TSEM2  TSX-Plus EMT Overlay  MACRO V05.04

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.
                                ;
                                ;  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,CLASN
                        .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
                        .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
                        .GLOBL  INTPRI,UPPN,VPAR6,NFRESB,LACTIV
                        .GLOBL  SYTIMH,SYTIML,SYSDAT,SETSPD,VMAXMC,MSGABT,MONABT
                        .GLOBL  PNAME,MAXQVL,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,MAPPLS
                        .GLOBL  RPAR,RPDR,ODTBAS,CLRDIR
                        .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,$DBGMD
                        .GLOBL  EMTMAP,CHNNUM,LJSW,LSTHL
                        .GLOBL  CLOSE,INDDBL,INDDBS,R$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
                        .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
                        .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
```

```
                                        .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  002732'                .WORD   TRPSET          ;  3  SET TRAP
000012  000000G               .WORD   RENAME          ;  4  RENAME
000014  003470'                .WORD   SVSTAT          ;  5  SAVE STATUS
000016  003614'                .WORD   REOPEN          ;  6  REOPEN
000020  000000G               .WORD   CLOSE           ;  7  CLOSE
000022  001664'                .WORD   READ            ; 10  READ(C)(W)
000024  001076'                .WORD   WRITE           ; 11  WRITE(C)(W)
000026  004100'                .WORD   DOWAIT          ; 12  WAIT
000030  000000G               .WORD   BADEMT          ; 13  COPY CHANNEL
000032  000000G               .WORD   RTDEV           ; 14  DEVICE
000034  005154'                .WORD   CDFN            ; 15  DEFINE CHANNELS
000036  000544'                .WORD   EMT16           ; 16  GROUP 16 EMTS
000040  000000G               .WORD   BADEMT          ; 17
000042  003224'                .WORD   GTJB            ; 20  GTJB
000044  004160'                .WORD   GETTIM          ; 21  GET TIME OF DAY
000046  004404'                .WORD   MRKT            ; 22  MARK TIME
000050  004544'                .WORD   CMKT            ; 23  CANCEL MARK TIME
000052  004320'                .WORD   TIMWAT          ; 24  TIMED WAIT
000054  000000G               .WORD   BADEMT          ; 25  SEND DATA
000056  000000G               .WORD   BADEMT          ; 26  RECEIVE DATA
000060  005032'                .WORD   CHSTAT          ; 27  CHANNEL STATUS
000062  002752'                .WORD   SFPA            ; 30  SET FPP EXCEPTION
000064  000000G               .WORD   EMTXIT          ; 31  PROTECT VECTORS
000066  002422'                .WORD   SPFUN           ; 32  SPECIAL DEV FUNCTIONS
000070  000000G               .WORD   EMTXIT          ; 33  SET EXTRA SWAP ADDRESSES
000072  005166'                .WORD   GVAL            ; 34  .GVAL
000074  002706'                .WORD   SCCA            ; 35  .SCCA
000076  000740'                .WORD   EMTPLX          ; 36  .CRRG, .CRAW, .MAP, etc.
000100  000000G               .WORD   BADEMT          ; 37  Multi-terminal support
000102  004232'                .WORD   SDTTM           ; 40  .SDTTM
000104  007000'                .WORD   ESPCPS          ; 41  .SPCPS
000106  000000G               .WORD   SFDATE          ; 42  .SFDAT -- Set file date
000110  000000G               .WORD   SFPROT          ; 43  .FPROT -- Set file protection
        000044          E375MX  =       <.-EMTLST>/2
                                ;
                                ;  EMT 374 FUNCTIONS
000112  004100'         EMT374: .WORD   DOWAIT          ;  0  WAIT
000114  000000G               .WORD   RTSPND          ;  1  SUSPEND THIS JOB
000116  000000G               .WORD   RTRSUM          ;  2  RESUME THIS JOB
000120  003764'                .WORD   PURGE           ;  3  PURGE A CHANNEL
000122  002400'                .WORD   SERR            ;  4  SET SOFT ERRORS
000124  002412'                .WORD   HERR            ;  5  SET HARD ERRORS
000126  000000G               .WORD   CLOSE           ;  6  CLOSE
000130  000000G               .WORD   EMTXIT          ;  7  TEST & LOCK USR
000132  003306'                .WORD   DOCIN           ; 10  CHAIN
000134  000000G               .WORD   BADEMT          ; 11  MESSAGE WAIT
000136  004206'                .WORD   DATE            ; 12  .DATE
```

Branch vector for processing routines

```
                                        .WORD   ABTIO           ; 13  .ABTIO
                        000014  E374MX  =       <.-EMT374>/2
                        000060  RT11EX  =       <.-EMTLST>/2
                                ;
                                ;  EMT'S DEFINED BY TSX.
                                ;
000142  000566'         TSXEMT: .WORD   CLKOPN           ; 100 Open shared file channel
000144  000606'                 .WORD   CUNLK            ; 101 Unlock shared file channel
000146  000616'                 .WORD   RLOCK            ; 102 Lock a specific block
000150  000626'                 .WORD   TRYLK            ; 103 Try to lock a block
000152  000766'                 .WORD   XMSEND           ; 104 Send a message
000154  001006'                 .WORD   XMSGCK           ; 105 See if message is pending
000156  001026'                 .WORD   XMSGWT           ; 106 Wait for a message
000160  010150'                 .WORD   XSPLSP           ; 107 Get # of free spool blocks
000162  010032'                 .WORD   XGTLN            ; 110 Get TSX line #
000164  010162'                 .WORD   XODTMD           ; 111 Set/reset ODT mode
000166  010612'                 .WORD   XGTTAB           ; 112 Get table pointer
000170  000636'                 .WORD   ULK1BK           ; 113 Unlock a single shared file block
000172  000000                  .WORD   XHIOUT           ; 114 High-efficiency tty output
000174  000000                  .WORD   XHIIN            ; 115 High-efficiency tty input
000176  000000                  .WORD   XTERCK           ; 116 Check for TTY input error
000200  000000                  .WORD   XRDTIM           ; 117 Set TT read timeout
000202  000000                  .WORD   XHISET           ; 120 Turn high-efficiency mode on/off
000204  000646'                 .WORD   CSFACT           ; 121 Check if shared file has been modified
000206  000656'                 .WORD   SFSPND           ; 122 Suspend shared file channel
000210  010224'                 .WORD   CKINAC           ; 123 Check for input activation chars
000212  010242'                 .WORD   GTLICN           ; 124 Get TSX license number
000214  000576'                 .WORD   CAPOPN           ; 125 Open shared file with access protection
000216  000000                  .WORD   KMNEMT           ; 126 KMON EMT's
000220  006262'                 .WORD   SNDMSG           ; 127 Send a message to a line
000222  010664'                 .WORD   TBLEMT           ; 130 Access internal tsx table
000224  011364'                 .WORD   REQPRV           ; 131 Request operator privilege
000226  000000                  .WORD   DETEMT           ; 132 Detached job control emt
000230  011444'                 .WORD   GETBRK           ; 133 Establish break sentinal control
000232  000000                  .WORD   MOUNT            ; 134 Mount a file structure
000234  000000                  .WORD   DISMNT           ; 135 Dismount a file structure
000236  000000                  .WORD   PMEMT            ; 136 Performance monitor EMT's
000240  007672'                 .WORD   TRMEMT           ; 137 Get terminal type
000242  000000                  .WORD   RTEMT            ; 140 Real-time EMT's
000244  003102'                 .WORD   SETSIZ           ; 141 TSX-Plus Settop
000246  010002'                 .WORD   GETPPN           ; 142 Determine Project-Programmer number
000250  000000                  .WORD   SSEMT            ; 143 Control shared run-time systems
000252  007124'                 .WORD   JBINFO           ; 144 Get information about a specific line
000254  000000                  .WORD   GFINFO           ; 145 Get information about a file
000256  000000                  .WORD   SFTIME           ; 146 Set file creation time
000260  011172'                 .WORD   EMUNAM           ; 147 Get or set user name
000262  000000                  .WORD   SETPRI           ; 150 Set job execution priority
000264  000000                  .WORD   SPLEMT           ; 151 Spooling control for current channel
000266  006120'                 .WORD   TTEMT            ; 152 Terminal control
000270  010252'                 .WORD   INTEMT           ; 153 Set interactive/non-interactive mode
000272  010376'                 .WORD   EMTSPD           ; 154 Set terminal transmit/receive speed
000274  000000                  .WORD   CLASN            ; 155 Assign a CL unit to a line
000276  000000                  .WORD   ALCEMT           ; 156 Allocate or deallocate a device
000300  000000                  .WORD   MONEMT           ; 157 Monitor status of a job
000302  000000                  .WORD   CPYEMT           ; 160 Copy file/priv info from another job
000304  000720'                 .WORD   CKWIN            ; 161 Display window control
000306  000000                  .WORD   VLSEMT           ; 162 Switch between processes
```

Branch vector for processing routines

```
    115 000310  000000G                        .WORD   LDMEMT          ; 163 Mount a logical disk structure
    116 000312  010306'                        .WORD   SYFEMT          ; 164 Get system file dev:filspc.ext
    117         000145              EMTMAX  =       <.-EMTLST>/2
```

Branch vector for processing routines

```
                                ;-------------------------------------------------------------
                                ;  BRANCH VECTOR FOR EMTS IN THE RANGE 340 TO 357
                                ;
000314  000000G       LST16:    .WORD   TTYIN           ; 340  TTYIN
000316  000000G                 .WORD   TTYOUT          ; 341  TTYOUT
000320  000000G                 .WORD   DSTAT           ; 342  DSTATUS
000322  000000G                 .WORD   FETCH           ; 343  FETCH/REL
000324  000000G                 .WORD   CSIGEN          ; 344  CSIGEN
000326  000000G                 .WORD   CSISPC          ; 345  CSISPC
000330  000000G                 .WORD   EMTXIT          ; 346  LOCK USR
000332  000000G                 .WORD   EMTXIT          ; 347  RELEASE USR
000334  006034'                 .WORD   EXIT            ; 350  EXIT
000336  000000G                 .WORD   PRINT           ; 351  PRINT
000340  003370'                 .WORD   SRESET          ; 352  SRESET
000342  003016'                 .WORD   QSET            ; 353  QSET
000344  003030'                 .WORD   SETTOP          ; 354  SETTOP
000346  003340'                 .WORD   RCTRLO          ; 355  RCTRLO
000350  006746'                 .WORD   ASTXIT          ; 356  Completion routine exit
000352  003362'                 .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.
                                ;
000354  113701  000000G         DOEMT:  MOVB    CORUSR,R1       ;GET CURRENT JOB INDEX NUMBER
000360  032761  000000G 000000G         BIT     #$EMTTR,LSW6(R1);IS EMT TRACING WANTED?
000366  001406                         BEQ     2$              ;BR IF NOT
000370  032737  000000G 000000G         BIT     #PO$DBG,PRIVCO  ;Are we authorized to do EMT tracing?
000376  001402                         BEQ     2$              ;Br if not
000400  004737  013456'               CALL    EMTTRC          ;TRACE THE EMT
                                ;
                                ;  Make sure the EMT function code is ok.
                                ;
000404  113705  000001G         2$:     MOVB    EMTBLK+1,R5     ;GET FUNCTION CODE
000410  020527  000145                 CMP     R5,#EMTMAX      ;FUNCTION CODE OK?
000414  103402                         BLO     5$              ;BR IF OK
000416  000137  000000G                JMP     BADEMT          ;INVALID EMT FUNCTION CODE
                                ;
                                ;  Make sure channel number is ok.
                                ;
000422  120527  000120          5$:     CMPB    R5,#<140+<RT11EX-100>> ;IS THIS TSX REAL-TIME EMT?
000426  001443                         BEQ     4$              ;BR IF YES -- ALLOW ANY CHANNEL NUMBER
000430  005003                         CLR     R3
000432  153703  000000G                BISB    EMTBLK,R3       ;GET CHANNEL # WITHOUT SIGN EXTENSION
000436  012700  000000G                MOV     #NLCHN,R0       ;MAX # CHANNELS FOR EACH JOB
000442  032737  000000G 000000G         BIT     #UMODE,EMTPS    ;WAS EMT DONE IN USER OR KERNEL MODE?
000450  001406                         BEQ     3$              ;BR IF KERNEL MODE
000452  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON RUNNING NOW?
000460  001002                         BNE     3$              ;BR IF YES -- ALLOW ACCESS TO ALL CHANNELS
000462  012700  000000G                MOV     #NUCHN,R0       ;USER PROGRAM -- RESTRICT TO CHANS 0-15.
000466  020300                 3$:     CMP     R3,R0           ;IS THIS CHANNEL # TOO BIG?
000470  103032                         BHIS    CHNERR          ;BR IF YES
                                ;  Save address of CSW for channel.
000472  010337  000000G                MOV     R3,CHNNUM       ;SAVE USER'S CHANNEL NUMBER
000476  013700  000000G                MOV     CXTRMN,R0       ;GET ADDRESS OF SIMULATED RMON
000502  062700  000000G                ADD     #R$CHN,R0       ;POINT TO 1ST CHANNEL AREA
000506  020327  000021                 CMP     R3,#21          ;IS THIS CHANNEL IN EXTENDED CHANNEL AREA?
000512  103404                         BLO     1$              ;BR IF NOT
000514  162703  000021                 SUB     #21,R3          ;GET CHANNEL # IN EXTENDED AREA
000520  062700  000000C                ADD     #R$XCHN-R$CHN,R0;POINT TO EXTENDED CHANNEL AREA
000524  070327  000000G          1$:     MUL     #CHNSIZ,R3      ;MULTIPLY BY # BYTES PER CHANNEL
000530  060003                         ADD     R0,R3           ;GET ABSOLUTE ADDRESS OF CHANNEL AREA
000532  010337  000000G                MOV     R3,CHNADR       ;SAVE CURRENT CHANNEL ADDRESS
                                ;
                                ;  Enter EMT processing routine.
                                ;  R1 contains user index number.
                                ;  UR0 contains contents of user's R0, put value here to return in R0.
                                ;
000536  006305                  4$:     ASL     R5              ;GET 2* FUNCTION CODE VALUE
```

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

