MOV     LNPRIM(R1),R1   ;GET OUR PARENT INDEX
010322  013705  0000026                 MOV     EMTBLK+2,R5     ;WANTS LINE NUMBER FOR THIS SUBPROCESS
010326  001002                          BNE     2$              ;BR IF NOT ASKING FOR PARENT
010330  010105                          MOV     R1,R5           ;COPY PARENT'S PROCESS INDEX
010332  000411                          BR      3$              ;DON'T NEED SUBPROCESS MAPPING
010334  020527  0000006        2$:     CMP     R5,#MAXSEC      ;VALID SUBPROCESS NUMBER?
010340  101403                          BLOS    5$              ;BR IF OK
010342  012700  000001                  MOV     #1,R0           ;SIGNAL INVALID SUBPROCESS #
010346  000410                          BR      6$              ;AND GO REPORT ERROR
010350  066105  0000006        5$:     ADD     LSECPT(R1),R5   ;POINT TO PRIMARY LINE'S SUBPROCESS TABLE
010354  114505                          MOVB    -(R5),R5        ;GET SPECIFIED LINE'S INDEX (^1 IS OFFSET 0)
010356  006205                3$:     ASR     R5              ;CONVERT INDEX TO LINE #
010360  010537  0000006                 MOV     R5,URO          ;PASS TO USER
010364  001034                          BNE     XJNOP           ;IF IN USE, RETURN LINE # TO USER
010366  005000                          CLR     R0              ;ELSE REPORT LINE NOT ACTIVE
010370  000137  0000006        6$:     JMP     SETERR          ;RETURN WITH ERROR
                                ;
                                ;  THIS IS A REQUEST TO GET OUR CURRENT LINE NUMBER
                                ;
010374  006201                1$:     ASR     R1              ;CONVERT TO LINE #
010376  010137  0000006                 MOV     R1,URO          ;RETURN TO USER IN R0
010402  000425                          BR      XJNOP
                                ;
                                ;-------------------------------------------------------------
                                ;  THE .SPLSP EMT RETURNS IN R0 THE NUMBER OF FREE BLOCKS OR
                                ;  NUMBER OF BLOCKS IN USE IN THE SPOOL FILE.
                                ;
010404  105737  0000006        XSPLSP: TSTB    EMTBLK          ;NUMBER OF BLOCKS FREE OF IN USE?
010410  001417                          BEQ     1$              ;BR IF NUMBER OF FREE BLOCKS
010412  113701  0000006                 MOVB    NSPLDV,R1       ;GET # OF SPOOLED DEVICES
010416  001410                          BEQ     2$              ;BR IF NONE
010420  013700  0000006                 MOV     NSPLBL,R0       ;GET TOTAL BLOCKS IN SPOOL FILE
010424  163700  0000006                 SUB     NFRESB,R0       ;SUBTRACT THE NUMBER OF FREE BLOCKS
010430  070127  0000006                 MUL     #PVSPBL,R1      ;MULTIPLY BY # OF BLOCKS FOR EACH SPOOLED DEV
010434  160100                          SUB     R1,R0           ;AND SUBTRACT FROM BLOCKS IN USE
010436  000401                          BR      3$
010440  005000                2$:     CLR     R0              ;RETURN ZERO IF NO SPOOLER
010442  010037  0000006        3$:     MOV     R0,URO          ;RETURN # OF BLOCKS IN USE TO USER IN R0
010446  000403                          BR      XJNOP           ;RETURN
```

```
     58 010450  013737  000000G 000000G 1$:     MOV     NFRESB,URO      ;RETURN # OF FREE BLOCKS TO USER IN R0
     59 010456  000137  000000G         XJNOP:  JMP     EMTXIT
     60                                 ;
     61                                 ;-------------------------------------------------------
     62                                 ;  SET/RESET ODT CHARACTER ACTIVATION MODE
     63                                 ;
     64 010462  105737  000000G         XODTMD: TSTB    EMTBLK          ;SET OR RESET MODE?
     65 010466  001407                          BEQ     1$              ;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
```

```
                                ;-------------------------------------------------------------------------
                                ;  Set interactive or non-interactive mode for current program.
                                ;
010560  005737  000002G        INTEMT: TST     EMTBLK+2        ;Set or reset interactive mode?
010564  001404                        BEQ     1$              ;Reset interactive move
                                ;
                                ;  Set interactive mode
                                ;
010566  042761  000000G 000000G       BIC     #$NOINT,LSW7(R1);Clear non-interactive mode flag
010574  000405                        BR      9$
                                ;
                                ;  Reset interactive mode
                                ;
010576  052761  000000G 000000G 1$:    BIS     #$NOINT,LSW7(R1);Run job non-interactively
010604  005061  000000G               CLR     LITIME(R1)      ;Clear interactive timer
                                ;
                                ;  Finished
                                ;
010610  000137  000000G          9$:    JMP     EMTXIT          ;Finished
```

```
                                        .SBTTL  Return system (swap) file specification
                                ;-------------------------------------------------------------------
                                ;  Return dev:filspc.ext for system swap files
                                ;
                                ;  The form of the EMT argument block is:
                                ;       .BYTE   file_index,164
                                ;       .WORD   buffer_address
                                ;
 010614  113705  000000G       SYFEMT: MOVB    EMTBLK,R5       ;GET SYSTEM FILE INDEX
 010620  020527  000004                CMP     R5,#MXSYFN      ;VALID FILE INDEX NUMBER?
 010624  101403                        BLOS    1$              ;BRANCH IF OK
 010626  005000                        CLR     R0              ;ERROR CODE 0 IF BAD INDEX
 010630  000137  000000G               JMP     SETERR
                                ;
                                ;  Convert system file index number to table offset
                                ;
 010634  006305                 1$:     ASL     R5              ;CONVERT INDEX TO TABLE OFFSET
 010636  016504  010672'                MOV     SYFTBL(R5),R4   ;GET POINTER TO FILE NAME
                                ;
                                ;  Get and validate user's return buffer
                                ;
 010642  013705  000002G               MOV     EMTBLK+2,R5     ;GET USER'S VIRTUAL ADDRESS
 010646  010500                        MOV     R5,R0
 010650  004737  000000G               CALL    VALADW          ;MAKE SURE ADDRESS IS OK
                                ;
                                ;  Now return 4 RAD50 words of dev:filspc.ext to user buffer
                                ;
 010654  012700  000004                MOV     #4,R0           ;MOVE 4 WORDS
 010660  012446                 2$:     MOV     (R4)+,-(SP)     ;GET NEXT RAD50 WORD
 010662  106625                        MTPD    (R5)+           ;MOVE TO USER BUFFER
 010664  077003                        SOB     R0,2$
 010666  000137  000000G               JMP     EMTXIT          ;FINISHED
                                ;
                                ;  System file specification pointers
                                ;
 010672  000000G                SYFTBL: .WORD   SWDBLK           ; 0 SWAP FILE
 010674  000000G                        .WORD   SPLBLK          ; 1 SPOOL FILE
 010676  000000G                        .WORD   RSFBLK          ; 2 PLAS REGION SWAP FILE
 010700  000000G                        .WORD   UCLDAT          ; 3 USER DEFINED COMMAND (UCL) FILE
 010702  000000G                        .WORD   INDFIL          ; 4 IND TEMP FILE
         000004                 MXSYFN  = <<.-SYFTBL>/2>-1
```

```
                                        .SBTTL  Set transmit/receive speed for a line
                                ;-------------------------------------------------------------------------
                                ;  Set transmit/receive speed for a line.
                                ;
                                ;  The form of the EMT argument block is:
                                ;       .BYTE   n,154
                                ;       .WORD   line_number
                                ;       .WORD   speed_code
                                ;  where n=0 is set speed, n=1 is reset xoff, n=2 is set dtr, n=3 is clear dtr
                                ;
010704                          EMTSPD:
                                ;
                                ;  Get line number and see if it is valid
                                ;
010704  013702  000002G                 MOV     EMTBLK+2,R2     ;Get line number
010710  006302                          ASL     R2              ;Convert to line index number
010712  001001                          BNE     1$              ;Br if line number specified
010714  010102                          MOV     R1,R2           ;Use current line
010716  020227  000000G         1$:     CMP     R2,#LSTSL       ;Primary or virtual line?
010722  101002                          BHI     10$             ;Br if not
010724  016202  000000G                 MOV     LNPRIM(R2),R2   ;Map virtual line # to real line #
010730  020227  000000G         10$:    CMP     R2,#LSTHL       ;Is this a valid line number?
010734  101010                          BHI     5$              ;Br if not
010736  032762  000000G 000000G         BIT     #$HARD,LSW3(R2) ;Line connected to hardware?
010744  001404                          BEQ     5$              ;Br if not
010746  032762  000000G 000000G         BIT     #$DEAD,LSW3(R2) ;Is line installed?
010754  001404                          BEQ     4$              ;Br if yes
010756  012700  000002          5$:     MOV     #2,R0           ;Return error code 2 for invalid line #
010762  000137  000000G                 JMP     SETERR
                                ;
                                ;  See if we are privileged to perform this operation
                                ;
010766  032737  000000G 000000G 4$:     BIT     #P2$TRM,PRIVC2  ;Do we have privilege to change speed?
010774  001011                          BNE     11$             ;Br if yes
010776  105737  000000G                 TSTB    EMTBLK          ;EMT to reset XOFF status or line speed EMT?
011002  001402                          BEQ     12$             ;Br if changing line speed
011004  020201                          CMP     R2,R1           ;Resetting XOFF for our own line?
011006  001404                          BEQ     11$             ;Br if yes
011010  012700  000001          12$:    MOV     #1,R0           ;Return error code 1
011014  000137  000000G                 JMP     SETERR
                                ;
                                ;  See if this is actually the EMT to reset terminal XOFF status
                                ;
011020  010201                  11$:    MOV     R2,R1           ;Get line index number to R1
011022  123727  000000G 000001         CMPB    EMTBLK,#1       ;Set line speed, reset XOFF or DTR?
011030  002416                          BLT     2$              ;Br if want to set speed
011032  001422                          BEQ     20$             ;Br if want to reset XOFF
                                ;
                                ;  Want to raise or lower DTR, get current data set status
                                ;
011034  004737  000000G                 CALL    GETDSS          ;Get data set status into R0
011040  042700  000000G                 BIC     #MS$DTR,R0      ;Assume we want to lower DTR
011044  123727  000000G 000002         CMPB    EMTBLK,#2       ;Should we raise DTR?
011052  001002                          BNE     13$             ;Br if not
011054  052700  000000G                 BIS     #MS$DTR,R0      ;Set DTR up bit
011060  004737  000000G         13$:    CALL    SETDSS          ;Reset data set status
011064  000413                          BR      30$             ;And exit
```

Set transmit/receive speed for a line