```
                                        .SBTTL  Shared file control EMT's
                                ;-------------------------------------------------------
                                ;  PROCESS THE .CLKOPN EMT WHICH IS USED TO OPEN A CHANNEL
                                ;  TO A SHARED FILE.
                                ;
000566  004737  000672'        CLKOPN: CALL    CHKSF           ;See if shared file support is available
000572  000177  000000G               JMP     @DOCOPN         ;Do the open
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Open a shared file with access protection.
                                ;
000576  004737  000672'        CAPOPN: CALL    CHKSF           ;See if shared file support is available
000602  000177  000000G               JMP     @DOOPAP         ;Open the channel
                                ;
                                ;-------------------------------------------------------
                                ;  PROCESS THE .CUNLK EMT WHICH IS USED TO UNLOCK
                                ;  A SHARED FILE CHANNEL.
                                ;
000606  004737  000672'        CUNLK:  CALL    CHKSF           ;See if shared file support is available
000612  000177  000000G               JMP     @DOCULK         ;Unlock all blocks on channel
                                ;
                                ;-------------------------------------------------
                                ;  PROCESS THE .RLOCK EMT WHICH IS USED TO LOCK A
                                ;  SPECIFIC BLOCK.
                                ;
000616  004737  000672'        RLOCK:  CALL    CHKSF           ;See if shared file support is available
000622  000177  000000G               JMP     @DORLK          ;Lock the block
                                ;
                                ;-------------------------------------------------
                                ;  PROCESS THE .TRYLK EMT WHICH IS USED TO TRY TO LOCK
                                ;  A SPECIFIC BLOCK.
                                ;
000626  004737  000672'        TRYLK:  CALL    CHKSF           ;See if shared file support is available
000632  000177  000000G               JMP     @DOTLK          ;Try to lock the block
                                ;
                                ;-----------------------------------------------------------
                                ;  UNLOCK A SPECIFIC SHARED FILE BLOCK.
                                ;
000636  004737  000672'        ULK1BK: CALL    CHKSF           ;See if shared file support is available
000642  000177  000000G               JMP     @DOULK1         ;Unlock a block
                                ;
                                ;-----------------------------------------------------------
                                ;  CHECK FOR WRITES TO A SHARED FILE BY OTHER USERS.
                                ;
000646  004737  000672'        CSFACT: CALL    CHKSF           ;See if shared file support is available
000652  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.
                                ;
000656  004737  000672'        SFSPND: CALL    CHKSF           ;See if shared file support is available
000662  004777  000000G               CALL    @SFSVST         ;TELL SHARED FILE LOGIC
000666  000137  000000G               JMP     EMTXIT
                                ;
                                ;-----------------------------------------------------------------------
```

Shared file control EMT's

```
                                        ;  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.
                                        ;
000672  005737  0000000         CHKSF:  TST     VMXSF           ;Is shared file support genned in?
000676  001405                          BEQ     10$             ;Br if not
000700  032737  0000000 0000000         BIT     #P2$RLK,PRIVC2  ;Are we privileged to access shared files?
000706  001401                          BEQ     10$             ;Br if not
000710  000207                          RETURN
                                        ;
                                        ;  We cannot use shared file support
                                        ;
000712  005000                  10$:    CLR     R0              ;Return error code 0
000714  000137  0000000                 JMP     SETERR
                                        ;-------------------------------------------------------------------------------
                                        ;  Display window control EMT's
                                        ;
000720  005737  0000000         CKWIN:  TST     VMXWIN          ;Is window support genned in?
000724  001003                          BNE     1$              ;Br if yes
000726  005000                          CLR     R0              ;Return error code 0
000730  000137  0000000                 JMP     SETERR
000734  000137  0000000         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.
                                ;
000740  005737  000000G        EMTPLX: TST     VPLAS           ;Was PLAS support generated into system?
000744  001402                        BEQ     9$              ;Br if not
000746  000137  000000G                JMP     EMTPLS          ;Enter TSPLAS overlay
                                ;
                                ;  Error: PLAS support not generated into system
                                ;
000752  012700  000007        9$:     MOV     #7,R0           ;Return error code 7
000756  005037  000000G                CLR     UR0             ;Return 0 in R0
000762  000137  000000G                JMP     SETERR
```

```
                                        .SBTTL  Message communication EMT's
                                ;-----------------------------------------------------------------
                                ;  PROCESS THE .MSEND EMT WHICH QUEUES A MESSAGE FOR
                                ;  ANOTHER USER.
                                ;
000766  004737  001050'        XMSEND: CALL    MSGPRV          ;Do we have access to message facility?
000772  012700  000000G                MOV     #EMTBLK,R0      ;Point to EMT arg block
000776                                  OCALL   MSEND           ;SEND THE MESSAGE
001004  000417                         BR      CHKERR          ;CHECK FOR ERRORS
                                ;
                                ;-----------------------------------------------------------
                                ;  PROCESS THE .MSGCK EMT WHICH IS USED TO CHECK FOR
                                ;  A PENDING MESSAGE.
                                ;
001006  004737  001050'        XMSGCK: CALL    MSGPRV          ;Do we have access to message facility?
001012  012700  000000G                MOV     #EMTBLK,R0      ;POINT TO EMT ARG BLOCK
001016                                  OCALL   MSGCK           ;CHECK FOR MESSAGE
001024  000407                         BR      CHKERR          ;GO CHECK FOR ERRORS
                                ;
                                ;-----------------------------------------------------------
                                ;  PROCESS THE .MSGWT EMT WHICH IS USED TO WAIT FOR
                                ;  A MESSAGE FROM ANOTHER USER.
                                ;
001026  004737  001050'        XMSGWT: CALL    MSGPRV          ;Do we have access to message facility?
001032  012700  000000G                MOV     #EMTBLK,R0      ;POINT TO EMT ARG BLOCK
001036                                  OCALL   MSGWT           ;WAIT FOR THE MESSAGE
001044  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.
                                ;
001050  005737  000000G        MSGPRV: TST     VMAXMC          ;Is message facility genned into system?
001054  001405                         BEQ     10$             ;Br if not
001056  032737  000000G 000000G        BIT     #P2$MSG,PRIVC2  ;Are we authorized to use message facility?
001064  001401                         BEQ     10$             ;Br if not
001066  000207                         RETURN
                                ;
                                ;  We cannot access message facility
                                ;
001070  005000                 10$:    CLR     R0              ;Return error code 0
001072  000137  000000G                JMP     SETERR
```

```
                                        .SBTTL  .WRITE
                                ;-------------------------------------------------------------------------------
                                ;  Process a .WRITE EMT.
                                ;
001076  004737  011664'         WRITE:  CALL    CKIOST          ;SEE IF SCHEDULER WANTS TO STOP I/O
001102  013703  000000G                 MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL
001106  032763  000000G 000000G         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
001114  001004                         BNE     6$              ;BR IF YES
001116  012700  000002                 MOV     #2,R0           ;CHANNEL NOT OPEN
001122  000137  000000G                 JMP     SETERR
                                ;
                                ;  Check to make sure the buffer is on a word boundary
                                ;
001126  032737  000001  000004G 6$:     BIT     #1,EMTBLK+4     ;Is buffer address even?
001134  001414                         BEQ     15$             ;Br if yes
001136  016302  000000G                 MOV     C.CSW(R3),R2    ;Get CSW
001142  042702  177701                 BIC     #^C76,R2        ;Get device table index
001146  032762  000000G 000000G         BIT     #DX$EBA,DVFLAG(R2);Does this device require even buffer addr?
001154  001404                         BEQ     15$             ;Br if not
001156  012700  177766                 MOV     #-12,R0         ;Get abort code
001162  000137  000000G                 JMP     SETERR          ;Abort the operation
                                ;
                                ;  Make sure word count is ok
                                ;
001166  005737  000006G         15$:    TST     EMTBLK+6        ;Is word count negative (>32727.)?
001172  001405                         BEQ     17$             ;Br if word count zero
001174  100016                         BPL     18$             ;Branch if valid size (positive)
001176  012700  177766                 MOV     #-12,R0         ;Get abort code
001202  000137  000000G                 JMP     SETERR          ;Abort the operation
                                ;
                                ;  Word count is zero ==> Seek.
                                ;  Treat seek like nop but call completion routine if requested.
                                ;
001206  005037  000000G         17$:    CLR     @R0             ;Say zero words read
001212  023727  000010G 000001          CMP     EMTBLK+10,#1    ;Need to run completion routine?
001220  101402                         BLOS    19$             ;Br if not
001222  004737  000000G                 CALL    FAKCMP          ;Call completion routine
001226  000137  000000G         19$:    JMP     EMTXIT          ;Finished with read
                                ;
                                ;  See if we are writing to TT:
                                ;
001232  004737  011640'         18$:    CALL    CHKTT           ;IS CHANNEL OPENED TO "TT:"?
001236  103402                         BCS     10$             ;BR IF NOT WRITING TO TT
001240  000137  000000G                 JMP     WRITTT          ;ENTER THE TT WRITE ROUTINE IN TSTTY
                                ;
                                ;  See if we are writing to a spooled device.
                                ;
001244  032763  000000G 000000G 10$:    BIT     #CS$SPL,C.CSW(R3);IS THIS A SPOOLED DEVICE?
001252  001171                         BNE     WRTSPL          ;BR IF IT IS A SPOOLED DEVICE
                                ;
                                ;  See if we are attempting to write to a read-only device.
                                ;
001254  016302  000000G                 MOV     C.CSW(R3),R2    ;GET CSW
001260  032702  000000G                 BIT     #CS$RON,R2      ;IS USER AUTHORIZED FOR READ-ONLY ACCESS?
001264  001404                         BEQ     9$              ;BR IF NOT READ-ONLY
001266  012700  177763                 MOV     #-15,R0         ;SET ABORT ERROR CODE
001272  000137  000000G                 JMP     SETERR          ;ABORT EMT
```

```
     58 001276  042702  177701           9$:     BIC     #^C76,R2          ;GET DEVICE TABLE INDEX
     59 001302  032762  0000000 0000000          BIT     #DS$RON,DVSTAT(R2);IS THIS A READ-ONLY DEVICE?
     60 001310  001405                          BEQ     5$                ;BR IF NOT READ-ONLY
     61 001312  005037  0000000          8$:     CLR     URO               ;SAY 0 WORDS WRITTEN
     62 001316  005000                          CLR     R0                ;RETURN ERROR CODE OF 0
     63 001320  000137  0000000                 JMP     SETERR
     64                                     ;
     65                                     ;  See if error status is pending in CSW for the channel
     66                                     ;
     67 001324  032763  0000000 0000000 5$:     BIT     #CS$ERR,C.CSW(R3);DID A WRITE ERROR OCCUR?
     68 001332  001407                          BEQ     20$               ;BR IF NOT
     69 001334  042763  0000000 0000000          BIC     #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
     70 001342  012700  000001                  MOV     #1,R0             ;RETURN ERROR CODE OF 1
     71 001346  000137  0000000                 JMP     SETERR
     72                                     ;
     73                                     ;  Check if this is an I/O operation by IND
     74                                     ;
     75 001352  032761  0000000 0000000 20$:    BIT     #$INDRN,LSW5(R1);IS IND RUNNING?
     76 001360  001402                          BEQ     11$               ;BR IF NOT
     77 001362  004737  013350'                 CALL    INDIO             ;MAP I/O FOR IND
     78                                     ;
     79                                     ;  If this is a write by a user-mode program which has opened a file
     80                                     ;  structured device in non-file-structured mode, clear the directory
     81                                     ;  cache for the device.
     82                                     ;
     83 001366  005763  0000000          11$:    TST     C.SBLK(R3)        ;Is device opened non-file-structured mode?
     84 001372  001017                          BNE     16$               ;Br if not
     85 001374  032762  0000000 0000000          BIT     #DS$DIR,DVSTAT(R2);Is this a directory structured device?
     86 001402  001413                          BEQ     16$               ;Br if not
     87 001404  023727  0000020 0000000          CMP     EMTBLK+2,#DH$$LB;Could this block contain directory?
     88 001412  101007                          BHI     16$               ;Br if not
     89 001414  032737  0000000 0000000          BIT     #UMODE,EMTPS      ;Was EMT executed in user mode?
     90 001422  001403                          BEQ     16$               ;Br if not -- Probably TSUSR doing dir I/O
     91 001424                                  OCALL   CLRDIR            ;Clear out directory cache for device
     92                                     ;
     93                                     ;  Build an I/O queue entry for the write request.
     94                                     ;
     95 001432  004737  0000000          16$:    CALL    GETQ              ;GET AN I/O QUEUE ENTRY
     96                                     ;
     97                                     ;  R1 now points to a free I/O queue entry.
     98                                     ;  Move arguments from EMTBLK to the queue entry.
     99                                     ;
    100 001436  004737  011772'                 CALL    SETQ              ;Do common queue setup
    101 001442  103473                          BCS     3$                ;Br if we must abort operation
    102 001444  005761  0000000                 TST     Q.WCNT(R1)        ;Is word count for transfer zero?
    103 001450  001470                          BEQ     3$                ;Br if yes -- Nothing to write
    104 001452  020063  0000000                 CMP     R0,C.USED(R3)     ;Are we writing beyond previous end point?
    105 001456  101416                          BLOS    2$                ;Br if not
    106 001460  032713  0000000                 BIT     #CS$ENT,(R3)      ;Was file opened with a .ENTER?
    107 001464  001413                          BEQ     2$                ;Br if not
    108 001466  005763  0000000                 TST     C.SBLK(R3)        ;Is this a directory operation?
    109 001472  001004                          BNE     14$               ;Br if not
    110 001474  032762  0000000 0000000          BIT     #DS$DIR,DVSTAT(R2) ;Is this a directory structured device?
    111 001502  001004                          BNE     2$                ;Br if yes
    112 001504  010063  0000000          14$:    MOV     R0,C.USED(R3)     ;Remember high-water-mark within file
    113 001510  010061  000000C                 MOV     R0,Q.ICSW+C.USED(R1);Store into chan blk in Q element too
    114 001514  005461  0000000          2$:     NEG     Q.WCNT(R1)        ;Negative word count ==> write operation
```

```
                                        ;
                                        ;  If we shortened write request because we hit end of file, return error
                                        ;  code of 0 but go ahead and do the write.
                                        ;
001520  023737  000000@ 000006@         CMP     URO,EMTBLK+6    ;DID WE SHORTEN WRITE REQUEST?
001526  001403                          BEQ     7$              ;BR IF NOT
001530  052737  000000@ 000000@         BIS     #CFLAG,EMTPS    ;SET C-FLAG IN PS
                                        ;
                                        ;  Set address of write completion routine.
                                        ;
001536  013700  000010@         7$:     MOV     EMTBLK+10,R0    ;GET ADDRESS OF COMPLETION ROUTINE
001542  004737  012472'                  CALL    SETCR           ;STORE ADDR OF COMPL ROUTINE INTO QUEUE ELEM
001546  103431                          BCS     3$              ;BR IF FATAL ERROR
                                        ;
                                        ;  We have built the I/O queue element.
                                        ;  Initiate the I/O operation.
                                        ;
001550  004737  000000@                 CALL    QIO             ;QUEUE AN I/O REQUEST
                                        ;
                                        ;  See if we are writing to a shared file
                                        ;
001554  005737  000000@                 TST     JCDB            ;Does job have any shared files open?
001560  001402                          BEQ     12$             ;Br if not
001562  004777  000000@                 CALL    @SFWRIT         ;Check for writes to shared files
                                        ;
                                        ;  See if we should wait for the I/O operation to finish.
                                        ;
001566  005737  000010@         12$:    TST     EMTBLK+10       ;IS THIS AN .WRITW?
001572  001015                          BNE     1$              ;BR IF NOT
001574  004737  000000@                 CALL    IOWAIT          ;WAIT FOR I/O TO FINISH
                                        ;
                                        ;  See if a hardware error occured on the write
                                        ;
001600  032763  000000@ 000000@         BIT     #CS$ERR,C.CSW(R3);DID A WRITE ERROR OCCUR?
001606  001407                          BEQ     1$              ;BR IF NOT
001610  042763  000000@ 000000@         BIC     #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
001616  012700  000001                  MOV     #1,R0           ;RETURN ERROR CODE OF 1
001622  000137  000000@                 JMP     SETERR
                                        ;
                                        ;  Finished the EMT.
                                        ;
001626  000137  000000@         1$:     JMP     EMTXIT          ;EXIT FROM EMT
                                        ;
                                        ;  Some fatal error occured which causes us to have to abort the operation.
                                        ;
001632  000137  002364'         3$:     JMP     IOABRT          ;ABORT THE I/O REQUEST
                                        ;
                                        ;  Write to a spooled device.
                                        ;
001636                          WRTSPL: OCALL   SDMOVE          ;WRITE USER'S DATA TO SPOOL FILE
                                        ;
                                        ;  See if we need to queue a completion routine for this write request.
                                        ;
001644  023727  000010@ 000001  WRTCPL: CMP     EMTBLK+10,#1    ;WAS A COMPLETION ROUTINE SPECIFIED?
001652  101402                          BLOS    1$              ;BR IF NOT
001654  004737  000000@                 CALL    FAKCMP          ;QUEUE A COMPLETION ROUTINE REQUEST
                                        ;
```

.WRITE

```
    172                                 ;  Finished spooled write
    173                                 ;
    174 001660  000137  000000          1$:     JMP     EMTXIT
```

```
                                        .SBTTL  .READ
                                ;-------------------------------------------------------------------------
                                ;  Process a .READ EMT.
                                ;
001664  004737  011664'         READ:   CALL    CKIOST          ;SEE IF SCHEDULER WANTS TO STOP I/O
001670  013703  0000006G                 MOV     CHNADR,R3       ;GET ADDRESS OF THE CHANNEL
001674  032763  0000006G 000006G         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
001702  001004                          BNE     2$              ;BR IF YES
001704  012700  000002                  MOV     #2,R0           ;RETURN ERROR CODE
001710  000137  0000006G                 JMP     SETERR
                                ;
                                ;  Check to make sure the buffer is on a word boundary
                                ;
001714  032737  000001  000004G 2$:     BIT     #1,EMTBLK+4     ;Is buffer address even?
001722  001414                          BEQ     15$             ;Br if yes
001724  016302  0000006G                 MOV     C.CSW(R3),R2    ;Get CSW
001730  042702  177701                  BIC     #^C76,R2        ;Get device table index
001734  032762  0000006G 0000006G         BIT     #DX$EBA,DVFLAG(R2);Does this device require even buffer addr?
001742  001404                          BEQ     15$             ;Br if not
001744  012700  177766                  MOV     #-12,R0         ;Get abort code
001750  000137  0000006G                 JMP     SETERR          ;Abort the operation
                                ;
                                ;  See if the read request is so large that
                                ;  a handler might interpret it as a write
                                ;
001754  005737  0000066G        15$:    TST     EMTBLK+6        ;Is word count negative (>32727.)?
001760  001405                          BEQ     17$             ;Br if word count zero
001762  100016                          BPL     14$             ;Branch if valid size (positive)
001764  012700  177766                  MOV     #-12,R0         ;Get abort code
001770  000137  0000006G                 JMP     SETERR          ;Abort the operation
                                ;
                                ;  Word count is zero ==> Seek.
                                ;  Treat seek like nop but call completion routine if requested.
                                ;
001774  005037  0000006G        17$:    CLR     URO             ;Say zero words read
002000  023727  0000106G 000001         CMP     EMTBLK+10,#1    ;Need to run completion routine?
002006  101402                          BLOS    19$             ;Br if not
002010  004737  0000006G                 CALL    FAKCMP          ;Call completion routine
002014  000137  0000006G        19$:    JMP     EMTXIT          ;Finished with read
                                ;
                                ;  See if we are reading from TT:
                                ;
002020  004737  011640'         14$:    CALL    CHKTT           ;IS CHANNEL OPENED TO "TT:"?
002024  103402                          BCS     10$             ;BR IF NOT TT READ
002026  000137  0000006G                 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.
                                ;
002032  016300  0000006G        10$:    MOV     C.CSW(R3),R0    ;GET CSW
                                ;       BIT     #CS$SPL,R0      ;IS THIS A SPOOLED DEVICE?
                                ;       BNE     4$              ;BR IF SPOOLED DEVICE
002036  042700  177701                  BIC     #^C76,R0        ;GET DEVICE TABLE INDEX FOR DEVICE
002042  032760  0000006G 0000006G         BIT     #DS$WON,DVSTAT(R0);IS THIS A WRITE-ONLY DEVICE?
002050  001405                          BEQ     7$              ;BR IF NOT WRITE-ONLY
002052  005037  0000006G        4$:     CLR     URO             ;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?
```

```
                                        BEQ     12$             ;Br if not
                                        TST     NUMDCD          ;Are we doing shared-file data caching?
                                        BEQ     12$             ;Br if not
                                        CALL    @DCRD2          ;See if we need to update data cache
                                ;
                                ;  See if we should wait for the I/O operation to finish.
                                ;
                        12$:    TST     EMTBLK+10       ;SHOULD WE WAIT?
                                BNE     1$              ;BR IF NOT
                                CALL    IOWAIT          ;WAIT FOR I/O OPERATION TO FINISH
                                ;
                                ;  See if a hardware error occured on the read
                                ;
                                BIT     #CS$ERR,C.CSW(R3);DID A HARDWARE ERROR OCCUR ON READ?
                                BEQ     6$              ;BR IF NOT
                                BIC     #CS$ERR,C.CSW(R3);ACKNOWLEDGE ERROR STATUS
                                MOV     #1,R0           ;RETURN ERROR CODE OF 1
                                JMP     SETERR
                                ;
                                ;  See if the device reported end-of-file.
                                ;
                        6$:     BIT     #CS$EOF,C.CSW(R3);DID DEVICE REPORT END OF FILE?
                                BEQ     1$              ;BR IF NOT
                                BIC     #CS$EOF,C.CSW(R3);ACKNOWLEDGE EOF STATUS
                                CLR     UR0             ;SAY 0 WORDS READ
                                CLR     R0              ;RETURN ERROR CODE OF 0
                                JMP     SETERR
                                ;
                                ;  Finished operation.
                                ;
                        1$:     JMP     EMTXIT
                                ;
                                ;  We detected a fatal error before queueing the request.
                                ;  Return the I/O queue entry to the free list.
                                ;
                        IOABRT: CALL    QFREE           ;RETURN I/O QUEUE ENTRY TO FREE LIST
                                MOVB    INTERR,R0       ;GET ERROR CODE
                                JMP     SETERR          ;ERROR RETURN
```

```
                                        .SBTTL  .SERR
                                ;-------------------------------------------------
                                ;  PROCESS THE .SERR EMT
002400  112737  000001  000000G SERR:   MOVB    #1,SERFLG       ;REMEMBER USER ISSUED SERR
002406  000137  000000G                 JMP     EMTXIT

                                        .SBTTL  .HERR
                                ;-------------------------------------------------
                                ;  PROCESS THE .HERR EMT
002412  105037  000000G          HERR:   CLRB    SERFLG          ;TURN OFF .SERR CONTROL
002416  000137  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  .SPFUN
                                ;-------------------------------------------------------------------------------
                                ;  Perform special device functions.
                                ;
002422  004737  011664'          SPFUN:  CALL    CKIOST          ;See if scheduler wants to hold I/O starts
002426  013703  0000006          MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
002432  016302  0000006          MOV     C.CSW(R3),R2    ;GET CSW
002436  032702  0000006          BIT     #CS$OPN,R2      ;IS CHANNEL OPEN?
002442  001004                  BNE     1$              ;BR IF YES
002444  012700  000002           MOV     #2,R0           ;ERROR 2 IF CHANNEL NOT OPEN
002450  000137  0000006          JMP     SETERR
                                ;
                                ;  See if device can accept special functions.
                                ;
002454  042702  177701           1$:     BIC     #177701,R2      ;EXTRACT DEVICE TABLE INDEX #
002460  032762  0000006 0000006          BIT     #DS$SFN,DVSTAT(R2);CAN DEVICE HANDLE SPECIAL FUNCTIONS?
002466  001002                  BNE     2$              ;BR IF YES
002470  000137  0000006          4$:     JMP     EMTXIT          ;IGNORE .SPFUN REQUEST
                                ;
                                ;  Check function code
                                ;
002474  105737  0000116          2$:     TSTB    EMTBLK+11       ;FUNCTION CODE MUST BE NEGATIVE
002500  002403                  BLT     3$              ;BR IF OK
002502  005000                  CLR     R0              ;ERROR CODE 0 IF NOT NEGATIVE FUNCTION
002504  000137  0000006          JMP     SETERR
002510  112737  000377  0000106 3$:     MOVB    #377,EMTBLK+10  ;MAKE SURE EVEN BYTE = 377
                                ;
                                ;  Make sure user is privileged to do .SPFUN's
                                ;
002516  032762  0000006 0000006          BIT     #DS$DIR,DVSTAT(R2) ;Is this a directory-structured device?
002524  001413                  BEQ     7$              ;Br if not
002526  032737  0000006 0000006          BIT     #PO$SPF,PRIVCO  ;Is user authorized to do .SPFUN?
002534  001007                  BNE     7$              ;Br if yes
002536  120237  0000006          CMPB    R2,LDDEVX       ;Is this a .SPFUN to device LD?
002542  001404                  BEQ     7$              ;Br if yes -- That is ok
002544  012700  177747           MOV     #-31,R0         ;Error -31 -- Not authorized for .SPFUN
002550  000137  0000006          JMP     SETERR
                                ;
                                ;  Get a free I/O queue element and set it up.
                                ;
002554  004737  0000006          7$:     CALL    GETQ            ;GET A FREE I/O QUEUE ELEMENT
                                ;
                                ;  R1 now points to I/O queue element.
                                ;  Set special function code in queue element.
                                ;
002560  113761  0000116 0000006          MOVB    EMTBLK+11,Q.FUNC(R1);STORE SPECIAL-FUNCTION CODE VALUE
                                ;
                                ;  Set up buffer address and other info in queue element
                                ;
002566  004737  011772'                  CALL    SETQ            ;SET UP BUFFER ADDRESS AND WORD COUNT
002572  103443                  BCS     9$              ;BR IF SOME ERROR DETECTED
                                ;
                                ;  Set up address of completion routine.
                                ;
002574  013700  0000126                  MOV     EMTBLK+12,R0    ;GET ADDRESS OF COMPLETION ROUTINE
002600  004737  012472'                  CALL    SETCR           ;SET UP COMPL ADDR IN QUEUE ELEMENT
002604  103436                  BCS     9$              ;BR IF INVALID COMPLETION QUEUE ADDRESS
```

```
                                        ;
                                        ;  Initiate the I/O operation.
                                        ;
002606  004737  000000G                       CALL    QIO             ;QUEUE THE I/O REQUEST
                                        ;
                                        ;  See if we should wait for the I/O operation to finish.
                                        ;
002612  005737  0000120                       TST     EMTBLK+12       ;IS WAIT REQUESTED?
002616  001002                                BNE     13$             ;BR IF NOT
                                        ;
                                        ;  Wait for I/O to finish
                                        ;
002620  004737  000000G                       CALL    IOWAIT          ;WAIT FOR I/O TO FINISH
                                        ;
                                        ;  See if a hardware error or end-of-file occured.
                                        ;
002624  032763  000000G 000000G 13$:    BIT     #CS$ERR,C.CSW(R3);DID A HARDWARE ERROR OCCUR?
002632  001407                          BEQ     6$              ;BR IF NOT
002634  042763  000000G 000000G         BIC     #CS$ERR,C.CSW(R3);CLEAR ERROR FLAG IN CSW
002642  012700  000001                  MOV     #1,R0           ;REPORT ERROR
002646  000137  000000G                 JMP     SETERR
002652  032763  000000G 000000G 6$:     BIT     #CS$EOF,C.CSW(R3);DID WE HIT END-OF-FILE?
002660  001406                          BEQ     5$              ;BR IF NOT
002662  042763  000000G 000000G         BIC     #CS$EOF,C.CSW(R3);CLEAR END-OF-FILE FLAG IN CSW
002670  005000                          CLR     R0              ;RETURN ERROR CODE OF 0
002672  000137  000000G                 JMP     SETERR
                                        ;
                                        ;  Normal completion of .SPFUN
                                        ;
002676  000137  000000G         5$:     JMP     EMTXIT          ;FINISHED EMT
                                        ;
                                        ;  Some fatal error occured.  Abort request.
                                        ;
002702  000137  002364'         9$:     JMP     IOABRT          ;ABORT I/O REQUEST
```

```
                                        .SBTTL  .SCCA
                                ;-------------------------------------------------
                                ;  PROCESS THE .SCCA EMT
                                ;
002706  016137  000000G 000000G SCCA:   MOV     LSCCA(R1),URO   ;RETURN OLD SCCA RTN ADDRESS TO USER IN RO
002714  013700  000020G         MOV     EMTBLK+2,R0     ;GET ADDRESS OF SCCA ROUTINE
002720  004737  000000G         CALL    VALADW          ;VALIDATE THE ADDRESS
002724  010061  000000G         MOV     R0,LSCCA(R1)    ;SAVE ADDRESS OF SCCA ROUTINE FOR JOB
002730  000406                  BR      JNOP

                                        .SBTTL  .TRPSET
                                ;-------------------------------------------------
                                ;  PROCESS THE .TRPSET EMT
                                ;
002732  013700  000020G         TRPSET: MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S TRAP ROUTINE
002736  004737  000000G         CALL    VALADW          ;VALIDATE THE ADDRESS
002742  010037  000000G         MOV     R0,UTRPAD       ;SET TRAP ROUTINE ADDRESS FOR JOB
002746  000137  000000G         JNOP:   JMP     EMTXIT