```
                                        ;
                                        ;  Call hardware-dependent routine to set the line speed
                                        ;
011066  013700  000004G         2$:     MOV     EMTBLK+4,R0     ;Get speed code
011072  004737  000000G                 CALL    SETSPD          ;Set the line speed
011076  000406                          BR      30$             ;And exit
                                        ;
                                        ;  Reset XOFF status and start transmitter
                                        ;
011100  042761  000000G 000000G 20$:    BIC     #$CTRLS,LSW3(R1);Reset XOFF flag for line
011106  004777  000000G                 CALL    @TRNSTR         ;Try to start output to the line
011112  000400                          BR      30$             ;Finished
                                        ;
                                        ;  Finished
                                        ;
011114  000137  000000G         30$:    JMP     EMTXIT          ;Finished
```

Set transmit/receive speed for a line

```
                                ;-------------------------------------------------------------------------
                                ;  The .GTTAB emt is an obsolete TSX emt that is supposed to return in R0
                                ;  the address of a vector of pointers to tables within TSX.
                                ;  This emt is simulated here by moving some info into the low memory
                                ;  region of the job (372+) and returning that as the base of the
                                ;  pointer vector.
                                ;
011120  012702  000400          XGTTAB: MOV     #400,R2         ;STORE VALUES HERE
011124  012703  000372                  MOV     #372,R3         ;STORE VECTOR HERE
                                ;  TSLICH -- TSX command lead-in character.
011130  113746  0000006                 MOVB    VTSLCH,-(SP)    ;GET LEAD-IN CHARACTER
011134  106623                          MTPD    (R3)+           ;MOVE TO JOB SPACE
                                ;  Name of running program.
011136  010246                          MOV     R2,-(SP)        ;SET POINTER TO PROGRAM NAME
011140  106623                          MTPD    (R3)+           ;IN VECTOR
011142  012700  000004                  MOV     #4,R0           ;GET # WORDS TO MOVE
011146  012704  0000006                 MOV     #RUNDEV,R4      ;POINT TO NAME OF RUNNING PROGRAM
011152  012446                  1$:     MOV     (R4)+,-(SP)     ;MOVE NAME TO JOB SPACE
011154  106622                          MTPD    (R2)+
011156  077003                          SOB     R0,1$
                                ;
                                ;  Tell user that vector is located at 350.
                                ;
011160  012737  000350  0000006         MOV     #350,@R0        ;RETURN VECTOR ADDRESS IN USER'S R0
011166  000137  0000006                 JMP     EMTXIT          ;FINISHED
```

```
                                        .SBTTL  Access TSX system tables
                                ;-------------------------------------------------------------------------
                                ;  EMT's to get a table value, store a table value, set bits and clear bits.
                                ;  Operator privilege is required to modify a table.
                                ;
                                ;  EMT channel byte (byte 0 in arg block) indicates function desired...
                                ;
                                ;  0 ==> Get value from table.
                                ;  1 ==> Store value into table.
                                ;  2 ==> Set bits in table.
                                ;  3 ==> Clear bits in table.
                                ;  4 ==> Special table operation.
                                ;
011172  113705  000000G       TBLEMT: MOVB    EMTBLK,R5       ;GET SUB-FUNCTION CODE
011176  001412                        BEQ     1$              ;BR IF IT IS GET-VALUE
                                ;
                                ;  Operator privilege is required to modify a table.
                                ;
011200  032737  000000G 000000G       BIT     #PO$SYS,PRIVCO  ;IS THIS USER PRIVILEGED?
011206  001004                        BNE     2$              ;BR IF YES
011210  012700  000001                MOV     #1,R0           ;RETURN ERROR CODE OF 1 IF NOT
011214  000137  000000G               JMP     SETERR
011220  013703  000004G       2$:     MOV     EMTBLK+4,R3     ;GET VALUE FOR STORE
011224  013702  000002G       1$:     MOV     EMTBLK+2,R2     ;GET TABLE INDEX
                                ;
                                ;  If subfunction code = 4 then branch off to special processing routine
                                ;
011230  020527  000004                CMP     R5,#4           ;IS THIS A SPECIAL TABLE OPERATION?
011234  001006                        BNE     11$             ;BR IF NOT
011236  006302                        ASL     R2              ;GET WORD TABLE INDEX FOR OPERATION
011240  020227  000000                CMP     R2,#MXSPTF      ;MAKE SURE RANGE IS OK
011244  101043                        BHI     9$              ;BR IF INVALID
011246  000172  011444'               JMP     @SPLTF(R2)      ;ENTER PROCESSING ROUTINE
                                ;
                                ;  This is not a special table function
                                ;
011252  006302                11$:     ASL     R2              ;CONVERT TO WORD TABLE VALUE
011254  002007                        BGE     4$              ;BR IF POS TABLE
                                ;
                                ;  Negative table index
                                ;
011256  005402                        NEG     R2              ;MAKE TABLE INDEX POSITIVE
011260  020227  000006                CMP     R2,#MXNGTX      ;IS VALUE TOO BIG?
011264  103033                        BHIS    9$              ;BR IF TOO BIG
011266  016204  011472'               MOV     NEGTAB(R2),R4   ;GET TABLE ADDRESS
011272  000406                        BR      3$
                                ;
                                ;  Positive table index
                                ;
011274  020227  000024        4$:     CMP     R2,#MXPSTX      ;IS TABLE INDEX TOO BIG?
011300  103025                        BHIS    9$              ;BR IF TOO BIG
011302  016204  011446'               MOV     POSTAB(R2),R4   ;GET TABLE ADDRESS
011306  060104                        ADD     R1,R4           ;INDEX BY LINE NUMBER
                                ;
                                ;  Perform requested operation on table
                                ;
011310  120527  000001        3$:     CMPB    R5,#1           ;CHECK SUB-FUNCTION CODE
```

```
                                        BGT     5$              ;BR IF BIT SET OR RESET
                                        BEQ     6$              ;BR IF STORE VALUE
                                ;  Fetch table value
                                        MOV     @R4,UR0         ;RETURN TABLE VALUE IN USER'S R0
                                        BR      10$
                                ;  Store value into table.
                                6$:     MOV     R3,@R4          ;STORE INTO TABLE
                                        BR      10$
                                5$:     CMP     R5,#2           ;BIT SET OR RESET?
                                        BNE     7$              ;BR IF BIT CLEAR
                                ;  Set bits in table
                                        BIS     R3,@R4          ;SET BITS
                                        BR      10$
                                ;  Clear bits in table
                                7$:     BIC     R3,@R4          ;CLEAR BITS IN TABLE
                                        BR      10$
                                ;
                                ;  Finished
                                ;
                                10$:    JMP     EMTXIT
                                ;
                                ;  Invalid table index
                                ;
                                9$:     CLR     R0              ;RETURN ERROR CODE 0
                                        JMP     SETERR
                                ;
                                ;  Special table function routines
                                ;
                                ;  1 -- Set maximum execution priority for job
                                ;
                                SETMXP: TST     R3              ;PRIORITY MUST BE > 0
                                        BLE     1$              ;BR IF INVALID
                                        CMP     R3,#MAXPRI      ;MAY NOT EXCEED THIS
                                        BLOS    2$              ;BR IF OK
                                1$:     MOVB    VPRIDF,R3       ;IF BAD, SET TO NORMAL PRIORITY
                                2$:     MOVB    R3,MXJPRI       ;SET MAX PRIORITY FOR JOB
                                        CMPB    R3,LBSPRI(R1)   ;COMPARE WITH CURRENT JOB PRIORITY
                                        BHIS    3$              ;BR IF CURRENT PRIORITY IS OK
                                        MOVB    R3,LBSPRI(R1)   ;SET BASE PRIORITY FOR JOB
                                        MOVB    R3,LPRI(R1)     ;SET CURRENT PRIORITY FOR JOB
                                        BIT     #$VNOTT,LSW(R1) ;Is this a subprocess disconnected from term?
                                        BEQ     3$              ;Br if not
                                        OCALL   SPPRED          ;Reduce prio of disconnected subprocess
                                3$:     JMP     EMTXIT          ;FINISHED
                                ;
                                ;  Jump table for special table functions (subfunction code = 4)
                                ;
                                SPLTF:  .WORD   SETMXP          ; 0 -- Set maximum job priority
                                MXSPTF  =       <.-SPLTF>-2
                                ;
                                ;  Table of table addresses for positive offsets.
                                ;
                                POSTAB: .WORD   LCONTM          ;+00 -- CONNECT TIME
                                        .WORD   LSW             ;+01 -- LSW
                                        .WORD   LSW2            ;+02 -- LSW2
                                        .WORD   LSW3            ;+03 -- LSW3
                                        .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
```

```
                                        .SBTTL  Get or set user name
                                ;-------------------------------------------------------------------
                                ;  Get or set user name for this job, or get or set the progam name.
                                ;  Byte 0 indicates whether we are getting the user name (0), setting the
                                ;  user name (1), getting the program name (2), setting the program
                                ;  name (3).
                                ;
011500  010105                  EMUNAM: MOV     R1,R5           ;GET JOB INDEX NUMBER
011502  013700  000002G                MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S BUFFER
011506  004737  000000G                CALL    VALADW          ;VALIDATE THE ADDRESS
011512  123727  000000G 000002        CMPB    EMTBLK,#2       ;GET OR SET PROGRAM NAME?
011520  002030                        BGE     3$              ;BR IF GET/SET PROGRAM NAME
011522  012702  000006                MOV     #6,R2           ;6 WORDS PER USER NAME
011526  070502                        MUL     R2,R5           ;* 12 BYTES PER USER NAME
011530  062705  000000G               ADD     #LUNAME,R5      ;POINT TO USER NAME TABLE ENTRY
011534  105737  000000G               TSTB    EMTBLK          ;IS THIS A REQUEST TO GET OR SET USER NAME
011540  001414                        BEQ     1$              ;BR IF GETTING THE NAME
                                ;
                                ;  This is a request to set the user name.
                                ;  SETNAME privilege is required to do this.
                                ;
011542  032737  000000G 000000G       BIT     #PO$NAM,PRIVCO  ;IS THIS JOB PRIVILEGED?
011550  001004                        BNE     2$              ;BR IF YES
011552  012700  000001                MOV     #1,R0           ;RETURN ERROR IF NOT
011556  000137  000000G               JMP     SETERR
011562  106520                  2$:     MFPD    (R0)+           ;GET NEXT WORD OF USER NAME
011564  012625                        MOV     (SP)+,(R5)+     ;MOVE TO SYSTEM TABLE
011566  077203                        SOB     R2,2$           ;LOOP TO MOVE ALL OF NAME
011570  000432                        BR      9$
                                ;
                                ;  This is a request to get the user name.
                                ;
011572  012546                  1$:     MOV     (R5)+,-(SP)     ;STACK A WORD OF THE USER NAME
011574  106620                        MTPD    (R0)+           ;MOVE TO USER'S BUFFER
011576  077203                        SOB     R2,1$
011600  000426                        BR      9$
                                ;
                                ;  This is a request to get or set the program name
                                ;
011602  003007                  3$:     BGT     4$              ;BR IF SETTING
011604  016146  000000G               MOV     LPRG1(R1),-(SP) ;GET FIRST 3 CHARS OF PROG NAME
011610  106620                        MTPD    (R0)+           ;MOVE TO USER'S BUFFER
011612  016146  000000G               MOV     LPRG2(R1),-(SP) ;GET SECOND 3 CHARS OF PROG NAME
011616  106620                        MTPD    (R0)+           ;MOVE TO USER'S BUFFER
011620  000416                        BR      9$
                                ;
                                ;    This is a request to set the program name
                                ;
011622  106520                  4$:     MFPD    (R0)+           ;GET FIRST 3 CHARS
011624  012605                        MOV     (SP)+,R5        ;OF NEW PGM NAME
011626  020527  175000                CMP     R5,#175000      ;IS THIS A VALID RAD50 VALUE
011632  103013                        BHIS    5$              ;BR IF NOT
011634  010561  000000G               MOV     R5,LPRG1(R1)    ;SET FIRST 3 CHARS OF PROG NAME
011640  106520                        MFPD    (R0)+           ;GET SECOND 3 CHARS
011642  012605                        MOV     (SP)+,R5        ;OF NEW PGM NAME
011644  020527  175000                CMP     R5,#175000      ;IS THIS A VALID RAD50 VALUE
011650  103004                        BHIS    5$              ;BR IF YES
```

Get or set user name

```
     58 011652  010561  000000G                 MOV     R5,LPRG2(R1)    ;SET SECOND 3 CHARS OF PROG NAME
     59 011656  000137  000000G         9$:     JMP     EMTXIT          ;FINISHED
     60
     61 011662  012700  000002         5$:     MOV     #2,R0           ;SIGNAL ERROR CODE
     62 011666  000137  000000G                 JMP     SETERR
```

```
                                        .SBTTL  Request operator privilege
                                ;-------------------------------------------------------------------
                                ;  Request operator privilege.
                                ;  This is only legal if user has SYSPRV privilege.
                                ;
011672  123727  000000G 000001  REQPRV: CMPB    EMTBLK,#1       ;Request to release locked program?
011700  001417                         BEQ     2$              ;Br if yes
011702  032737  000000G 000000G        BIT     #PO$SPV,PRIVCO  ;Does job have SETPRV privilege?
011710  001410                         BEQ     1$              ;Br if not
011712  052737  000000C 000000G        BIS     #<PO$SPV!PO$SYS!PO$NAM!PO$BYP>,PRIVCO ;ENABLE PRIVILEGES
011720  052761  000000G 000000G        BIS     #$PRGLK,LSW5(R1);Lock program to line
011726  000137  000000G                JMP     EMTXIT
011732  005000                 1$:     CLR     R0              ;RETURN ERROR CODE OF 0
011734  000137  000000G                JMP     SETERR
                                ;
                                ;  Release program lock
                                ;
011740  042761  000000G 000000G 2$:     BIC     #$PRGLK,LSW5(R1);Program no longer locked to line
011746  000137  000000G                JMP     EMTXIT
```

```
                                        .SBTTL  Establish Break key sentinal control
                                ;-------------------------------------------------------------------
                                ;  Set up control so that pressing the Break key will cause an
                                ;  asynchronous completion routine to be entered.
                                ;
011752                          GETBRK:
                                ;
                                ;  See if we are connecting a completion routine to a break character
                                ;  or to any input activation condition.
                                ;
011752  105737  000000G               TSTB    EMTBLK          ;Connecting to break character?
011756  001402                        BEQ     3$              ;Br if yes
011760  000137  000000G               JMP     STTCPL          ;Go connect compl to activation condition
                                ;
                                ;  Set completion routine for break character.
                                ;  Cancel any existing break control.
                                ;  Return address of existing breakpoint routine to user in R0.
                                ;
011764  005061  000000G         3$:   CLR     LBRKCH(R1)      ;NO BREAK CHARACTER
011770  005037  000000G               CLR     UR0             ;ASSUME NO BREAK ROUTINE NOW
011774  016100  000000G               MOV     LBRKCQ(R1),R0   ;GET ADDRESS OF BREAK COMPLETION QUEUE ENTRY
012000  001410                        BEQ     1$              ;BR IF NONE
012002  016037  000000G 000000G       MOV     CQ$RTN(R0),UR0  ;RETURN ADDRESS OF CURRENT BREAKPOINT ROUTINE
012010  005061  000000G               CLR     LBRKCQ(R1)      ;SAY NO COMPLETION ENTRY
012014  010001                        MOV     R0,R1           ;GET ADDRESS OF QUEUE ELEMENT TO R1 FOR QFREE
012016  004737  000000G               CALL    QFREE           ;RELEASE THE QUEUE ELEMENT
                                ;
                                ;  See if we need to establish a completion routine.
                                ;
012022  005737  000004G         1$:   TST     EMTBLK+4        ;DID HE SPECIFY THE ADDRESS OF A COMPL RTN?
012026  001437                        BEQ     2$              ;BR IF NOT
                                ;
                                ;  Get a queue element and set up as a completion queue element.
                                ;
012030  004737  000000G               CALL    GETQ            ;GET A QUEUE ELEMENT
012034  113702  000000G               MOVB    CORUSR,R2       ;GET USER INDEX NUMBER
012040  110261  000000G               MOVB    R2,CQ$JOB(R1)   ;SET JOB NUMBER IN QUEUE ELEMENT
012044  112761  000000G 000000G       MOVB    #S$TWFN,CQ$RNS(R1);SET EXECUTION STATE
012052  116261  000000G 000000G       MOVB    LPRI(R2),CQ$PRI(R1);SET EXECUTION PRIORITY
012060  112761  000000G 000000G       MOVB    #CP$STD,CQ$CP(R1);Set compl routine class priority
012066  013700  000004G               MOV     EMTBLK+4,R0     ;GET ADDRESS OF COMPL ROUTINE
012072  004737  000000G               CALL    UACHKW          ;MAKE SURE ADDRESS IS VALID
012076  103415                        BCS     9$              ;BR IF INVALID
012100  010061  000000G               MOV     R0,CQ$RTN(R1)   ;SET COMPL ROUTINE ADDRESS IN QUEUE ELEMENT
012104  013761  000000G 000000G       MOV     EMTMAP,CQ$PA5(R1);SET EMT ENTRY MAPPING FOR PAR 5
012112  010162  000000G               MOV     R1,LBRKCQ(R2)   ;SET ADDRESS OF BREAK QUEUE ELEMENT
                                ;
                                ;  Remember special user-defined break character.
                                ;
012116  113700  000002G               MOVB    EMTBLK+2,R0     ;GET USER-DEFINED BREAK CHARACTER
012122  010062  000000G               MOV     R0,LBRKCH(R2)   ;SAVE FOR TT INTERRUPT PROCESSING
                                ;
                                ;  Finished
                                ;
012126  000137  000000G         2$:   JMP     EMTXIT
                                ;
                                ;  Error -- Invalid argument address.
```

Establish Break key sentinal control