                                        .SBTTL  .SFPA
                                ;-------------------------------------------------
                                ;  PROCESS THE .SFPA EMT
                                ;
002752  013700  000020G         SFPA:   MOV     EMTBLK+2,R0     ;GET ADDRESS OF TRAP ROUTINE
002756  004737  000000G         CALL    VALADB          ;VALIDATE IT (NOTE: ADDRESS MAY = 1)
002762  010037  000000G         MOV     R0,UFPTRP       ;SET ADDRESS OF TRAP ROUTINE
002766  005700                  TST     R0              ;IS USER GOING TO USE FPU UNIT?
002770  001407                  BEQ     2$              ;BR IF NOT
002772  032737  000000G 000000G BIT     #CW$FPU,CONFIG  ;DOES SYSTEM HAVE A FPU UNIT?
003000  001403                  BEQ     2$              ;BR IF NOT
003002  110137  000000G         MOVB    R1,FPUUSE       ;REMEMBER USER IS ACCESSING FPU
003006  000402                  BR      3$
003010  105037  000000G 2$:     CLRB    FPUUSE          ;SAY FPU NOT IN USE
003014  000754          3$:     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.
                                ;
003016  013737  000040G 000000G QSET:   MOV     EMTBLK+4,UR0    ;Return address of Q buffer in R0
003024  000137  000000G         JMP     EMTXIT          ;Finished with EMT

                                        .SBTTL  .SETTOP
                                ;-------------------------------------------------------------------
                                ;  PROCESS THE .SETTOP EMT
                                ;
003030  013700  000000G         SETTOP: MOV     UR0,R0          ;GET USER'S REQUESTED TOP ADDRESS
003034  005200                  SETTP1: INC     R0              ;BOUND ADDRESS UP TO WORD IF IT'S ODD
003036  001001                  BNE     2$              ;BR IF DID NOT OVERFLOW
003040  005300                  DEC     R0              ;IF OVERFLOWED, FORCE BACK TO 177777
003042  042700  000001  2$:     BIC     #1,R0           ;FORCE ADDRESS TO BE EVEN
003046  020037  000000G         CMP     R0,ODTBAS       ;DOES REQUEST EXCEED JOB'S LIMIT?
003052  103404                  BLO     1$              ;BR IF NOT
003054  013700  000000G         MOV     ODTBAS,R0       ;GET MAX AMT JOB IS ALLOWED TO USE
003060  162700  000002          SUB     #2,R0           ;POINT TO WORD BELOW TOP LIMIT
```

.SETTOP

```
                                ;RETURN TOP ADDRESS IN USER'S R0
                                ;AND IN LOCATION 50
                                ;FINISHED
                        ;
                        ;-------------------------------------------------------------------------
                        ;  The TSX-Plus SETSIZ emt is used to change the memory space allocated
                        ;  to a job.
                        ;
                        ;  The form of the EMT argument block is:
                        ;
                        ;       .BYTE   0,141
                        ;       .WORD   .top-addr       ;Requested top address for job
                        ;
SETSIZ: MOV     EMTBLK+2,R0     ;GET REQUESTED TOP OF JOB ADDRESS
        TSTB    VSWPFL          ;IS THIS A NON-SWAPPING SYSTEM?
        BEQ     SETTP1          ;IF NON-SWAPPING THEN CAN'T CHANGE SIZE
        CMPB    R1,EXCJOB       ;DOES JOB HAVE EXCLUSIVE ACCESS TO SYSTEM?
        BEQ     SETTP1          ;IF YES THEN CAN'T CHANGE SIZE
        MOV     MXJADR,R2       ;GET ADDRESS ABOVE TOP OF AVAILABLE SPACE
        SUB     #2,R2           ;GET ADDRESS OF TOP AVAILABLE WORD
        CMP     R0,R2           ;IS REQUEST ABOVE TOP ALLOWED?
        BLOS    1$              ;BR IF NOT
        MOV     R2,R0           ;CONSTRAIN TO TOP ALLOWED ADDRESS
1$:     MOV     R0,URO          ;RETURN TOP ADDRESS TO USER IN R0
        MOV     R0,-(SP)        ;AND PUT IN USER'S LOCATION 50
        MTPD    @#50
        CMP     R0,#160000      ;IS JOB NOW USING MORE THAN 56KB?
        BLO     2$              ;BR IF NOT
        BIS     #VIMAGE,LJSW(R1);SET VIRTUAL-JOB FLAG
        MOV     LJSW(R1),-(SP)
        MTPD    @#JSWLOC
        BIS     #$VIRJB,LSW9(R1);ALSO SET IN INTERNAL SYSTEM TABLE
2$:     ADD     #2,R0           ;GET ADDRESS ABOVE TOP WORD WANTED
        MOV     R0,ODTBAS       ;CONSTRAIN .SETTOP TO THIS
        CALL    SUTOP           ;DO MEMORY ALLOCATION FOR JOB
        JMP     EMTXIT          ;FINISHED
```

```
                                        .SBTTL  .GTJB
                                ;-------------------------------------------------------
                                ;  PROCESS THE .GTJB EMT
                                ;
003224  013700  000002G         GTJB:   MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S BUFFER
003230  004737  000000G                 CALL    VALADW          ;VALIDATE BUFFER ADDRESS
003234  010146                         MOV     R1,-(SP)        ;RETURN JOB #
003236  006216                         ASR     (SP)
003240  106620                         MTPD    (R0)+           ;[WORD 1]
003242  013746  000000G                 MOV     ODTBAS,-(SP)    ;HIGH MEMORY LIMIT
003246  106620                         MTPD    (R0)+           ;[WORD 2]
003250  005046                         CLR     -(SP)           ;LOW MEMORY
003252  106620                         MTPD    (R0)+           ;[WORD 3]
003254  012746  000000C                 MOV     #RMNBAS+R$CHN-2,-(SP);START OF JOB CHANNEL AREA
003260  106620                         MTPD    (R0)+           ;[WORD 4]
003262  005046                         CLR     -(SP)           ;ADDRESS OF JOB'S IMPURE AREA
003264  106620                         MTPD    (R0)+           ;[WORD 5]
003266  010146                         MOV     R1,-(SP)        ;JOB'S TERMINAL NUMBER
003270  006216                         ASR     (SP)
003272  106620                         MTPD    (R0)+           ;[WORD 6]
003274  013746  000000G                 MOV     ODTBAS,-(SP)    ;VIRTUAL HIGH LIMIT
003300  106620                         MTPD    (R0)+           ;[WORD 7]
003302  000137  000000G         GTJJ:   JMP     EMTXIT
```

```
                                        .SBTTL  .CHAIN
                                ;-------------------------------------------------------
                                ;  PROCESS A .CHAIN REQUEST.
                                ;
003306  105237  000000G          DOCIN:  INCB    CINFLG          ;SAY .CHAIN IN PROGRESS
                                ;
                                ;  Move chain data to job context area.
                                ;
003312  012702  000500                  MOV     #500,R2         ;POINT TO START OF CHAIN DATA AREA
003316  012703  000000G                 MOV     #CINDAT,R3      ;POINT TO CHAIN DATA SAVE AREA IN CONTEXT BLK
003322  012704  000140                  MOV     #<<1000-500>/2>,R4 ;GET # WORDS TO MOVE
003326  106522                  1$:     MFPD    (R2)+           ;GET A WORD FROM USER'S CHAIN DATA AREA
003330  012623                          MOV     (SP)+,(R3)+     ;MOVE TO CONTEXT BLOCK AREA
003332  077403                          SOB     R4,1$
003334  004737  000000G                 CALL    STOP            ;EXIT TO KMON TO DO .CHAIN

                                        .SBTTL  .RCTRLO
                                ;-------------------------------------------------------
                                ;  PROCESS THE .RCTRLO EMT
                                ;
003340  032761  000000G 000000G RCTRLO: BIT     #$CFOPN,LSW4(R1);Is input coming from a command file?
003346  001003                          BNE     1$              ;Br if yes -- Don't reset ctrl-O
003350  042761  000000G 000000G         BIC     #$CTRLO,LSW3(R1);Reset Ctrl-O flag
003356  000137  000000G          1$:     JMP     EMTXIT
```

```
                                        .SBTTL  .HRESET
                                ;-------------------------------------------------------------
                                ;  Process the .HRESET EMT
                                ;  (Halt all I/O for job and then purge all I/O channels)
                                ;
003362  004737  000000G         HRESET: CALL    IOHALT          ;HALT ALL I/O FOR JOB
003366  000400                          BR      SRESET          ;NOW DO .SRESET

                                        .SBTTL  .SRESET
                                ;-------------------------------------------------------------
                                ;  PROCESS THE .SRESET AND .HRESET EMTS.
                                ;
003370                          SRESET:
                                ;
                                ;  Reset .SPND/.RSUM counter for job
                                ;
003370  005061  000000G                 CLR     LSPND(R1)       ;RESET .SPND COUNT FOR JOB
                                ;
                                ;  Cancel any pending .MRKT requests
                                ;
003374  004737  004724'                 CALL    CANMKT          ;CANCEL PENDING MARK-TIME REQUESTS
                                ;
                                ;  Cancel any job monitoring requests
                                ;
003400                                  OCALL   MONABT          ;Cancel any job monitoring
                                ;
                                ;  Cancel any message-read completion routine requests
                                ;
003406  005737  000000G                 TST     VMAXMC          ;Is message support genned in?
003412  001403                          BEQ     4$              ;Br if not
003414                                  OCALL   MSGABT          ;Abort any message-read cpl routines
                                ;
                                ;  Purge I/O channels
                                ;
003422  012702  177777G         4$:     MOV     #NUCHN-1,R2     ;GET HIGHEST USER CHANNEL #
003426  020227  000017          1$:     CMP     R2,#17          ;IS THIS CHANNEL 17?
003432  001004                          BNE     2$              ;BR IF NOT
003434  032761  000000G 000000G         BIT     #OVLBIT,LJSW(R1);IS THE PROGRAM OVERLAYED?
003442  001004                          BNE     3$              ;IF IT IS, DON'T CLOSE CHANNEL 17
003444                          2$:     .PURGE  R2              ;PURGE EACH ONE
003454  005302                  3$:     DEC     R2
003456  002363                          BGE     1$
                                ;
                                ;  Undo .CDFN
                                ;
003460  005037  000000G                 CLR     UCHAN           ;RESET USER CHANNEL DEFINITIONS
003464  000137  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  .SAVESTATUS
                                ;-------------------------------------------------------------
                                ;  Save channel status.
                                ;
003470  013703  000000G        SVSTAT: MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
003474  032763  000000G 000000G        BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
003502  001417                        BEQ     4$              ;BR IF NOT OPEN NOW
                                ;  Make sure file was opened with a lookup.
003504  032763  000000G 000000G        BIT     #CS$ENT,C.CSW(R3);OPENED WITH LOOKUP OR ENTER?
003512  001404                        BEQ     2$              ;BR IF LOOKUP
003514  012700  000001                MOV     #1,R0           ;RETURN ERROR CODE 1
003520  000137  000000G                JMP     SETERR
                                ;
                                ;  Wait for any I/O activity on the channel to finish
                                ;
003524  004737  000000G        2$:     CALL    IOWAIT          ;WAIT FOR I/O TO FINISH
                                ;
                                ;  Tell TSX shared file logic that savestatus is being done
                                ;
003530  005737  000000G                TST     JCDB            ;Any shared file channels open for job?
003534  001402                        BEQ     4$              ;Br if not
003536  004777  000000G                CALL    @SFSVST         ;TELL SHARED FILE SYSTEM
                                ;
                                ;  Move channel block to user's area
                                ;
003542  010301                 4$:     MOV     R3,R1           ;SAVE ADDRESS OF CHANNEL AREA
003544  013700  000002G                MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S SAVE AREA
003550  004737  000000G                CALL    VALADW          ;VALIDATE THE ADDRESS
003554  012704  000005                MOV     #5,R4           ;GET # WORDS TO MOVE
003560  012346                 3$:     MOV     (R3)+,-(SP)     ;MOVE DATA TO USER'S AREA
003562  106620                        MTPD    (R0)+
003564  077403                        SOB     R4,3$
                                ;
                                ;  Mark channel as closed
                                ;
003566  032761  0000000 000000G        BIT     #CS$OPN,C.CSW(R1);IS THE CHANNEL OPEN?
003574  001404                        BEQ     5$              ;BR IF CHANNEL IS NOT OPEN
003576  005061  000000G                CLR     C.CSW(R1)       ;PURGE THE CHANNEL
003602  000137  000000G                JMP     EMTXIT          ;FINISHED
003606  005000                 5$:     CLR     R0              ;RETURN ERROR CODE OF 0 IF CHANNEL CLOSED
003610  000137  000000G                JMP     SETERR
```

```
                                        .SBTTL  .REOPEN
                                ;-------------------------------------------------------------------------------
                                ;  Reopen a channel using saved status information.
                                ;
003614  013703  0000006         REOPEN: MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
                                ;
                                ;  Make sure channel is not now open.
                                ;
003620  032763  0000006 0000006         BIT     #CS$OPN,C.CSW(R3);IS CHANNEL OPEN NOW?
003626  001403                          BEQ     1$              ;BR IF NOT
003630  005000                          CLR     R0              ;RETURN ERROR CODE OF 0
003632  000137  0000006               JMP     SETERR
                                ;
                                ;  Channel is closed.  Restore status.
                                ;
003636  013700  0000026         1$:     MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S SAVE STATUS BLOCK
003642  004737  0000006               CALL    VALADW          ;VALIDATE THE ADDRESS
003646  012704  000005                MOV     #5,R4           ;GET # WORDS TO MOVE
003652  106520                  3$:     MFPD    (R0)+           ;GET A WORD FROM USER'S AREA
003654  012623                          MOV     (SP)+,(R3)+     ;MOVE INTO CHANNEL BLOCK
003656  077403                          SOB     R4,3$           ;MOVE ALL
                                ;
                                ;  Check consistency of savestatus information that we restored.
                                ;
003660  013703  0000006               MOV     CHNADR,R3       ;POINT TO CHANNEL BLOCK
003664  016300  0000006               MOV     C.CSW(R3),R0    ;GET CHANNEL STATUS WORD (CSW)
003670  032700  0000006               BIT     #CS$OPN,R0      ;CHANNEL SHOULD BE OPEN NOW
003674  001425                          BEQ     9$              ;ERROR IF NOT
003676  032700  0000006               BIT     #CS$ENT,R0      ;CHANNEL SHOULD HAVE BEEN OPENED VIA LOOKUP
003702  001022                          BNE     9$              ;ERROR IF NOT
003704  105063  0000006               CLRB    C.NUMQ(R3)      ;THERE SHOULD BE NO ACTIVE I/O OPERATIONS
003710  126327  0000006 000007        CMPB    C.DEVQ(R3),#7   ;CHECK VALIDITY OF UNIT NUMBER
003716  101014                          BHI     9$              ;ERROR IF TOO BIG
003720  042700  000000C               BIC     #^C<CS$NMX>,R0  ;GET DEVICE INDEX NUMBER
003724  020037  0000006               CMP     R0,NUMDEV       ;MAKE SURE IT IS A VALID NUMBER
003730  101007                          BHI     9$              ;BR IF TOO BIG
                                ;
                                ;  Tell shared file system about the reopen.
                                ;
003732  005737  0000006               TST     JCDB            ;Does job have any shared file channels?
003736  001402                          BEQ     4$              ;Br if not
003740  004777  0000006               CALL    @SFRSST         ;WE MAY BE REOPENING A SHARED FILE
                                ;
                                ;  Finished
                                ;
003744  000137  0000006         4$:     JMP     EMTXIT
                                ;
                                ;  Error -- Savestatus info we restored was invalid.
                                ;           Return error code of 1.
                                ;
003750  005063  0000006         9$:     CLR     C.CSW(R3)       ;PURGE THE CHANNEL
003754  012700  000001                MOV     #1,R0           ;RETURN ERROR CODE 1
003760  000137  0000006               JMP     SETERR
```