```
     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                                          .SBTTL  CHKTT -- CHECK I/O TO TT DEVICE
      2                                ;-----------------------------------------------------------------------
      3                                ;  CHKTT is called to see if the current I/O operation is being directed
      4                                ;  to device TT or some other device.
      5                                ;
      6                                ;  Inputs:
      7                                ;   CHNADR = Address of current channel block
      8                                ;
      9                                ;  Outputs:
     10                                ;   C-flag cleared if I/O is directed to TT
     11                                ;   C-flag set if I/O not directed to TT
     12                                ;
     13 012146  017700  0000000       CHKTT:  MOV     @CHNADR,R0      ;GET DEV TABLE INDEX
     14 012152  002005                        BGE     1$              ;BRANCH IF NOT OPEN
     15 012154  032700  000076                BIT     #76,R0          ;IS THIS TT DEVICE (INDEX # = 0)?
     16 012160  001002                        BNE     1$              ;BR IF NOT TT
     17 012162  000241                        CLC                     ;SAY THIS IS TT
     18 012164  000207                        RETURN
     19 012166  000261                1$:     SEC
     20 012170  000207                        RETURN
```

```
                                        .SBTTL  CKIOST -- See if scheduler wants to stop I/O
                                ;-------------------------------------------------------------------------
                                ;  If the job scheduler wants to stop all I/O for a job so that
                                ;  it can have a chance to swap the job, it sets the LIOHLD flag.
                                ;  The CKIOST routine checks to see if this flag is set
                                ;  and if so, suspends the job until all I/O is finished.
                                ;
012172  010146                  CKIOST: MOV     R1,-(SP)
012174  032737  000000G 000000G         BIT     #UMODE,EMTPS    ;Was this EMT executed in kernel mode?
012202  001434                          BEQ     2$              ;Br if yes -- Don't stop system EMT's
012204  105737  000000G                 TSTB    CURCP           ;Was EMT executed by a completion routine?
012210  001031                          BNE     2$              ;Br if yes -- I/O hold flag stops cpl rtns
012212  113701  000000G                 MOVB    CORUSR,R1       ;Get current job index #
012216  005761  000000G         1$:     TST     LIOHLD(R1)      ;Is I/O hold flag set for job?
012222  001424                          BEQ     2$              ;Br if not
012224                                  DISABL                  ;;;** Disable interrupts **
012232  105761  000000G                 TSTB    LIOCNT(R1)      ;;;Is any I/O in progress now?
012236  001407                          BEQ     3$              ;;;Br if not
012240  012700  000000G                 MOV     #S$IOWT,R0      ;;;Get I/O wait state code
012244  004737  000000G                 CALL    QNSPND          ;Enqueue in I/O wait state (enable ints)
012250  004737  000000G                 CALL    CHKABT          ;See if we were aborted while asleep
012254  000760                          BR      1$              ;See if we should continue waiting
012256  005061  000000G         3$:     CLR     LIOHLD(R1)      ;;;Reset I/O hold flag
012262                                  ENABL                   ;** Enable interrupts **
012270  004737  000000G                 CALL    SCHED           ;Call scheduler to probably swap job
                                ;
                                ;  Finished
                                ;
012274  012601                  2$:     MOV     (SP)+,R1
012276  000207                          RETURN
```

```
                                        .SBTTL  SETQ   -- Do setup of I/O queue element
                                ;-------------------------------------------------------------------------
                                ;  SETQ is called to set up parameters in an I/O queue element.
                                ;
                                ;  Inputs:
                                ;  R1 = Address of queue element
                                ;  EMTBLK = EMT arguments
                                ;  CHNNUM = User's channel number
                                ;  CHNADR = Address of CSW for channel
                                ;
                                ;  Outputs:
                                ;  R0 = Highest block number that will be accessed by I/O
                                ;  C-flag set if a fatal error is detected (Error code is returned in INTERR).
                                ;  UR0 = Number of words that will be transfered.
                                ;
012300  010246                  SETQ:   MOV     R2,-(SP)
012302  010346                          MOV     R3,-(SP)
012304  010446                          MOV     R4,-(SP)
                                ;
                                ;  Set up misc. parameters
                                ;
012306  013761  000000G 000000G         MOV     CHNNUM,Q.CHAN(R1);SET USER'S CHANNEL #
012314  013703  000000G                 MOV     CHNADR,R3       ;GET ADDRESS OF CSW FOR CHANNEL
012320  010361  000000G                 MOV     R3,Q.UCSW(R1)   ;SET POINTER TO CSW
012324  013761  000000G 000000G         MOV     @#KPAR6,Q.PA6(R1);Save context blk mapping to access channel
012332  016302  000000G                 MOV     C.CSW(R3),R2    ;GET CHANNEL STATUS WORD
012336  042702  177701                  BIC     #^C76,R2        ;EXTRACT DEVICE INDEX NUMBER
012342  110261  000000G                 MOVB    R2,Q.DEVX(R1)   ;SAVE IN QUEUE ELEMENT CELL
012346  116361  000000G 000000G         MOVB    C.DEVQ(R3),Q.UNIT(R1);SET UNIT NUMBER
012354  113702  000000G                 MOVB    CORUSR,R2       ;GET USER INDEX #
012360  006202                          ASR     R2              ;Convert to job number
012362  110261  000000G                 MOVB    R2,Q.JOB(R1)    ;Store job # into queue element
012366  020227  000037                  CMP     R2,#31.         ;Will number fit in old Q.JNUM field?
012372  101402                          BLOS    12$             ;Br if yes
012374  012702  000037                  MOV     #31.,R2         ;Signal that Q.JNUM field not valid
012400  072227  000003          12$:    ASH     #3,R2           ;Position for Q.JNUM field
012404  150261  000000G                 BISB    R2,Q.JNUM(R1)   ;STORE JOB #
                                ;
                                ;  Copy original channel block into internal channel block that is
                                ;  part of I/O queue element.
                                ;
012410  010100                          MOV     R1,R0           ;Get address of I/O queue block
012412  062700  000000G                 ADD     #Q.ICSW,R0      ;Get address of CSW within queue block
012416  010061  000000G                 MOV     R0,Q.CSW(R1)    ;Set pointer to internal CSW
012422  010302                          MOV     R3,R2           ;Get pointer to original channel block
012424  012220                          MOV     (R2)+,(R0)+     ;Copy original channel block to internal one
012426  012220                          MOV     (R2)+,(R0)+
012430  012220                          MOV     (R2)+,(R0)+
012432  012220                          MOV     (R2)+,(R0)+
012434  011210                          MOV     (R2),(R0)
                                ;
                                ;  Set up buffer address
                                ;
012436  013702  000006G                 MOV     EMTBLK+6,R2     ;GET # WORDS TO BE TRANSFERED
012442  001004                          BNE     5$              ;BR IF WORD COUNT SPECIFIED
012444  123727  000001G 000032          CMPB    EMTBLK+1,#32    ;IS THIS A .SPFUN REQUEST?
012452  001104                          BNE     1$              ;BR IF NOT -- THIS MUST BE A SEEK REQUEST
```

```
     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$:     CALL    CVTUAD          ;CONVERT USER ADDRESS TO PHYSICAL ADDRESS
     72 012532  010061  000000G                 MOV     R0,Q.BUFF(R1)   ;BUFFER ADDRESS RELATIVE TO Q.PAR
     73 012536  010261  000000G                 MOV     R2,Q.PAR(R1)    ;RELOCATION BASE ADDRESS
     74                                     ;
     75                                     ;  Set up word count and block number for transfer.
     76                                     ;
     77 012542  005037  000000G                 CLR     UR0             ;SAY # WORDS TRANSFERED = 0 (IN CASE OF ERROR)
     78 012546  013704  000006G                 MOV     EMTBLK+6,R4     ;GET WORD COUNT
     79 012552  062704  000377                  ADD     #377,R4         ;CONVERT TO BLOCK COUNT
     80 012556  105004                          CLRB    R4
     81 012560  000304                          SWAB    R4
     82 012562  013700  000002G                 MOV     EMTBLK+2,R0     ;GET BLOCK # FOR REQUEST
     83 012566  011302                          MOV     (R3),R2         ;GET CSW
     84 012570  042702  177701                  BIC     #^C76,R2        ;GET DEVICE INDEX
     85 012574  032762  000000G 000000G          BIT     #DS$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                          BHIS    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,UR0          ;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?
```

```
                                        BNE     11$             ;BR IF NOT
                                        MOV     R0,-(SP)        ;SAVE HIGH BLOCK NUMBER
                                        OCALL   LDIO            ;ALTER QUEUE ELEMENT FOR LOGICAL DISK I/O
                                        MOV     (SP)+,R0        ;RESTORE HIGH BLOCK NUMBER
                                        BCS     8$              ;BR IF NOT VALID LOGICAL DISK
                                11$:    CLC                     ;SIGNAL SUCCESS ON RETURN
                                        BR      4$
                                ;  Invalid buffer address
                                9$:     MOVB    #-12,INTERR     ;RETURN ERROR CODE 12
                                        BR      6$
                                ;  Invalid block number (return EOF status)
                                8$:
                                ;       BIS     #CS$EOF,(R3)    ;SET END-OF-FILE FLAG IN CSW
                                7$:     CLRB    INTERR          ;RETURN ERROR CODE OF 0
                                6$:     SEC                     ;SIGNAL ERROR ON RETURN
                                ;
                                ;  Copy Channel Status Word from user's channel status block into
                                ;  internal CSW word in I/O queue element (Q.ICSW).
                                ;  (Note, Do not alter C-flag)
                                ;
                                4$:     MOV     C.CSW(R3),Q.ICSW(R1);Copy CSW into Q element
                                ;
                                ;  Finished
                                ;
                                        MOV     (SP)+,R4
                                        MOV     (SP)+,R3
                                        MOV     (SP)+,R2
                                        RETURN
```

```
                                        .SBTTL  SETCR  -- Set completion routine address in queue element
                                ;-------------------------------------------------------------------------
                                ;  SETCR is called to store a completion routine address into an
                                ;  I/O queue element.
                                ;
                                ;  Inputs:
                                ;   R0 = Completion routine address
                                ;   R1 = Address of I/O queue element
                                ;
                                ;  Outputs:
                                ;   c-flag set if a fatal error is detected.
                                ;
013000  020027  000001          SETCR:  CMP     R0,#1               ;COMPL RTN ADDR OF 1 ==> .READW/.WRITW
013004  101403                          BLOS    2$                  ;BR IF YES
013006  004737  000000G                 CALL    UACHKW              ;VALIDATE ADDRESS OF COMPLETION ROUTINE
013012  103406                          BCS     1$                  ;BR IF ILLEGAL
013014  010061  000000G         2$:     MOV     R0,Q.COMP(R1)       ;STORE ADDRESS INTO QUEUE ELEMENT
013020  013761  000000G 000000G         MOV     EMTMAP,Q.PA5(R1)    ;SET EMT ENTRY PAR 5 MAPPING VALUE
013026  000241                          CLC                         ;SIGNAL GOOD RETURN
013030  000207                  1$:     RETURN
```

```
                                        .SBTTL  CVTUAD -- Convert user address to physical address
                                ;-------------------------------------------------------------------------------
                                ;  CVTUAD is called to convert a user specified virtual buffer address into
                                ;  the corresponding physical address.
                                ;  The type of conversion done depends on whether the EMT was executed in
                                ;  user or kernel mode:
                                ;
                                ;  User mode EMT:
                                ;   The address is assumed to be within the job's virtual address region.
                                ;
                                ;  Kernel mode EMT:
                                ;   1. If the virtual address is greater than or equal to 140000 it is mapped
                                ;      into the job's context block.
                                ;   2. If the virtual address is greater than or equal to 120000 it is mapped
                                ;      into the system mapped region using the value of EMTMAP which contains
                                ;      the mapping value that kernel PAR 5 had when the EMT was executed.
                                ;   3. If the virtual address = 0 it is mapped to the base of the program
                                ;      space. (Used for reading in SAV files).
                                ;   4. Otherwise the address is assumed to already be physical and is returned
                                ;      unchanged.
                                ;
                                ;  If the address needs to be mapped, the virtual address returned will be
                                ;  in the range 140000-157777 so that it will map through page 6.
                                ;
                                ;  Inputs:
                                ;   R0 = Virtual address to be mapped.
                                ;
                                ;  Outputs:
                                ;   R0 = 16-bit offset within page.
                                ;   R2 = PAR page base.
                                ;
013032  010146                  CVTUAD: MOV     R1,-(SP)
013034  010346                          MOV     R3,-(SP)
013036  113701  000000G                 MOVB    CORUSR,R1       ;GET JOB INDEX NUMBER
                                ;
                                ;  Determine if the EMT was done in user or kernel mode.
                                ;
013042  032737  000000G 000000G         BIT     #UMODE,EMTPS    ;WAS EMT DONE IN USER OR KERNEL MODE?
013050  001455                          BEQ     1$              ;BR IF IN KERNEL MODE
                                ;
                                ;  EMT was done in user mode.
                                ;  Map user virtual address to physical address.
                                ;
013052  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON RUNNING?
013060  001405                          BEQ     8$              ;BR IF NOT
013062  020027  000000G                 CMP     R0,#CXTBAS      ;IS KMON DOING I/O TO JOB CONTEXT AREA?
013066  103051                          BHIS    5$              ;BR IF YES
013070  162700  000000G                 SUB     #KMNBAS,R0      ;SUBTRACT KMON ADDRESS BIAS
013074  010003                  8$:     MOV     R0,R3           ;GET VIRTUAL ADDRESS
013076  005002                          CLR     R2              ;CLEAR FOR SHIFT
013100  073227  000003                  ASHC    #3,R2           ;GET PAR VALUE FROM ADDRESS
013104  006302                          ASL     R2              ;CONVERT TO WORD INDEX
013106  032761  000000G 000000G         BIT     #$UDSPC,LSW11(R1) ;Has user enabled I/D split?
013114  001006                          BNE     4$              ;Check D-space if so
013116  005762  000000G                 TST     RPDR(R2)        ;IS THIS I-space PAGE SPECIALLY MAPPED?
013122  001413                          BEQ     6$              ;BR IF NOT
013124  016202  000000G                 MOV     RPAR(R2),R2     ;GET BASE 64-BYTE BLOCK # FOR THIS PAGE
```

```
     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$:      CMP     R0,#CXTBAS      ;IS ADDRESS IN JOB CONTEXT BLOCK?
     78 013210  103405                          BLO     2$              ;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$:      CMP     R0,#VPAR5       ;IS ADDRESS IN MAPPED SYSTEM REGION?
     90 013230  103405                          BLO     12$             ;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  0000000               ADD     #VPAR6,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
```

```
                                        .SBTTL  LDIO   -- Process I/O to logical disks
                                ;-------------------------------------------------------------------
                                ;  LDIO is called from the SETQ routine in TSEMT when it determines
                                ;  that the queue element being set up is for an I/O operation to
                                ;  a logical disk.
                                ;  This routine converts the logical device #, unit #, and block #
                                ;  into a physical device #, unit #, and block #.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of I/O queue element
                                ;   EMTBLK = Current EMT argument block
                                ;
013312  010446                  LDIO:   MOV     R4,-(SP)
                                ;
                                ;  Change logical device number and unit number into physical
                                ;  device number and unit number.
                                ;
013314  116104  000000G                 MOVB    Q.UNIT(R1),R4   ;GET LOGICAL DISK UNIT NUMBER
013320  042704  177770                  BIC     #^C7,R4         ;GET UNIT # ONLY
013324  006304                          ASL     R4              ;CONVERT TO WORD TABLE INDEX
013326  016400  000000G                 MOV     LDPDEV(R4),R0   ;GET PHYSICAL UNIT # AND DEVICE #
013332  001456                          BEQ     2$              ;BR IF THIS LOGICAL UNIT NOT IN USE
013334  110061  000000G                 MOVB    R0,Q.DEVX(R1)   ;SET PHYSICAL DEVICE INDEX NUMBER
013340  142761  000007  000000G         BICB    #7,Q.UNIT(R1)   ;CLEAR OLD UNIT #
013346  000300                          SWAB    R0              ;GET PHYSICAL UNIT # TO LOW BYTE
013350  150061  000000G                 BISB    R0,Q.UNIT(R1)   ;SET PHYSICAL UNIT #
                                ;
                                ;  See if this is a .SPFUN to get device size or LD unit tables
                                ;
013354  116100  000000G                 MOVB    Q.FUNC(R1),R0   ;IS THIS A .SPFUN REQUEST?
013360  002023                          BGE     1$              ;BR IF NOT
013362  105061  000000G                 CLRB    Q.DEVX(R1)      ;TELL QIO NOT TO DO ANY ACTUAL I/O
013366  120027  000372                  CMPB    R0,#372         ;IS THIS FUNCTION TO GET DEVICE TABLE?
013372  001006                          BNE     5$              ;BR IF NOT
013374  005737  000006G                 TST     EMTBLK+6        ;IS THIS A REQUEST TO UPDATE HANDLER TABLE?
                                ;  RT V5.4 changed logic to: -wcnt as write, +wcnt as read
                                ;  Remove the next line to always read (-wcnt and +wcnt ==> read)
                                ;       BPL     4$              ;IGNORE IT IF YES
013400  000240                          NOP                     ;Reserve room to patch BR back in if needed
013402  004737  013476'                 CALL    LDRST           ;PASS DEVICE TABLE INFO TO USER
013406  000426                          BR      4$              ;FINISHED
013410  120027  000373          5$:     CMPB    R0,#373         ;IS THIS FUNCTION TO GET DEVICE SIZE?
013414  001023                          BNE     4$              ;BR IF NOT -- IGNORE ALL OTHER SPECIAL FUNCTS
013416  016446  000000G                 MOV     LDSIZE(R4),-(SP);GET LOGICAL DISK SIZE
013422  106677  000004G                 MTPD    @EMTBLK+4       ;MOVE INTO USER'S BUFFER
013426  000416                          BR      4$              ;FINISHED
                                ;
                                ;  Check to make sure that the transfer is completely within the
                                ;  area of the logical disk.
                                ;
013430  016100  000000G          1$:     MOV     Q.WCNT(R1),R0   ;GET WORD COUNT FOR TRANSFER
013434  062700  000377                  ADD     #377,R0         ;CONVERT TO BLOCK COUNT
013440  105000                          CLRB    R0
013442  000300                          SWAB    R0
013444  066100  000000G                 ADD     Q.BLKN(R1),R0   ;ADD TRANSFER STARTING BLOCK NUMBER
013450  020064  000000G                 CMP     R0,LDSIZE(R4)   ;IS TRANSFER WITHIN LOGICAL DISK?
013454  101005                          BHI     2$              ;BR IF NOT
```

```
                                        ;
                                        ;  Bias logical block number by base block of logical disk file
                                        ;
013456  066461  0000006 0000006        ADD     LDBASE(R4),Q.BLKN(R1) ;BIAS BLOCK NUMBER
                                        ;
                                        ;  Transfer is ready to go
                                        ;
013464  000241                  4$:     CLC                     ;SIGNAL SUCCESS ON RETURN
013466  000401                          BR      3$
                                        ;
                                        ;  End of file hit by transfer
                                        ;
013470  000261                  2$:     SEC                     ;SIGNAL ERROR ON RETURN
                                        ;
                                        ;  Finished
                                        ;
013472  012604                  3$:     MOV     (SP)+,R4
013474  000207                          RETURN
```