```
                                        .SBTTL  .PURGE
                                ;-------------------------------------------------------------------------------
                                ;  Purge a channel.
                                ;
003764  013703  0000000G        PURGE:  MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
003770  032763  0000000G 0000000G       BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN NOW?
003776  001434                        BEQ     1$              ;BR IF NOT OPEN
004000  004737  0000000G              CALL    IOWAIT          ;WAIT FOR I/O ON CHANNEL TO FINISH
004004  113701  0000000G              MOVB    CHNNUM,R1       ;GET CHANNEL NUMBER
004010  105737  0000000G              TSTB    NSPLDV          ;Are there any spooled devices?
004014  001403                        BEQ     2$              ;BR IF NOT
004016                                OCALL   SDCLOS          ;SEE IF THIS IS A SPOOLED DEVICE
004024  005737  0000000G        2$:     TST     JCDB            ;Does job have any shared file channels?
004030  001402                        BEQ     4$              ;Br if not
004032  004777  0000000G              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.
                                ;
004036  016300  0000000G        4$:     MOV     C.CSW(R3),R0    ;GET CHANNEL STATUS WORD
004042  042700  000000C              BIC     #^C<CS$NMX>,R0  ;MASK OUT ALL BUT DEVICE INDEX NUMBER
004046  032760  0000000G 0000000G       BIT     #DS$NRD,DVSTAT(R0);IS DEVICE A MAGNETIC TAPE?
004054  001405                        BEQ     1$              ;BR IF NOT
004056  005737  0000000G              TST     IOABFL          ;IS IO ABORT IN EFFECT?
004062  001002                        BNE     1$              ;BR IF YES
004064  000137  0000000G              JMP     CLOSE           ;TREAT PURGE ON MAG TAPE LIKE CLOSE
004070  005063  0000000G        1$:     CLR     C.CSW(R3)       ;MARK CHANNEL AS CLOSED
004074  000137  0000000G              JMP     EMTXIT          ;FINISHED
```

```
                                        .SBTTL  .WAIT
                                ;-----------------------------------------------------------------------
                                ;  .WAIT -- Wait for all I/O to finish on a channel.
                                ;
004100  013703  000000G         DOWAIT: MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL
                                ;
                                ;  Make sure channel is open
                                ;
004104  032763  000000G 000000G         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
004112  001003                          BNE     1$              ;BR IF YES
004114  005000                          CLR     R0              ;RETURN ERROR CODE OF 0 IF NOT OPEN
004116  000137  000000G                 JMP     SETERR
                                ;
                                ;  Wait for all I/O to finish
                                ;
004122  004737  000000G         1$:     CALL    IOWAIT          ;WAIT FOR I/O TO FINISH
                                ;
                                ;  Check for hardware errors
                                ;
004126  032763  000000G 000000G         BIT     #CS$ERR,C.CSW(R3);DID A HARD ERROR OCCUR?
004134  001407                          BEQ     9$              ;BR IF NOT
004136  042763  000000G 000000G         BIC     #CS$ERR,C.CSW(R3);CLEAR THE ERROR FLAG IN THE CSW
004144  012700  000001                  MOV     #1,R0           ;Return error code 1
004150  000137  000000G                 JMP     SETERR
                                ;
                                ;  Finished
                                ;
004154  000137  000000G         9$:     JMP     EMTXIT          ;RETURN
```

```
                                        .SBTTL  .GTIM
                                ;-------------------------------------------------------------------------
                                ;  Get time of day.
                                ;
004160  013700  000002G        GETTIM: MOV     EMTBLK+2,R0     ;GET ADDR OF USER'S TIME BUFFER
004164  004737  000000G                CALL    VALADW          ;VALIDATE THE ADDRESS
004170  013746  000000G                MOV     SYTIMH,-(SP)    ;RETURN HIGH-ORDER TIME
004174  106620                        MTPD    (R0)+
004176  013746  000000G                MOV     SYTIML,-(SP)    ;RETURN LOW-ORDER TIME
004202  106610                        MTPD    (R0)
004204  000403                        BR      UPDATE          ;GO UPDATE DATE CELL

                                        .SBTTL  .DATE
                                ;-------------------------------------------------------------------------
                                ;  Get current date (return to user in R0).
                                ;
004206  013737  000000G 000000G DATE:   MOV     SYSDAT,@R0      ;RETURN DATE TO USER IN R0
                                ;  Move current date value to cell in user's simulated monitor vector table.
004214  013702  000000G        UPDATE: MOV     CXTRMN,R2       ;GET BASE ADDRESS OF JOB VECTOR
004220  013762  000000G 000000G        MOV     SYSDAT,R$DATE(R2);SET DATE VALUE
004226  000137  000000G                JMP     EMTXIT

                                        .SBTTL  .SDTTM
                                ;-------------------------------------------------------------------------
                                ;  .SDTTM -- Set system date and time.
                                ;
004232  032737  000000G 000000G SDTTM:  BIT     #P0$OPR,PRIVCO  ;IS THIS A PRIVILEGED USER RUNNING?
004240  001003                        BNE     1$              ;BR IF YES
004242  005000                        CLR     R0              ;RETURN ERROR CODE 0
004244  000137  000000G                JMP     SETERR
                                ;
                                ;  Set date
                                ;
004250  013702  000002G        1$:     MOV     EMTBLK+2,R2     ;GET ADDRESS OF DATE/TIME VALUE BLOCK
004254  010200                        MOV     R2,R0           ;VALIDATE THE ADDRESS
004256  004737  000000G                CALL    VALADW
004262  106522                        MFPD    (R2)+           ;GET NEW DATE VALUE
004264  012600                        MOV     (SP)+,R0
004266  002402                        BLT     2$              ;IF NEGATIVE THEN DON'T CHANGE DATE
004270  010037  000000G                MOV     R0,SYSDAT       ;SET NEW SYSTEM DATE
                                ;
                                ;  Set time
                                ;
004274  106522                2$:     MFPD    (R2)+           ;GET HIGH-ORDER TIME VALUE
004276  012600                        MOV     (SP)+,R0
004300  002405                        BLT     3$              ;IF NEGATIVE, DON'T CHANGE TIME VALUE
004302  010037  000000G                MOV     R0,SYTIMH       ;SET NEW HIGH-ORDER TIME VALUE
004306  106522                        MFPD    (R2)+           ;GET NEW LOW-ORDER TIME VALUE
004310  012637  000000G                MOV     (SP)+,SYTIML    ;SET LOW-ORDER TIME VALUE
                                ;
                                ;  Finished
                                ;
004314  000137  000000G        3$:     JMP     EMTXIT
```

```
                                        .SBTTL  .TWAIT
                                ;-----------------------------------------------------------------------
                                ;  Do a timed wait
                                ;
004320  013700  000020          TIMWAT: MOV     EMTBLK+2,R0     ;GET POINTER TO TIME VALUE BLOCK
004324  004737  000000G                 CALL    VALADW          ;VALIDATE THE ADDRESS
004330  106520                          MFPD    (R0)+           ;GET HIGH-ORDER TIME VALUE
004332  012661  000000G                 MOV     (SP)+,LSLEPH(R1)
004336  106510                          MFPD    (R0)            ;GET LOW-ORDER TIME VALUE
004340  012661  000000G                 MOV     (SP)+,LSLEPL(R1)
                                ;
                                ;  Suspend job for timed interval
                                ;
004344  004737  000000G         1$:     CALL    CHKABT          ;MAKE SURE WE HAVEN'T BEEN ABORTED
004350  005761  000000G                 TST     LSLEPH(R1)      ;IS THERE STILL SOME SLEEP TIME LEFT?
004354  002411                          BLT     3$              ;BR IF NOT (TIME HAS GONE NEGATIVE)
004356  003003                          BGT     2$              ;BR IF SOME TIME LEFT
004360  005761  000000G                 TST     LSLEPL(R1)      ;CHECK LOW-ORDER VALUE
004364  001405                          BEQ     3$              ;BR IF TIME IS UP
004366  012700  000000G         2$:     MOV     #S$TMWT,R0      ;STATE BECOMES TIMED WAIT
004372  004737  000000G                 CALL    QHDSPN          ;SUSPEND THE JOB
004376  000762                          BR      1$
                                ;
                                ;  Timed wait is finished
                                ;
004400  000137  000000G         3$:     JMP     EMTXIT
```

```
                                        .SBTTL  .MRKT
                                ;-------------------------------------------------------------------------
                                ;  Start a mark-time request.
                                ;
004404                          MRKT:
                                ;
                                ;  Get an I/O queue entry.
                                ;
004404  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)
                                ;
004410  113761  000000G 000000G        MOVB    CORUSR,CQ$JOB(R1);SET JOB # IN QUEUE ELEMENT
                                ;  Set time value.
004416  013700  000020G                 MOV     EMTBLK+2,R0             ;GET ADDRESS OF TIME VALUE ARG BLOCK
004422  004737  000000G                 CALL    UACHKW                  ;MAKE SURE ADDRESS IS LEGAL
004426  103440                         BCS     9$                      ;BR IF INVALID ADDRESS
004430  106520                         MFPD    (R0)+                   ;GET HIGH-ORDER TIME VALUE
004432  012661  000000G                 MOV     (SP)+,CQ$HOT(R1);SET HIGH-ORDER TIME VALUE
004436  106510                         MFPD    (R0)                    ;GET LOW-ORDER TIME VALUE
004440  012661  000000G                 MOV     (SP)+,CQ$LOT(R1)
                                ;  Set address of completion routine
004444  013700  000004G                 MOV     EMTBLK+4,R0             ;GET ADDRESS OF COMPLETION ROUTINE
004450  004737  000000G                 CALL    UACHKW                  ;MAKE SURE ADDRESS IS LEGAL
004454  103425                         BCS     9$                      ;BR IF NOT LEGAL
004456  010061  000000G                 MOV     R0,CQ$RTN(R1)   ;SET ADDRESS OF COMPLETION ROUTINE
                                ;  Set mapping for kernel PAR 5 that was in effect when EMT was executed.
004462  013761  000000G 000000G        MOV     EMTMAP,CQ$PA5(R1);SET EMT ENTRY MAPPING FOR PAR 5
                                ;  Set ID number
004470  013761  000006G 000000G        MOV     EMTBLK+6,CQ$RO(R1);SET COMPLETION ID NUMBER
                                ;
                                ;  Add queue entry to mark-time list.
                                ;
004476                                  DISABL                          ;** DISABLE **
004504  013761  000000G 000000G        MOV     MRKTHD,CQ$LNK(R1);ADD TO FRONT OF LIST
004512  010137  000000G                 MOV     R1,MRKTHD
004516                                  ENABL                           ;** ENABLE **
                                ;
                                ;  Finished
                                ;
004524  000137  000000G                 JMP     EMTXIT
                                ;
                                ;  Error -- Invalid address in MRKT argument list.
                                ;
004530  004737  000000G         9$:     CALL    QFREE                   ;RELEASE THE QUEUE ELEMENT
004534  012700  177766                 MOV     #-12,R0                 ;RETURN ERROR CODE
004540  000137  000000G                 JMP     SETERR                  ;ABORT EMT
```