```
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ;  Process the Logical Disk special function code 372 that is used to
                                ;  pass the LD data tables to the user.
                                ;  The information returned consists of 4 tables with one entry in each
                                ;  table for each logical disk unit (0-7).
                                ;  The four tables are:
                                ;    FLAGS (1 word per unit)
                                ;        100000 = Unit is allocated
                                ;        020000 = Unit is read-only
                                ;        003400 = Unit number mask
                                ;        000076 = Device index number mask
                                ;    OFFSET (1 word per unit)
                                ;        Starting block number of logical disk on real disk.
                                ;    SIZE (1 word per unit)
                                ;        Number of blocks allocated for logical disk.
                                ;    NAME (4 words per unit)
                                ;        File spec for logical disk file.
                                ;
013476  010246                  LDRST:  MOV     R2,-(SP)
013500  010346                          MOV     R3,-(SP)
013502  010446                          MOV     R4,-(SP)
                                ;
                                ;  Pass flag table info to user
                                ;
013504  013702  000040G                 MOV     EMTBLK+4,R2     ;Get address of user's buffer
013510  123727  000000G 000001          CMPB    LDVERS,#1       ;LD version format 5.3& earlier?
013516  001406                          BEQ     11$             ;Skip first two words if old format
013520  012746  045640                  MOV     #^RLD ,-(SP)    ;Pass LD device name to user
013524  106622                          MTPD    (R2)+
013526  012746  000000G                 MOV     #MAXLD,-(SP)    ;Pass number of LD units to user
013532  106622                          MTPD    (R2)+
013534  005003                  11$:    CLR     R3              ;Init LD index
013536  016300  000000G         1$:     MOV     LDPDEV(R3),R0   ;Get real disk device # and unit #
013542  001417                          BEQ     2$              ;Br if unit is not allocated
013544  052700  100000                  BIS     #100000,R0      ;Set unit-allocated flag in status
013550  032763  000000G 000000G         BIT     #LD$RON,LDFLAG(R3);Is the unit read-only?
013556  001411                          BEQ     2$              ;Br if not read-only
013560  123727  000000G 000001          CMPB    LDVERS,#1       ;LD version format 5.3& earlier?
013566  001403                          BEQ     22$             ;Use old mask for old format
013570  052700  040000                  BIS     #040000,R0      ;Set read-only status flag (5.4&later)
013574  000402                          BR      2$
013576  052700  020000          22$:    BIS     #020000,R0      ;Set read-only status flag (5.3&earlier)
013602  010046                  2$:     MOV     R0,-(SP)        ;Pass status flags to user
013604  106622                          MTPD    (R2)+
013606  062703  000002                  ADD     #2,R3           ;Increment unit index
013612  020327  000000C                 CMP     R3,#MAXLD*2     ;Done all units?
013616  103747                          BLO     1$              ;Br if more to do
                                ;
                                ;  Offset info table
                                ;
013620  012704  000000G                 MOV     #LDBASE,R4      ;Point to table with base values for disks
013624  012700  000000G                 MOV     #MAXLD,R0       ;Get number of units
013630  012446                  3$:     MOV     (R4)+,-(SP)     ;Pass offset info to user
013632  106622                          MTPD    (R2)+
013634  077003                          SOB     R0,3$
                                ;
                                ;  Size info table
```

```
                                        ;
013636  012704  0000000G                        MOV     #LDSIZE,R4      ;Point to table with size info
013642  012700  0000000G                        MOV     #MAXLD,R0       ;Get # LD units
013646  012446                          4$:     MOV     (R4)+,-(SP)     ;Pass size info to user
013650  106622                                  MTPD    (R2)+
013652  077003                                  SOB     R0,4$
                                        ;
                                        ;  File name table
                                        ;
013654  012704  0000000G                        MOV     #LDNAME,R4      ;Point to table with file name info
013660  012700  000000C                         MOV     #MAXLD*4,R0     ;Get # words to move
013664  012446                          5$:     MOV     (R4)+,-(SP)     ;Pass info to user
013666  106622                                  MTPD    (R2)+
013670  077003                                  SOB     R0,5$
                                        ;
                                        ;  Finished
                                        ;
013672  012604                                  MOV     (SP)+,R4
013674  012603                                  MOV     (SP)+,R3
013676  012602                                  MOV     (SP)+,R2
013700  000207                                  RETURN
```

```
                                        .SBTTL  INDIO  -- Perform I/O for IND data segment
                                ;-------------------------------------------------------------------------
                                ;  INDIO is called for each Read and Write operation when IND is running.
                                ;  It determines if the read/write is directed to the IND data segment
                                ;  overlay and if so, redirects it to the appropriate position in the
                                ;  INDTMP.TSX file.
                                ;  This is done by altering the block number in EMTBLK and changing
                                ;  CHNADR to point to a channel block that is opened to the INDTMP file.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;   CHNNUM = Channel number
                                ;   CHNADR = Address of CSW for channel
                                ;
                                ;  Outputs:
                                ;   If I/O is redirected to INDTMP file...
                                ;   R3 = Address of channel block for INDTMP file.
                                ;   CHNADR = Address of channel block for INDTMP file.
                                ;   EMTBLK+2 = Block # in INDTMP file of area for data segment for this job.
                                ;
013702  010546                 INDIO:  MOV     R5,-(SP)
                                ;
                                ;  Determine if this I/O is directed to the IND data segment overlay
                                ;
013704  023727  000000G 000017          CMP     CHNNUM,#17      ;IS I/O BEING DONE TO IND.SAV FILE?
013712  001034                          BNE     9$              ;BR IF NOT
013714  023737  000002G 000000G        CMP     EMTBLK+2,INDDBL ;ARE WE ACCESSING DATA SEGMENT OVERLAY?
013722  001030                          BNE     9$              ;BR IF NOT
                                ;
                                ;  IND is accessing its data segment overlay.
                                ;  Determine if the data segment has been written to the temp file yet.
                                ;
013724  032761  000000G 000000G        BIT     #$INDDW,LSW6(R1);HAS DATA SEGMENT BEEN WRITTEN TO TEMP FILE?
013732  001007                          BNE     1$              ;BR IF YES
013734  123727  000001G 000010        CMPB    EMTBLK+1,#10    ;IS THIS A READ OR WRITE REQUEST?
013742  001420                          BEQ     9$              ;BR IF READ -- GO TO SAV FILE TILL 1ST WRITE
013744  052761  000000G 000000G        BIS     #$INDDW,LSW6(R1);REMEMBER 1ST WRITE HAS BEEN DONE TO TEMP FILE
                                ;
                                ;  Change info to cause I/O to be directed to SY:INDTMP.TSX file.
                                ;  The position within the INDTMP file is a function of the Job index #.
                                ;
013752  010105                 1$:     MOV     R1,R5           ;GET JOB #
013754  006205                         ASR     R5              ;DIVIDE BY 2
013756  005305                         DEC     R5              ;MAKE RELATIVE TO 0
013760  070537  000000G               MUL     INDDBS,R5       ;TIMES # OF BLOCKS IN THE OVERLAY SEGMENT
013764  010537  000002G               MOV     R5,EMTBLK+2     ;REPLACE BLOCK # FOR I/O OPERATION
013770  013703  000000G               MOV     CXTRMN,R3       ;GET ADDRESS OF JOB'S SIMULATED RMON
013774  062703  000000G               ADD     #R$INTC,R3      ;POINT TO CHANNEL BLOCK FOR INDTMP FILE
014000  010337  000000G               MOV     R3,CHNADR       ;DO I/O TO INDTMP FILE (SET NEW CSW ADDRESS)
                                ;
                                ;  Finished
                                ;
014004  012605                 9$:     MOV     (SP)+,R5
014006  000207                         RETURN
```

```
                                        .SBTTL  EMTTRC -- Trace EMT execution
                                ;-------------------------------------------------------------------
                                ;  EMTTRC is called from TSEMT when each EMT is processed and the
                                ;  EMT TRACE mode is turned on.  The EMT information has been
                                ;  set up in EMTBLK.
                                ;
014010  010246                  EMTTRC: MOV     R2,-(SP)
014012  010346                          MOV     R3,-(SP)
014014  010446                          MOV     R4,-(SP)
014016  010546                          MOV     R5,-(SP)
                                ;
                                ;  Determine if we should trace this EMT
                                ;
014020  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON RUNNING?
014026  001101                          BNE     9$              ;BR IF YES -- DON'T TRACE KMON
014030  032761  000000G 000000G         BIT     #$CCLRN,LSW5(R1);IS CCL RUNNING?
014036  001075                          BNE     9$              ;BR IF YES -- DON'T TRACE CCL
014040  032737  000000G 000000G         BIT     #UMODE,EMTPS    ;WAS EMT EXECUTED IN USER MODE?
014046  001471                          BEQ     9$              ;BR IF NOT -- DON'T TRACE KERNEL EMT'S
                                ;
                                ;  We want to trace this EMT
                                ;  Begin display line
                                ;
014050  012700  000015                  MOV     #15,R0          ;PRINT CR
014054                                  .TTYOUT
014060  012700  000012                  MOV     #12,R0          ;PRINT LF
014064                                  .TTYOUT
                                ;  Print EMT address
014070  013705  000000G                 MOV     EMTADR,R5       ;GET EMT ADDRESS
014074  012703  000005                  MOV     #5,R3           ;PRINT 16-BIT VALUE
014100  004737  014244'                 CALL    PRTOCT
                                ;  Print EMT instruction low-order byte
014104  113705  000000G                 MOVB    CUREMT,R5       ;GET INSTRUCTION ARGUMENT BYTE
014110  012703  000003                  MOV     #3,R3           ;PRINT 8-BITS
014114  004737  014244'                 CALL    PRTOCT          ;PRINT IT
                                ;  Print function code
014120  113705  000001G                 MOVB    EMTBLK+1,R5     ;GET FUNCTION CODE
014124  123727  000000G 000374          CMPB    CUREMT,#374     ;IS THIS AN EMT 374?
014132  001003                          BNE     3$              ;BR IF NOT
014134  162705  000046                  SUB     #E375MX,R5      ;GET 374 FUNCTION CODE
014140  000405                          BR      2$              ;GO PRINT IT
014142  020527  000062          3$:     CMP     R5,#RT11EX      ;IS THIS A TSX EMT?
014146  103402                          BLO     2$              ;BR IF NOT
014150  062705  000016                  ADD     #<100-RT11EX>,R5;GET ORIGINAL FUNCTION CODE VALUE
014154  004737  014244'          2$:     CALL    PRTOCT
                                ;  Print channel number
014160  113705  000000G                 MOVB    EMTBLK,R5       ;GET CHANNEL #
014164  004737  014244'                 CALL    PRTOCT          ;PRINT IT
                                ;  Print EMT argument values
014170  012704  000002G                 MOV     #EMTBLK+2,R4    ;POINT TO ARGUMENT BLOCK
014174  012702  000005                  MOV     #5,R2           ;PRINT 5 ARGUMENT VALUES
014200  010203                          MOV     R2,R3           ;PRINT 16-BITS EACH
014202  012405          1$:     MOV     (R4)+,R5        ;GET ARGUMENT VALUE
014204  004737  014244'                 CALL    PRTOCT          ;PRINT IT
014210  077204                          SOB     R2,1$           ;LOOP IF MORE TO PRINT
                                ;  End of line
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
```

```
                                        .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
                                ;
014244  010346                  PRTOCT: MOV     R3,-(SP)
014246  010446                          MOV     R4,-(SP)
014250  010546                          MOV     R5,-(SP)
                                ;
                                ;  If this is a 16-bit value, print 1st digit now
                                ;
014252  020327  000005                  CMP     R3,#5           ;IS THIS A 16-BIT VALUE?
014256  001007                          BNE     1$              ;BR IF NOT
014260  012700  000030                  MOV     #30,R0          ;GET 60 SHIFTED
014264  006105                          ROL     R5              ;GET 1ST BIT OF VALUE
014266  006100                          ROL     R0              ;INTO R0
014270                                  .TTYOUT                 ;PRINT IT
014274  000404                          BR      2$
                                ;
                                ;  This is an 8-bit value, left justify the 8 bit quantity
                                ;
014276  042705  177400          1$:     BIC     #^C<377>,R5     ;CLEAR ALL BUT 8 BITS
014302  072527  000007                  ASH     #7,R5           ;LEFT JUSTIFY THE VALUE
                                ;
                                ;  Begin loop to print the value
                                ;
014306  012704  000006          2$:     MOV     #6,R4           ;WILL BECOME 60 WHEN SHIFTED
014312  073427  000003                  ASHC    #3,R4           ;MOVE 3 BITS FROM R5 TO R4
014316  010400                          MOV     R4,R0           ;GET TO R0 FOR TTYOUT
014320                                  .TTYOUT                 ;SEND THE CHARACTER
014324  077310                          SOB     R3,2$           ;LOOP IF MORE DIGITS TO PRINT
                                ;
                                ;  Put in 2 spaces following value
                                ;
014326  112700  000040                  MOVB    #' ,R0          ;GET SPACE
014332                                  .TTYOUT                 ;SEND IT
014336                                  .TTYOUT                 ;  "    "
                                ;
                                ;  Finished
                                ;
014342  012605                          MOV     (SP)+,R5
014344  012604                          MOV     (SP)+,R4
014346  012603                          MOV     (SP)+,R3
014350  000207                          RETURN
        000001                          .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)
```

PRTOCT -- Print an octal value

Operating system: RT-11

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

Cross reference table (CREF V05.05)

```
$CCLRN    1-103    58-16
$CFOPN    1-84     14-21
$CTRLO    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
$NOINT    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
```

Cross reference table (CREF V05.05)

```
        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
```

Cross reference table (CREF V05.05)

```
        20-22
CS$NMX   1-142     1-143    17-34    18-21    26-11
CS$OPN   1-143     8-7       9-7     11-8     16-6     16-36    17-9     17-27    18-6     19-9     20-9     26-9
        26-46
CS$RON   1-110     8-54
CS$SPL   1-149     8-48
CSFACT   2-83      5-45#
CSIARE   1-148     1-174
CSIGEN   1-91      3-8
CSISPC   1-91      3-9
CUNLK    2-67      5-19#
CURCP    1-127    36-7     51-11
CUREMT   1-107    58-33     58-38
CVTUAD  52-71     54-32#
CW$FPU   1-132    12-29
CXTBAS   1-113    54-46     54-77
CXTBUF   1-85     38-79
CXTRMN   1-128     4-43     21-19    27-25    27-30    28-8     28-22    28-39    57-47
DATE     2-59     21-17#
DCRD1    1-111     9-84
DCRD2    1-112     9-118
DELETE   1-87      2-8
DETEMT   1-78      2-92
DH$$LB   1-107     8-87
DIABLO   1-162    40-33
DIEARG   1-42
DIEMSG   1-42
DISMNT   1-89      2-95
DOCIN    2-57     14-5#
DOCOPN   1-92      5-7
DOCULK   1-92      5-20
DOEMT    1-70      4-13#
DOGVAL  27-17#    28-55
DOOPAP   1-92      5-13
DORLK    1-92      5-27
DOSFCK   1-93      5-46
DOTLK    1-93      5-34
DOULK1   1-93      5-40
DOWAIT   2-18      2-49    20-5#
DS$DIR   1-110     8-85     8-110    11-30    52-85
DS$NRD   1-143    18-22
DS$RON   1-110     8-59
DS$SFN   1-97     11-16
DS$WON   1-111     9-55
DSTAT    1-90      3-6
DVFLAG   1-109     8-18     9-18
DVSTAT   1-136     8-59     8-85     8-110    9-55     11-16    11-30    18-22    52-85
DX$EBA   1-109     8-18     9-18
E374MX   2-61#
E375MX   1-69      2-46#   58-40
ECO      1-186#
EC1      1-187#   19-27
EC2      1-188#   19-17
EC3      1-189#
EC4      1-190#
EMT16    2-22      4-64#
```

Cross reference table (CREF V05.05)

```
EMT374    2-49#     2-61
EMT376    1-69     37-8#
EMTADR    1-105    37-10     58-29
EMTASP    1-91      1-91
EMTBLK    1-103     4-22      4-32      4-64      7-7       7-16      7-25      8-14      8-25      8-35      8-87      8-119
          8-125     8-142     8-168     9-14      9-26      9-36      9-86      9-101     9-122     11-22     11-26*    11-46
          11-55     11-65     12-6      12-15     12-24     12-42     12-72     13-5      16-27     17-16     19-22     21-5
          21-34     22-5      23-16     23-24     23-31     24-6      24-31     26-17     26-63     27-5      27-11     27-12
          30-17     30-23     32-22*    32-26     32-40     32-54     32-70     33-12     33-16     36-15     38-15     38-24
          38-41     40-44     41-8      41-20     41-46     41-64     42-4      43-9      43-22     44-15     44-35     44-45
          44-53     44-61     46-14     46-23     46-24     47-9      47-11     47-16     48-6      49-11     49-30     49-41
          49-50     52-54     52-56     52-58     52-63     52-70     52-78     52-82     52-102    52-105    52-108    55-35
          55-45*    56-25     57-27     57-35     57-46*    58-37     58-47     58-50
EMTCAD    1-126     35-9      35-14*
EMTCAS    1-127     35-10
EMTERR    1-141
EMTLST    2-8#      2-46      2-62      2-119     4-59
EMTMAP    1-121     23-29     49-45     53-18     54-95
EMTMAX    2-119#    4-23
EMTPLS    1-93      6-8
EMTPLX    2-38      6-6#
EMTPS     1-103     4-34      8-89      8-121*    51-9      52-59     54-38     58-18
EMTRAD    1-126     35-13*
EMTSP     1-91
EMTSPD    2-110     44-11#
EMTTRC    4-18      58-7#
EMTWIN    1-93      5-79
EMTXIT    1-75      2-33      2-35      2-44      2-56      3-10      3-11      5-55      7-27      8-38      8-156     8-174
          9-39      9-145     10-7      10-15     11-18     11-87     12-18     12-43     12-61     12-94     13-23     14-25
          15-47     16-39     17-46     18-37     19-11     20-28     21-21     21-53     22-26     23-42     24-46     24-57
          26-51     26-64     27-32     27-45     28-11     28-24     28-42     30-27     32-76     35-15     36-18     38-106
          38-139    38-155    38-167    40-23     40-50     41-59     41-80     42-19     43-32     44-73     45-25     46-77
          46-101    47-59     48-12     48-19     49-55
EMUNAM    2-105     47-8#
ENTER     1-87      2-10
ESPCPS    2-41      36-7#
EXCJOB    1-107     12-75
EXIT      3-12      29-5#
FAKCMP    1-111     8-37      8-170     9-38
FETCH     1-90      3-7
FPUUSE    1-132     12-31*    12-33*
FRECXT    1-84      38-82
GETBRK    2-93      49-6#
GETCXT    1-85      38-72
GETDSS    1-180     44-51
GETPPN    2-100     40-44#
GETQ      1-71      8-95      9-97      11-41     23-9      49-35
GETTIM    2-25      21-5#
GETUCH    1-72      32-61
GFINFO    1-89      2-103
GTJB      2-24      13-5#
GTJJ      13-23#
GTLICN    1-84      41-90
GTMSBF    32-47     33-9#     33-25
GTSPAC    1-176     31-7
GVAL      2-36      27-5#
```

Cross reference table (CREF V05.05)