```
                                        .SBTTL  .CMKT
                                ;-------------------------------------------------------------------------------
                                ;  Cancel mark-time request.
                                ;
004544  010103                  CMKT:   MOV     R1,R3           ;COPY JOB INDEX #
004546  013702  0000020                 MOV     EMTBLK+2,R2     ;GET ID VALUE
004552  001460                          BEQ     1$              ;BR IF WE SHOULD CANCEL ALL REQUESTS FOR JOB
                                ;
                                ;  Cancel a specifiec mark-time request for this job.
                                ;
004554  012704  000000C                 MOV     #MRKTHD-CQ$LNK,R4;FAKE UP POINTER TO LIST HEAD
004560                                  DISABL                  ;** DISABLE **
004566  016401  0000000         4$:     MOV     CQ$LNK(R4),R1   ;GET ADDRESS OF NEXT MARK-TIME QUEUE ENTRY
004572  001442                          BEQ     5$              ;BR IF REACHED END OF LIST
004574  120361  0000000                 CMPB    R3,CQ$JOB(R1)   ;DOES THIS ENTRY BELONG TO OUR JOB?
004600  001007                          BNE     2$              ;BR IF NOT
004602  132761  0000000 0000000         BITB    #QF$IOT,CQ$FLG(R1);IS THIS A .TIMIO ENTRY?
004610  001003                          BNE     2$              ;BR IF YES
004612  020261  0000000                 CMP     R2,CQ$RO(R1)    ;DO ID NUMBERS MATCH?
004616  001402                          BEQ     3$              ;BR IF YES -- FOUND ENTRY TO DELETE
004620  010104                  2$:     MOV     R1,R4           ;CHAIN FORWARD TO NEXT ENTRY IN LIST
004622  000761                          BR      4$
                                ;
                                ;  Found specified queue entry -- delete from list.
                                ;
004624  016164  0000000 0000000 3$:     MOV     CQ$LNK(R1),CQ$LNK(R4);RELINK LIST AROUND US
004632                                  ENABL                   ;** ENABLE **
                                ;
                                ;  See if we need to return remaining time.
                                ;
004640  013700  0000040                 MOV     EMTBLK+4,R0     ;DOES USER WANT REMAINING TIME?
004644  001411                          BEQ     6$              ;BR IF NOT
004646  004737  0000000                 CALL    UACHKW          ;MAKE SURE RECIEVING AREA IS LEGAL
004652  103406                          BCS     6$              ;BR IF NOT LEGAL
004654  016146  0000000                 MOV     CQ$HOT(R1),-(SP);RETURN HIGH-ORDER TIME
004660  106620                          MTPD    (R0)+
004662  016146  0000000                 MOV     CQ$LOT(R1),-(SP);RETURN LOW-ORDER TIME
004666  106610                          MTPD    (R0)
                                ;
                                ;  Return queue element to free list.
                                ;
004670  004737  0000000         6$:     CALL    QFREE           ;FREE THE Q ELEMENT
                                ;
                                ;  Finished
                                ;
004674  000137  0000000                 JMP     EMTXIT
                                ;
                                ;  Could not find specified queue element -- return error code 0.
                                ;
004700                          5$:     ENABL                   ;** ENABLE **
004706  005000                          CLR     R0              ;RETURN ERROR CODE 0
004710  000137  0000000                 JMP     SETERR
                                ;
                                ;  Cancel all mark-time requests for this job.
                                ;
004714  004737  004724'         1$:     CALL    CANMKT          ;CANCEL ALL MARK-TIME REQUESTS FOR JOB
004720  000137  0000000                 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.
                                ;
004724  010146                  CANMKT: MOV     R1,-(SP)
004726  010446                          MOV     R4,-(SP)
004730  012704  000000C          1$:     MOV     #MRKTHD-CQ$LNK,R4;DUMMY POINTER TO LIST HEAD
004734                                  DISABL                  ;** DISABLE **
004742  016401  000000G          2$:     MOV     CQ$LNK(R4),R1   ;GET NEXT ENTRY IN LIST
004746  001423                          BEQ     4$              ;BR IF REACHED END OF LIST
004750  123761  000000G 000000G          CMPB    CORUSR,CQ$JOB(R1);IS THIS ENTRY FOR THIS JOB?
004756  001015                          BNE     3$              ;BR IF NOT
004760  132761  000000G 000000G          BITB    #QF$IOT,CQ$FLG(R1);IS THIS A HANDLER .TIMIO ENTRY?
004766  001011                          BNE     3$              ;BR IF YES
004770  016164  000000G 000000G          MOV     CQ$LNK(R1),CQ$LNK(R4);RELINK LIST AROUND US
004776                                  ENABL                   ;** ENABLE **
005004  004737  000000G                  CALL    QFREE           ;RETURN QUEUE ELEMENT TO FREE LIST
005010  000747                          BR      1$
005012  010104                  3$:     MOV     R1,R4           ;LINK TO NEXT ELEMENT IN LIST
005014  000752                          BR      2$
                                ;
                                ;  Finished
                                ;
005016                          4$:     ENABL                   ;** ENABLE **
005024  012604                          MOV     (SP)+,R4
005026  012601                          MOV     (SP)+,R1
005030  000207                          RETURN
```

```
                                        .SBTTL  .CSTAT
                                ;-------------------------------------------------------------------------
                                ;  Return channel status
                                ;
005032  013703  000000G        CHSTAT: MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
                                ;
                                ;  Make sure channel is open.
                                ;
005036  032763  000000G 000000G        BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
005044  001003                        BNE     1$              ;BR IF YES
005046  005000                        CLR     R0              ;RETURN ERROR CODE OF 0
005050  000137  000000G               JMP     SETERR
                                ;
                                ;  Channel is open.  Return status.
                                ;
005054  013704  000002G        1$:     MOV     EMTBLK+2,R4     ;GET ADDRESS OF USER'S STATUS BLOCK
005060  010400                        MOV     R4,R0           ;VALIDATE ADDRESS
005062  004737  000000G               CALL    VALADW
                                ;  Point R0 to user's channel status block passed in EMT argument.
005066  010437  000000G               MOV     R4,UR0          ;Point R0 to channel status block
                                ;  Return CSW
005072  016346  000000G               MOV     C.CSW(R3),-(SP)
005076  106624                        MTPD    (R4)+
                                ;  Return starting block number of file
005100  016346  000000G               MOV     C.SBLK(R3),-(SP)
005104  106624                        MTPD    (R4)+
                                ;  Return allocated length of file.
005106  016346  000000G               MOV     C.LEN@(R3),-(SP)
005112  106624                        MTPD    (R4)+
                                ;  Return highest block number written so far
005114  016346  000000G               MOV     C.USED(R3),-(SP)
005120  106624                        MTPD    (R4)+
                                ;  Return device unit number.
005122  005046                        CLR     -(SP)           ;ENSURE HIGH BYTE IS CLEAR
005124  116316  000000G               MOVB    C.DEV@(R3),(SP) ;MOVE UNIT NUMBER INTO LOW BYTE
005130  106624                        MTPD    (R4)+
                                ;  Return Rad50 device name.
005132  016301  000000G               MOV     C.CSW(R3),R1    ;GET DEVICE TABLE #
005136  042701  177701                BIC     #^C76,R1        ;GET DEV # ONLY
005142  016146  000000G               MOV     PNAME(R1),-(SP) ;GET RAD50 DEVICE NAME
005146  106624                        MTPD    (R4)+
                                ;
                                ;  Finished
                                ;
005150  000137  000000G               JMP     EMTXIT


                                        .SBTTL  .CDFN
                                ;-------------------------------------------------------------------------
                                ;  Define channels.
                                ;
005154  013737  000002G 000000G CDFN:   MOV     EMTBLK+2,UCHAN  ;SAVE ADDRESS OF USER CHANNEL SPACE
005162  000137  000000G               JMP     EMTXIT
```

```
                                        .SBTTL  .GVAL
                                ;-----------------------------------------------------------------------
                                ;  .GVAL, .PVAL, .PEEK, .POKE
                                ;
005166  113700  000000G        GVAL:   MOVB    EMTBLK,R0       ;GET SUB-FUNCTION CODE (.GVAL/.PVAL etc.)
005172  120027  000003                CMPB    R0,#3           ;MAKE SURE IT IS REASONABLE
005176  101404                        BLOS    1$              ;BR IF OK
005200  012700  177767                MOV     #-11,R0         ;INVALID FUNCTION CODE
005204  000137  000000G               JMP     SETERR
005210  006300                1$:     ASL     R0              ;GET 2 * FUNCTION CODE
005212  013702  000002G               MOV     EMTBLK+2,R2     ;GET OFFSET VALUE
005216  013703  000004G               MOV     EMTBLK+4,R3     ;GET VALUE TO STORE (.PVAL & .POKE)
005222  000170  006024'               JMP     @GVJMP(R0)      ;JUMP TO PROCESSING ROUTINE
                                ;
                                ;  .GVAL  --  Get value from RMON table
                                ;
005226  032702  000001        DOGVAL: BIT     #1,R2           ;OFFSET CANNOT BE ODD
005232  001005                        BNE     2$              ;BR IF ODD
005234  005702                        TST     R2              ;IS OFFSET VALUE NEGATIVE?
005236  002415                        BLT     TSXGVL          ;IF YES THEN HE'S GETTING A TSX-PLUS VALUE
005240  020227  000000G               CMP     R2,#MAXGVL      ;IS IT TOO BIG?
005244  103403                        BLO     1$              ;BR IF OK
005246  005000                2$:     CLR     R0              ;RETURN ERROR CODE OF 0
005250  000137  000000G               JMP     SETERR
005254  063702  000000G         1$:   ADD     CXTRMN,R2       ;ADD BASE ADDR OF USER'S RMON AREA
005260  016237  000002  000000G       MOV     2(R2),URO       ;RETURN VALUE IN USER'S R0
005266  000137  000000G               JMP     EMTXIT
                                ;
                                ;  Get TSX-Plus system value
                                ;
005272  005402                TSXGVL: NEG     R2              ;GET POSITIVE OFFSET VALUE
005274  020227  000042                CMP     R2,#MXTSXV      ;IS IT IN LEGAL RANGE?
005300  101403                        BLOS    1$              ;BR IF OK
005302  005000                        CLR     R0              ;RETURN ERROR CODE 0
005304  000137  000000G               JMP     SETERR
005310  005000                1$:     CLR     R0              ;INITIALIZE VALUE TO ZERO
005312  016202  005326'               MOV     TSXGVC(R2),R2   ;GET ADDRESS OF ROUTINE TO SET VALUE
005316  004712                        CALL    @R2             ;CALL ROUTINE TO GET VALUE
005320  010037  000000G               MOV     R0,URO          ;RETURN VALUE TO USER IN R0
005324  000137  000000G               JMP     EMTXIT          ;FINISHED
                                ;
                                ;  TSX-Plus value offset vector
                                ;
        005326'                 TSXGVC  =       .-2
005330  005372'                        .WORD   TGJOBN          ; -2.  -- Get job #
005332  005400'                        .WORD   TGLDIN          ; -4.  -- Get Lead-in character
005334  005406'                        .WORD   TGPRIV          ; -6.  -- Determine if this is a privileged job
005336  005422'                        .WORD   TGIOMP          ; -8.  -- Determine if I/O page mapping in effect
005340  005436'                        .WORD   TGPROJ          ;-10.  -- Job's project number
005342  005444'                        .WORD   TGPROG          ;-12.  -- Job's programmer number
005344  005452'                        .WORD   TGLICN          ;-14.  -- TSX-Plus site license number
005346  005460'                        .WORD   TGCPRI          ;-16.  -- Current execution priority
005350  005466'                        .WORD   TGMPRI          ;-18.  -- Maximum authorized priority
005352  005510'                        .WORD   TGUCLB          ;-20.  -- Number of data blocks/job in UCL file
005354  005516'                        .WORD   TGPRIL          ;-22.  -- Primary line #
005356  005544'                        .WORD   TGSYNM          ;-24.  -- Name of physical SY device
005360  005474'                        .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
                        ;
005372  010100          TGJOBN: MOV     R1,R0           ;GET JOB INDEX NUMBER
005374  006200                  ASR     R0              ;CONVERT TO JOB NUMBER
005376  000207                  RETURN
                        ;
                        ;  Get Lead-in character
                        ;
005400  113700  000000G TGLDIN: MOVB    VTSLCH,R0       ;GET LEAD-IN CHARACTER VALUE
005404  000207                  RETURN
                        ;
                        ;  Determine if this is a privileged job
                        ;
005406  032737  000000G 000000G TGPRIV: BIT     #PO$SYS,PRIVCO  ;ARE WE PRIVILEGED?
005414  001401                  BEQ     1$              ;BR IF NOT
005416  005200                  INC     R0              ;SET PRIVILEGED FLAG
005420  000207          1$:     RETURN
                        ;
                        ;  Determine if I/O page mapping is in effect
                        ;
005422  032761  000000G 000000G TGIOMP: BIT     #$IOMAP,LSW6(R1);IS I/O PAGE MAPPING IN EFFECT?
005430  001401                  BEQ     1$              ;BR IF NOT
005432  005200                  INC     R0              ;SAY MAPPING IS IN EFFECT
005434  000207          1$:     RETURN
                        ;
                        ;  Get Project number
                        ;
005436  016100  000000G TGPROJ: MOV     LPROJ(R1),R0    ;GET PROJECT #
005442  000207                  RETURN
                        ;
                        ;  Get programmer number
                        ;
005444  016100  000000G TGPROG: MOV     LPROG(R1),R0    ;GET PROGRAMMER #
005450  000207                  RETURN
                        ;
                        ;  Get TSX-Plus site license number
                        ;
005452  013700  000000G TGLICN: MOV     TSXSIT,R0       ;GET LICENSE NUMBER
005456  000207                  RETURN
                        ;
                        ;  Get current job execution priority
                        ;
005460  116100  000000G TGCPRI: MOVB    LPRI(R1),R0     ;Get current job priority
005464  000207                  RETURN
                        ;
                        ;  Get maximum authorized job priority
                        ;
005466  113700  000000G TGMPRI: MOVB    MXJPRI,R0       ;Get maximum authorized execution priority
005472  000207                  RETURN
                        ;
                        ;  Get value of PRILOW
```

```
                                ;
005474  113700  000000G      TGPRLO: MOVB    VPRILO,R0
005500  000207                      RETURN
                                ;
                                ;  Get value of PRIHI
                                ;
005502  113700  000000G      TGPRHI: MOVB    VPRIHI,R0
005506  000207                      RETURN
                                ;
                                ;  Get number of blocks per job in TSXUCL data base file
                                ;
005510  013700  000000G      TGUCLB: MOV     UCLBLK,R0       ;Get number of blocks per job
005514  000207                      RETURN
                                ;
                                ;  Get number of controlling primary line (0 if we are a primary line)
                                ;
005516  016100  000000G      TGPRIL: MOV     LNPRIM(R1),R0   ;Get number of our primary line
005522  020001                      CMP     R0,R1           ;Are we the primary line?
005524  001001                      BNE     1$              ;Br if not
005526  005000                      CLR     R0              ;Return 0 if we are a primary line
005530  006200              1$:     ASR     R0              ;Convert from 2*job # to job #
005532  000207                      RETURN
                                ;
                                ;  Get number of parent job
                                ;
005534  016100  000000G      TGPRNT: MOV     LPARNT(R1),R0   ;Get parent job index number
005540  006200                      ASR     R0              ;Convert to job number
005542  000207                      RETURN
                                ;
                                ;  Get physical name of SY device
                                ;
005544  013700  000000G      TGSYNM: MOV     SYNAME,R0       ;Get SY physical device name
005550  000207                      RETURN
                                ;
                                ;  Get system version number
                                ;
005552  012700  000000G      TGVERS: MOV     #TSXVRS,R0      ;Get system version number
005556  000207                      RETURN
                                ;
                                ;  Get relative subprocess number
                                ;
005560  010546                TGRSUB: MOV     R5,-(SP)        ;Save R5
005562  005000                      CLR     R0              ;Primary and detached are relative 0
005564  020127  000000G             CMP     R1,#LSTDL       ;Is this a virtual line?
005570  003414                      BLE     8$              ;Branch if not
005572  016105  000000G             MOV     LNPRIM(R1),R5   ;Get owner line number
005576  016505  000000G             MOV     LSECPT(R5),R5   ;Get pointer to owner's subprocess list
005602  005200              1$:     INC     R0              ;Count subprocess number
005604  120125                      CMPB    R1,(R5)+        ;Is this our line?
005606  001405                      BEQ     8$              ;Done if line matches
005610  020027  000000G             CMP     R0,#MAXSEC      ;Any more possible?
005614  002772                      BLT     1$              ;Keep checking if so
005616  012700  177777              MOV     #-1,R0          ;Whoa! This should never happen!
005622  012605              8$:     MOV     (SP)+,R5        ;Restore R5
005624  000207                      RETURN
```

```
                                        ;
                                        ;   .PVAL  --  Store value into monitor table
                                        ;
005626  032702  000001          PVAL:  BIT     #1,R2            ;IS ADDRESS ODD?
005632  001071                         BNE     GVBAD            ;BR IF YES -- ERROR
005634  020227  000000G                CMP     R2,#MAXGVL       ;IS VALUE IN PROPER RANGE?
005640  101066                         BHI     GVBAD            ;BR IF NOT
005642  063702  000000G                ADD     CXTRMN,R2        ;ADD BASE ADDRESS OF JOB'S RMON AREA
005646  016237  000002  000000G        MOV     2(R2),UR0        ;RETURN ORIGINAL VALUE IN R0
005654  010362  000002                 MOV     R3,2(R2)         ;STORE NEW VALUE
005660  000137  000000G                JMP     EMTXIT           ;FINISHED
                                        ;
                                        ;   .PEEK -- Get value from specified memory location
                                        ;
005664  032702  000001          PEEK:  BIT     #1,R2            ;IS ADDRESS ODD?
005670  001052                         BNE     GVBAD            ;BR IF YES -- ERROR
005672  020227  000000G                CMP     R2,#RMNBAS       ;IS ADDRESS WITHIN RMON TABLE?
005676  103413                         BLO     PKLOW            ;BR IF TOO LOW
005700  020227  000000C                CMP     R2,#RMNBAS+MAXGVL ;IS IT TOO BIG?
005704  101010                         BHI     PKLOW            ;BR IF YES
005706  162702  177776G                SUB     #<RMNBAS-2>,R2   ;MAKE ADDRESS INTO OFFSET
005712  063702  000000G                ADD     CXTRMN,R2        ;ADD BASE ADDRESS OF JOB'S RMON AREA
005716  011237  000000G          PKGET: MOV     (R2),UR0         ;RETURN VALUE
005722  000137  000000G                JMP     EMTXIT
                                        ;  Peek into low memory area or I/O page
005726  032737  000000G 000000G PKLOW: BIT     #PO$MEM,PRIVCO   ;Are we authorized to access phys memory?
005734  001430                         BEQ     GVBAD            ;BR IF NOT
005736  000767                         BR      PKGET            ;GO GET THE VALUE
                                        ;
                                        ;   .POKE -- Store value into specified memory location
                                        ;
005740  032702  000001          POKE:  BIT     #1,R2            ;IS ADDRESS ODD?
005744  001024                         BNE     GVBAD            ;ERROR IF YES
005746  020227  000000G                CMP     R2,#RMNBAS       ;IS ADDRESS WITHIN RMON TABLE?
005752  103414                         BLO     POLOW            ;BR IF TOO LOW
005754  020227  000000C                CMP     R2,#RMNBAS+MAXGVL ;IS IT TOO BIG?
005760  101011                         BHI     POLOW            ;BR IF YES
005762  162702  177776G                SUB     #<RMNBAS-2>,R2   ;CONVERT TO OFFSET WITHIN TABLE
005766  063702  000000G                ADD     CXTRMN,R2        ;ADD BASE OF JOB'S RMON AREA
005772  011237  000000G          POVAL: MOV     (R2),UR0         ;GET ORIGINAL VALUE
005776  010312                         MOV     R3,(R2)          ;SET NEW VALUE
006000  000137  000000G                JMP     EMTXIT           ;FINISHED
                                        ;  Poke into low memory or I/O page
006004  032737  000000G 000000G POLOW: BIT     #PO$MEM,PRIVCO   ;Are we authorized to access phys memory?
006012  001401                         BEQ     GVBAD            ;BR IF NOT
006014  000766                         BR      POVAL            ;GO STORE VALUE
                                        ;
                                        ;  Invalid offset specified
                                        ;
006016  005000                  GVBAD: CLR     R0               ;RETURN ERROR # 0
006020  000137  000000G                JMP     SETERR
                                        ;
                                        ;  Sub-function jump vector
                                        ;
006024  005226'                 GVJMP: .WORD   DOGVAL           ;0 - .GVAL
006026  005664'                        .WORD   PEEK             ;1 - .PEEK
006030  005626'                        .WORD   PVAL             ;2 - .PVAL
```

```
     58 006032  005740'                         .WORD   POKE            ;3 - .POKE
```

.EXIT

```
                                        .SBTTL  .EXIT
                                ;-------------------------------------------------------------------------------
                                ;  Program exit
                                ;
006034  005061  000000G         EXIT:   CLR     LEMTPC(R1)          ;CLEAR ADDRESS OF LAST USER-MODE EMT
                                ;
                                ;  See if program wants to pass some commands to TSKMON.
                                ;
006040  032761  000000C 000000G         BIT     #PASLIN!SCHAIN,LJSW(R1);DOES PROGRAM WANT TO PASS COMMANDS?
006046  001422                          BEQ     2$                  ;BR IF NOT
                                ;  Move commands from job chain data area to context area chain data cell.
006050  012704  000510                  MOV     #510,R4             ;ADDRESS OF CELL WITH BYTE COUNT
006054  106524                          MFPD    (R4)+               ;GET COUNT OF # BYTES TO MOVE
006056  012602                          MOV     (SP)+,R2
006060  020227  000266                  CMP     R2,#<1000-512>      ;LIMIT TO LARGEST POSSIBLE SIZE
006064  101402                          BLOS    3$                  ;BR IF OK
006066  012702  000266                  MOV     #<1000-512>,R2
006072  012703  0000100         3$:     MOV     #CINDAT+10,R3       ;POINT TO CHAIN DATA CELL IN CONTEXT AREA
006076  010223                          MOV     R2,(R3)+            ;STORE BYTE COUNT
006100  001405                          BEQ     2$                  ;BR IF NULL COMMAND BEING PASSED TO US
006102  005202                          INC     R2                  ;ROUND UP
006104  006202                          ASR     R2                  ;GET # WORDS TO MOVE
006106  106524                  1$:     MFPD    (R4)+               ;GET NEXT WORD OF DATA TO MOVE
006110  012623                          MOV     (SP)+,(R3)+         ;MOVE TO CONTEXT AREA
006112  077203                          SOB     R2,1$               ;MOVE ALL OF COMMANDS
                                ;
                                ;  Stop program execution and enter TSKMON
                                ;
006114  004737  0000000         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.
                                ;
006120  013702  0000020         TTEMT:  MOV     EMTBLK+2,R2     ;Get function_value
006124  162702  000101                  SUB     #'A,R2          ;Convert letter into index value
006130  100416                  1$:     BMI     9$              ;Br if not legal value
006132  020227  0000000                 CMP     R2,#MAXCC       ;Is the letter ok?
006136  103013                          BHIS    9$              ;Br if too big
006140  006302                          ASL     R2              ;Convert to word table index
006142  013700  0000040                 MOV     EMTBLK+4,R0     ;Get control_value
006146  016237  006176' 006160'         MOV     TTFUN(R2),1$    ;Set address of processing routine
006154  004737  0000000                 CALL    OVRHC           ;Call overlay handler
006160  000000                  1$:     .WORD   0               ;Store address of routine to call here
006162  000137  0000000                 JMP     EMTXIT          ;Finished
                                ;
                                ;  Error -- Invalid function value
                                ;
006166  012700  000001          9$:     MOV     #1,R0           ;Return error code 1
006172  000137  0000000                 JMP     SETERR
```

```
                                ;-----------------------------------------------------------------
                                ;  Vector of addresses of processing routines
                                ;
006176  0000000               TTFUN:  .WORD   SETRBF           ;A
006200  0000000                       .WORD   CMDB             ;B
006202  0000000                       .WORD   CMDC             ;C
006204  0000000                       .WORD   GTSPAC           ;D
006206  0000000                       .WORD   CMDE             ;E
006210  0000000                       .WORD   CMDF             ;F
006212  0000000                       .WORD   CMDG             ;G
006214  0000000                       .WORD   CMDH             ;H
006216  0000000                       .WORD   CMDI             ;I
006220  0000000                       .WORD   CMDJ             ;J
006222  0000000                       .WORD   CMDK             ;K
006224  0000000                       .WORD   CMDL             ;L
006226  0000000                       .WORD   CMDM             ;M
006230  0000000                       .WORD   CMDN             ;N
006232  0000000                       .WORD   CMDO             ;O
006234  0000000                       .WORD   RSSPAC           ;P
006236  0000000                       .WORD   SFWAC            ;Q
006240  0000000                       .WORD   CMDR             ;R
006242  0000000                       .WORD   CMDS             ;S
006244  0000000                       .WORD   CMDT             ;T
006246  0000000                       .WORD   CMDU             ;U
006250  0000000                       .WORD   SFWL             ;V
006252  0000000                       .WORD   CMDW             ;W
006254  0000000                       .WORD   CMDX             ;X
006256  0000000                       .WORD   CMDY             ;Y
006260  0000000                       .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
                                ;
006262  032737  0000000 0000000 SNDMSG: BIT     #PO$SND,PRIVCO  ;Are we authorized to send messages?
006270  001003                          BNE     4$              ;Br if yes
006272  005000                          CLR     R0              ;Return error code 0
006274  000137  0000000                 JMP     SETERR
006300  032737  0000000 0000000 4$:     BIT     #PO$OPR,PRIVCO  ;Does user have OPER privilege?
006306  001003                          BNE     5$              ;Br if yes
006310  042737  000001  0000000         BIC     #1,EMTBLK       ;Cannot override gag unless oper privilege
                                ;
                                ;  See if a valid job number was specified.
                                ;
006316  013703  0000020         5$:     MOV     EMTBLK+2,R3     ;GET NUMBER OF LINE TO RECEIVE MESSAGE
006322  006303                          ASL     R3              ;CONVERT TO JOB INDEX NUMBER
006324  020327  000002                  CMP     R3,#2           ;CAN'T BE LESS THAN 2
006330  103473                          BLO     9$              ;BR IF TOO SMALL
006332  020327  0000000                 CMP     R3,#LSTSL       ;MAKE SURE NOT TOO LARGE
006336  101070                          BHI     9$              ;BR IF TOO LARGE
006340  032763  0000000 0000000         BIT     #$DILUP,LSW(R3) ;IS JOB LOGGED ON?
006346  001464                          BEQ     9$              ;BR IF NOT
006350  032763  0000000 0000000         BIT     #$DETCH,LSW(R3) ;IS JOB A DETACH LINE?
006356  001060                          BNE     9$              ;BR - DETACH LINES CAN'T RECEIVE
006360  032763  0000000 0000000         BIT     #$TTGAG,LSW7(R3);IS LINE GAGGED?
006366  001414                          BEQ     3$              ;BR IF NOT
006370  032763  0000000 0000000         BIT     #$INKMN,LSW4(R3);IS KMON RUNNING NOW?
006376  001010                          BNE     3$              ;BR IF YES
006400  032737  000001  0000000         BIT     #1,EMTBLK       ;DOES HE WANT TO OVERRIDE GAG?
006406  001004                          BNE     3$              ;BR IF YES
006410  012700  000001                  MOV     #1,R0           ;RETURN ERROR CODE 1 IF LINE IS GAGGED
006414  000137  0000000                 JMP     SETERR
                                ;
                                ;  Get a free message buffer.
                                ;
006420  004737  006526'         3$:     CALL    GTMSBF          ;GET A FREE MESSAGE BUFFER
006424  103004                          BCC     6$              ;CONTINUE IF NO ERROR
006426  012700  000002                  MOV     #2,R0           ;RETURN ERROR CODE 2 IF NO MESSAGE BUFFERS
006432  000137  0000000                 JMP     SETERR
                                ;
                                ;  Move message from user's buffer to system message buffer.
                                ;
006436  013703  0000040         6$:     MOV     EMTBLK+4,R3     ;GET ADDRESS OF USER'S MESSAGE BUFFER
006442  010300                          MOV     R3,R0           ;VALIDATE ADDRESS
006444  004737  0000000                 CALL    VALADB
006450  010402                          MOV     R4,R2           ;GET ADDRESS OF SYSTEM MESSAGE BUFFER BLOCK
```

SNDMSG -- Send a message to a line

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

```
                                        .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.
                                ;
006526                          GTMSBF: DISABL                  ;** DISABLE **
006534  005737  0000000               TST     NMUMB           ;ARE THERE ANY AVAILABLE MESSAGE BLOCKS?
006540  003004                        BGT     3$              ;BR IF SO
006542  032737  000002  0000000       BIT     #2,EMTBLK       ;DOES USER WANT TO WAIT?
006550  001035                        BNE     7$              ;BR IF NOT
006552  013704  0000000       3$:     MOV     SNMSHD,R4       ;GET ADDRESS OF 1ST FREE BLOCK
006556  001013                        BNE     1$              ;BR IF OK
006560  032737  000002  0000000       BIT     #2,EMTBLK       ;DOES USER WANT TO WAIT?
006566  001026                        BNE     7$              ;BR IF NOT
                                ;
                                ;  No message buffers available.
                                ;  Suspend job until one becomes available.
                                ;
006570  012700  0000000       2$:     MOV     #S$WSMB,R0      ;WAIT FOR MESSAGE BUFFER STATE.
006574  004737  0000000               CALL    @NSPND          ;WAIT FOR MESSAGE BUFFER ** ENABLE **
006600  004737  0000000               CALL    CHKABT          ;WERE WE ABORTED WHILE ASLEEP?
006604  000750                        BR      GTMSBF          ;TRY AGAIN
                                ;
                                ;  Got a buffer.  Unlink from free chain.
                                ;
006606  005337  0000000       1$:     DEC     NMUMB           ;ONE LESS FREE BUFFER
006612  016437  0000000 0000000       MOV     SB$LNK(R4),SNMSHD;REMOVE FROM FREE LIST
006620                                ENABL                   ;** ENABLE **
006626  010400                        MOV     R4,R0           ;INITIALIZE SB$PNT
006630  062700  0000000               ADD     #SB$TXT,R0
006634  010064  0000000               MOV     R0,SB$PNT(R4)
006640  000241                        CLC                     ;SAY NO ERRORS
006642  000401                        BR      9$
                                ;
                                ;  No free message buffers, flag error return
                                ;
006644  000261                7$:     SEC
                                ;
                                ;  Finished
                                ;
006646  000207                9$:     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.
                                ;
006650  010146                  QMSG:   MOV     R1,-(SP)
                                ;
                                ;  Add message buffer to queue for line
                                ;
006652                                  DISABL                  ;** DISABLE **
006660  016101  0000000                 MOV     LNPRIM(R1),R1   ;OBTAIN THE CURRENT PRIMARY LINE NUMBER
006664  016100  0000000                 MOV     LMSGBF(R1),R0   ;GET ADDRESS OF HEAD OF MESSAGE LIST FOR JOB
006670  001411                          BEQ     3$              ;BR IF NO MESSAGES QUEUED FOR JOB
006672  005760  0000000         1$:     TST     SB$LNK(R0)      ;IS THIS LAST ENTRY IN LIST?
006676  001403                          BEQ     2$              ;BR IF YES
006700  016000  0000000                 MOV     SB$LNK(R0),R0   ;CHAIN TO NEXT QUEUED MESSAGE BLOCK
006704  000772                          BR      1$
006706  010460  0000000         2$:     MOV     R4,SB$LNK(R0)   ;ADD OUR MESSAGE TO TAIL OF LIST
006712  000402                          BR      4$
006714  010461  0000000         3$:     MOV     R4,LMSGBF(R1)   ;ADD OUR MESSAGE BLOCK TO LIST FOR JOB
006720  005064  0000000         4$:     CLR     SB$LNK(R4)      ;SAY WE ARE AT END OF LIST
006724                                  ENABL                   ;** ENABLE **
                                ;
                                ;  Initiate transmission to the line.
                                ;
006732  116101  0000000                 MOVB    LNMAP(R1),R1    ;GET VIRTUAL LINE #
006736  004777  0000000                 CALL    @TRNSTR         ;INITIATE TRANSMISSION
                                ;
                                ;  Finished
                                ;
006742  012601                          MOV     (SP)+,R1
006744  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 006746                         ASTXIT:
      6                                ;
      7                                ;  Set up information that will sidetrack control in EMTXIT.
      8                                ;
      9 006746  013702  0000000               MOV     EMTCAD,R2       ;Get pointer to top entry on return stack
     10 006752  020227  0000000               CMP     R2,#EMTCAS      ;Were we expecting a completion routine exit?
     11 006756  001002                        BNE     1$              ;Br if yes
     12 006760  000137  0000000               JMP     BADEMT          ;Illegal EMT if nothing on cpl rtn return stk
     13 006764  012237  0000000     1$:       MOV     (R2)+,EMTRAD    ;Set return address for compl routine
     14 006770  010237  0000000               MOV     R2,EMTCAD       ;Save new stack pointer
     15 006774  000137  0000000               JMP     EMTXIT          ;EMTXIT will do the actual exit for us
```