```
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
```

Cross reference table (CREF V05.05)

```
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
```

Cross reference table (CREF V05.05)

```
MINTIM    1-167    38-110
MONABT    1-101    15-25
MONEMT    1-90      2-113
MOUNT     1-89      2-94
MRKT      2-26     23-5#
MRKTHD    1-157    23-36     23-37*    24-11     25-11
MS$DTR    1-180    44-52     44-55
MSEND     1-94      7-8
MSGABT    1-101    15-31
MSGCK     1-94      7-17
MSGPRV    7-6       7-15      7-24      7-33#
MSGWT     1-94      7-26
MXJADR    1-132    12-77
MXJIFN   38-16     38-182#
MXJPRI    1-130    27-116    46-93*
MXNGTX   46-43     46-127#
MXPSTX   46-50     46-120#
MXSPTF   46-31     46-106#
MXSYFN   43-10     43-41#
MXTSXV   27-37     27-67#
NEGTAB   46-45     46-124#   46-127
NFRESB    1-100    41-51     41-58
NLCHN     1-92      4-33
NMUMB     1-139    33-10     33-29*
NSPLBL    1-100    41-50
NSPLDV    1-135    18-10     41-48     41-87
NUCHN     1-102     4-38     15-35
NUMDCD    1-111     9-82      9-116
NUMDEV    1-156    17-35
ODTBAS    1-108    12-54     12-56     12-92*    13-10     13-21
OVLBIT    1-148    15-38
OVRHC     1-182     7-8       7-17      7-26      8-91      8-164     15-25     15-31     18-12     30-25     38-72     38-78
         38-82     46-100    52-117
PO$BYP    1-96     48-10
PO$DBG    1-92      4-16
PO$MEM    1-133    28-26     28-44
PO$NAM    1-95     47-22     48-10
PO$OPR    1-95     21-27     32-20
PO$SND    1-96     32-16
PO$SPF    1-96     11-32
PO$SPV    1-97     48-8      48-10
PO$SYS    1-95     27-82     38-79     46-19     48-10
P2$MSG    1-94      7-35
P2$RLK    1-93      5-63
P2$TRM    1-95     44-33
PASLIN    1-125    29-9
PEEK     28-15#    28-56
PKGET    28-23#    28-28
PKLOW    28-18     28-20     28-26#
PMEMT     1-78      2-96
PNAME     1-102    26-41
POKE     28-32#    28-58
POLOW    28-35     28-37     28-44#
POSTAB   46-52     46-110#   46-120
POVAL    28-40#    28-46
PRINT     1-91      3-13
```

Cross reference table (CREF V05.05)

```
PRIVCO   1-103    4-16    11-32   21-27   27-82   28-26   28-44   32-16   32-20   38-76   46-19   47-22
         48-8    48-10*
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#   23-35*  23-38*  24-12*  24-27*  24-50*  25-12*  25-20*  25-28*  33-9*   33-31*  34-13*
        34-25*   51-16*  51-24*
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$IOT   1-109   24-17   25-17
QFREE    1-73     9-150   23-46   24-42   25-21   49-26   49-59
QHDSPN   1-156   22-21
QIO      1-72     8-132    9-110   11-61
QMSG     1-80    32-72   34-9#
QNSPND   1-139   33-23   51-20
QSET     3-15    12-42#
QUME     1-162   40-34
R$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
```

Cross reference table (CREF V05.05)

```
RTDEV     1-87      2-20
RTEMT     1-89      2-98
RTRSUM    1-88      2-51
RTSPND    1-88      2-50
RUNDEV    1-86     45-17
S$CPU     1-171    39-8
S$HICP    1-115    39-28
S$INWT    1-169    39-10
S$IOFN    1-154    39-7
S$IOWT    1-141    39-17     51-19
S$LOW     1-115    39-9
S$MSWT    1-169    39-15
S$NEDQ    1-136    39-18
S$OTFN    1-154    39-27
S$OTLO    1-114    39-6
S$OTWT    1-169    39-11
S$QCCB    1-114    39-20
S$QMIO    1-114    39-19
S$QSPD    1-114    39-21
S$QUSR    1-170    39-16
S$RT      1-115    39-29
S$SFWT    1-169     1-170    39-14
S$SPCB    1-170    39-24
S$SPDB    1-170    39-23
S$SPND    1-169    39-13
S$TMWT    1-155    22-20     39-12
S$TTFN    1-115    39-26
S$TTSC    1-114    39-25
S$TWFN    1-114    39-5      49-38
S$WFM     1-115    39-30
S$WSMB    1-151    33-22     39-22
SB$END    1-138    32-60
SB$LNK    1-139    33-30     34-17     34-19     34-21*    34-24*
SB$PNT    1-139    33-34*
SB$TXT    1-138    32-58     33-33
SCCA      2-37     12-5#
SCHAIN    1-125    29-9
SCHED     1-117    51-25
SDCLOS    1-71     18-12
SDMOVE    1-111     8-164
SDTTM     2-40     21-27#
SERFLG    1-106    10-4      10-6*     10-12     10-14*
SERR      2-53     10-4#
SETC      1-172    41-79     41-89
SETCR     8-126     9-102    11-56     53-13#
SETDSS    1-180    44-56
SETERR    1-75      4-72      5-70      5-78      6-14      7-42      8-10      8-21      8-29      8-57      8-63      8-71
          8-152     9-10      9-21      9-30      9-59      9-67      9-76      9-132     9-141     9-152     11-11     11-25
         11-37     11-78     11-83     16-12     16-41     17-12     17-53     18-33     20-12     20-24     21-30     23-48
         24-52     26-56     27-9      27-24     27-40     28-51     30-32     32-19     32-43     32-50     32-81     36-10
         36-14     38-19     38-30     38-37     41-34     43-13     44-29     44-40     46-22     46-82     47-25     47-62
         48-14     49-61
SETMXP   46-88#    46-105
SETPRI    1-89      2-106
SETQ      8-100     9-99     11-50     52-16#
SETRBF    1-176    31-4
```

Cross reference table (CREF V05.05)

```
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#
```

Cross reference table (CREF V05.05)

```
TGUCLB  27-59     27-131#
TGVERS  27-65     27-156#
TIMWAT   2-28     22-5#
TRMEMT   2-97     40-8#
TRNSTR   1-140    34-30     44-68
TRPSET   2-11     12-15#
TRYLK    2-69      5-33#
TSEM2    1-23#     1-70     14-22
TSXEMT   2-66#
TSXGVC  27-42     27-49#    27-67
TSXGVL  27-20     27-36#
TSXSIT   1-97     27-106
TSXVRS   1-86     27-156
TTEMT    2-108    30-17#
TTFUN   30-24     31-4#
TTOPTS   1-137
TTREAD   1-111     9-45
TTYIN    1-90      3-4
TTYOUT   1-90      3-5
UACHKB   1-112    52-61     52-68
UACHKW   1-71     23-17     23-25     24-33     49-42     53-15
UCHAN    1-148    15-46*    26-63*
UCLBLK   1-106    27-131
UCLDAT   1-179    43-39
UFPTRP   1-128    12-26*
ULK1BK   2-77      5-39#
UMODE    1-141     4-34      8-89     51-9      52-59     54-38     58-18
UPDATE  21-11     21-19#
UPPN     1-100    46-125    46-126
URO      1-135     6-13*     8-34*     8-61*     8-119     9-35*     9-57*     9-74*     9-86*     9-139*    10-5*     10-13*
        12-5*     12-42*    12-49     12-58*    12-82*    21-17*    26-21*    27-31*    27-44*    28-9*     28-23*    28-40*
        40-22*    41-31*    41-39*    41-56*    41-58*    45-24*    46-61*    49-20*    49-23*    52-77*    52-106*
UTRPAD   1-106    12-17*
VALADB   1-104    12-25     32-56
VALADW   1-104    12-7      12-16     13-6      16-28     17-17     21-6      21-36     22-6      26-19     36-16     38-43
        40-45     43-24     47-10
VIMAGE   1-97     12-87
VLSEMT   1-86      2-116
VMAXMC   1-101     7-33     15-29
VMXSF    1-93      5-61
VMXWIN   1-93      5-75
VPAR5    1-115    54-89
VPAR6    1-100    54-71     54-108    54-118
VPLAS    1-99      6-6
VPRIDF   1-131    46-92
VPRIHI   1-131    27-126
VPRILO   1-131    27-121
VSWPFL   1-131    12-73
VT100    1-161    40-28
VT2007   1-162    40-35
VT2008   1-162    40-36
VT52     1-161    40-13     40-27
VTSLCH   1-129    27-77     45-11
WRITE    1-70      2-17      8-5#
WRITTT   1-109     8-44
WRTCPL   8-168#    9-87
```

Cross reference table (CREF V05.05)

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				
XHISET	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				

Cross reference table (CREF V05.05)

```
...CM3   15-40
...CM5   58-25    58-27    58-58    58-60    59-20    59-33    59-39    59-40
.PURGE    1-65#   15-40
.TTYOU    1-65#   58-25    58-27    58-58    58-60    59-20    59-33    59-39    59-40
DIE       1-43#
DISABL    1-28#   23-35    24-12    25-12    33-9     34-13    51-16
ENABL     1-32#   23-38    24-27    24-50    25-20    25-28    33-31    34-25    51-24
OCALL     1-54#    7-8      7-17     7-26     8-91     8-164    15-25    15-31    18-12    38-72    38-78    38-82
         46-100   52-117
```