```
                                        .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.
                                ;
 007000  105737  0000000        ESPCPS: TSTB    CURCP           ;ARE WE IN A COMPLETION ROUTINE NOW?
 007004  001003                         BNE     1$              ;BR IF YES
 007006  005000                         CLR     R0              ;RETURN ERROR CODE 0 IF NOT
 007010  000137  0000000                JMP     SETERR
 007014  005737  0000000        1$:     TST     SPCPS           ;IS ANOTHER .SPCPS PENDING NOW?
 007020  001404                         BEQ     2$              ;BR IF NOT
 007022  012700  000001                 MOV     #1,R0           ;RETURN ERROR CODE 1 IF YES
 007026  000137  0000000                JMP     SETERR
 007032  013700  0000020        2$:     MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S INFORMATION BLOCK
 007036  004737  0000000                CALL    VALADW          ;VALIDATE THE ADDRESS
 007042  010037  0000000                MOV     R0,SPCPS        ;SAVE ADDRESS OF INFO BLOCK
 007046  000137  0000000                JMP     EMTXIT          ;EXIT -- SYSXIT WILL DO REAL PROCESSING
```

```
                                        .SBTTL  EMT376 -- EMT 376 Processing
                                ;-------------------------------------------------------------------------------
                                ;  Process EMT 376
                                ;
                                ;  Inputs:
                                ;   R4 = EMT function code.
                                ;
007052  120427  000376          EMT376: CMPB    R4,#376         ;Is this an EMT 376?
007056  001011                          BNE     1$              ;Br if not
007060  013704  0000000G                MOV     EMTADR,R4       ;Get address of EMT instruction
007064  106564  000002                  MFPD    2(R4)           ;Get value that follows instruction
007070  012600                          MOV     (SP)+,R0        ;Get code word
007072  000300                          SWAB    R0              ;Get code byte to low-order
007074  120027  000373                  CMPB    R0,#373         ;Overlay I/O error?
007100  001402                          BEQ     2$              ;Br if yes
007102  000137  0000000G          1$:     JMP     BADEMT          ;Invalid EMT if not
                                ;
                                ;  Overlay I/O error
                                ;
007106  010437  0000000G          2$:     MOV     R4,ABRTAD       ;Set address of abort
007112  012737  177750  0000000G        MOV     #-30,ABRTCD     ;Set abort code
007120  004737  0000000G                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
                                ;
007124  113705  000030          JBINFO: MOVB    EMTBLK+3,R5     ;GET SUB-FUNCTION NUMBER
007130  020527  000011                  CMP     R5,#MXJIFN      ;IS IT TOO BIG?
007134  101404                          BLOS    1$              ;BR IF OK
007136  012700  000001                  MOV     #1,R0           ;ERROR CODE 1 IF TOO BIG
007142  000137  000000G                 JMP     SETERR
                                ;
                                ;  Subfunction number is valid.
                                ;  Check job number.
                                ;
007146  113701  000020          1$:     MOVB    EMTBLK+2,R1     ;GET SPECIFIED JOB NUMBER
007152  001404                          BEQ     2$              ;ZERO IS INVALID
007154  006301                          ASL     R1              ;CONVERT TO LINE INDEX NUMBER
007156  020127  000000G                 CMP     R1,#LSTSL       ;IS IT A VALID NUMBER?
007162  101404                          BLOS    3$              ;BR IF YES
007164  012700  000002          2$:     MOV     #2,R0           ;ERROR 2 IF INVALID LINE NUMBER
007170  000137  000000G                 JMP     SETERR
                                ;
                                ;  See if line is currently logged on
                                ;
007174  032761  000000G 000000G 3$:     BIT     #$KINIT,LSW(R1) ;IS THE LINE LOGGED ON NOW?
007202  001003                          BNE     4$              ;BR IF YES
007204  005000                          CLR     R0              ;ERROR # 0 IF NOT
007206  000137  000000G                 JMP     SETERR
                                ;
                                ;  Check the address specified for the user's result buffer
                                ;
007212  013704  000040          4$:     MOV     EMTBLK+4,R4     ;GET USER'S VIRTUAL ADDRESS
007216  010400                          MOV     R4,R0
007220  004737  000000G                 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.
                                ;
007224  006305                          ASL     R5              ;CONVERT SUB-FUNCTION # TO WORD INDEX
007226  000175  007562'                 JMP     @JIJMPX(R5)     ;ENTER PROCESSING ROUTINE
                                ;
                                ;  #0 -- Get job status flags
                                ;
007232  005000                  JISTAT: CLR     R0              ;DEVELOP FLAGS IN R0
```

JBINFO -- Get information about a specific job

```
                                        ;  See if job is detached or virtual
007234  020127  0000000                         CMP     R1,#LSTPL       ;PRIMARY LINE?
007240  101410                                  BLOS    1$              ;BR IF YES
007242  020127  0000000                         CMP     R1,#LSTDL       ;DETACHED LINE?
007246  101403                                  BLOS    2$              ;BR IF YES
007250  052700  0000000                         BIS     #JIVLN,R0       ;SET VIRTUAL-LINE FLAG
007254  000402                                  BR      1$
007256  052700  0000000                 2$:     BIS     #JIDLN,R0       ;SET DETACHED-LINE FLAG
                                        ;  See if job is locked in memory
007262  032761  0000000 0000000       1$:     BIT     #$MLOCK,LSW6(R1);IS JOB LOCKED IN MEMORY?
007270  001402                                  BEQ     4$              ;BR IF NOT
007272  052700  0000000                         BIS     #JIMLOK,R0      ;SET MEMORY-LOCKED FLAG
                                        ;  See if job has operator privilege
007276  010046                          4$:     MOV     R0,-(SP)        ;SAVE OUR STATUS CODES
007300                                          OCALL   GETCXT          ;GET EXCLUSIVE ACCESS TO CONTEXT BUFFER
007306  103002                                  BCC     6$              ;BR IF GOT IT
007310  004737  0000000                         CALL    STOP            ;JOB WAS ABORTED WHILE WAITING
007314  010102                          6$:     MOV     R1,R2           ;GET INDEX # OF JOB WE ARE COPYING FROM
007316  012703  0000000                         MOV     #PRIVCO,R3      ;ADDR OF INFO TO GET
007322  012700  000002                          MOV     #2,R0           ;GET # BYTES TO COPY
007326                                          OCALL   REDCXT          ;GET INFO FROM CONTEXT BLOCK OF OTHER JOB
007334  032777  0000000 0000000               BIT     #P0$SYS,@CXTBUF ;IS THIS A PRIVILEGED JOB?
007342  001402                                  BEQ     5$              ;BR IF NOT
007344  052716  0000000                         BIS     #JIPRIV,(SP)    ;SET PRIVILEGE FLAG
007350                                  5$:     OCALL   FRECXT          ;FREE THE CONTEXT BUFFER
007356  012600                                  MOV     (SP)+,R0        ;GET STATUS FLAGS
007360  000422                                  BR      JIRET1          ;GO RETURN VALUE
                                        ;
                                        ;  #1 -- Get job's run state
                                        ;
007362  012700  007606'         JIRUN:  MOV     #JIRNST,R0      ;Point to state table
007366  116102  0000000                 MOVB    LSTATE(R1),R2   ;Get job's current execution state
007372  120220                  1$:     CMPB    R2,(R0)+        ;Search for state in table
007374  001406                          BEQ     2$              ;Br if found it
007376  005200                          INC     R0              ;Skip over value
007400  020027  007672'                 CMP     R0,#JIRNND      ;Checked all entries?
007404  103772                          BLO     1$              ;Br if yes
007406  005000                          CLR     R0              ;Don't recognize state, return 0
007410  000406                          BR      JIRET1          ;Return the value
007412  111000                  2$:     MOVB    (R0),R0         ;Get value to return
007414  000404                          BR      JIRET1          ;Return the value
                                        ;
                                        ;  #2 -- Get amount of memory in use by job
                                        ;
007416  016100  0000000         JIMEM:  MOV     LNBLKS(R1),R0   ;GET # 256-WORD BLOCKS USED BY JOB
007422  066100  0000000                 ADD     LNSBLK(R1),R0   ;ADD # BLOCKS USED BY PLAS REGIONS
007426  010046                  JIRET1: MOV     R0,-(SP)        ;STORE VALUE INTO USER'S BUFFER
007430  106614                          MTPD    (R4)
007432  000137  0000000                 JMP     EMTXIT          ;FINISHED WITH EMT
                                        ;
                                        ;  #3 -- Get job connect time (minutes)
                                        ;
007436  013700  0000000         JICONT: MOV     MINTIM,R0       ;GET CURRENT SYSTEM UP-TIME
007442  166100  0000000                 SUB     LCONTM(R1),R0   ;GET # MINUTES JOB HAS BEEN CONNECTED
007446  000767                          BR      JIRET1
                                        ;
                                        ;  #4 -- Get base 256-word block # assigned to job area
```

```
                        ;
007450  016100  0000000 JIBASE: MOV     LBASE(R1),R0    ;GET BASE BLOCK #
007454  000764                 BR      JIRET1
                        ;
                        ;  #5 -- Get name of program being run by job
                        ;
007456  016102  0000000 JIPROG: MOV     LPRG1(R1),R2    ;GET FIRST 3 CHARS OF NAME
007462  016103  0000000         MOV     LPRG2(R1),R3    ;GET SECOND 3 CHARS OF NAME
007466  000411                 BR      JIRET2
                        ;
                        ;  #6 -- Get project/programmer number
                        ;
007470  016102  0000000 JIPPN:  MOV     LPROJ(R1),R2    ;GET PROJECT NUMBER
007474  016103  0000000         MOV     LPROG(R1),R3    ;GET PROGRAMMER NUMBER
007500  000404                 BR      JIRET2
                        ;
                        ;  #7 -- Get CPU time used by job
                        ;
007502  016102  0000000 JICPU:  MOV     LCPUHI(R1),R2   ;GET HIGH-ORDER CPU TIME
007506  016103  0000000         MOV     LCPULO(R1),R3   ;GET LOW-ORDER CPU TIME
007512  010246          JIRET2: MOV     R2,-(SP)        ;RETURN FIRST WORD OF RESULT
007514  106624                 MTPD    (R4)+
007516  010346                 MOV     R3,-(SP)        ;RETURN SECOND WORD OF RESULT
007520  106624                 MTPD    (R4)+
007522  000137  0000000        JMP     EMTXIT          ;FINISHED WITH EMT
                        ;
                        ;  #8 -- Job execution priority
                        ;
007526  116100  0000000 JIPRIO: MOVB    LPRI(R1),R0     ;Get current execution priority for job
007532  000735                 BR      JIRET1          ;Return
                        ;
                        ;  #9 -- Job name
                        ;
007534  010105          JINAME: MOV     R1,R5           ;Copy job index number
007536  012702  000006         MOV     #6,R2           ;6 words per name
007542  070502                 MUL     R2,R5           ;Name should be 12 bytes long
007544  062705  0000000        ADD     #LUNAME,R5      ;Convert to name table index
007550  012546          1$:    MOV     (R5)+,-(SP)     ;Fetch next two bytes of name
007552  106624                 MTPD    (R4)+           ;Move to user's buffer
007554  077203                 SOB     R2,1$           ;Loop through whole name
007556  000137  0000000        JMP     EMTXIT          ;Finished with EMT
                        ;
                        ;  Branch table used to enter job information routines
                        ;
007562  007232' JIJMPX: .WORD   JISTAT          ; 0 -- GET JOB STATE FLAGS
007564  007362'         .WORD   JIRUN           ; 1 -- GET JOB RUN STATE CODE
007566  007416'         .WORD   JIMEM           ; 2 -- GET AMT OF MEMORY USED BY JOB
007570  007436'         .WORD   JICONT          ; 3 -- GET CONNECT TIME FOR JOB
007572  007450'         .WORD   JIBASE          ; 4 -- GET BASE ADDRESS FOR JOB
007574  007456'         .WORD   JIPROG          ; 5 -- GET NAME OF PROGRAM BEING RUN
007576  007470'         .WORD   JIPPN           ; 6 -- GET PROJECT/PROGRAMMER #
007600  007502'         .WORD   JICPU           ; 7 -- GET CPU TIME USED BY JOB
007602  007526'         .WORD   JIPRIO          ; 8 -- GET JOB EXECUTION PRIORITY
007604  007534'         .WORD   JINAME          ; 9 -- GET JOB NAME
000011          MXJIFN  =       <<.-JIJMPX>/2>-1
```

```
                                        ;
                                        ;  Table used to convert TSX scheduler state codes into
                                        ;  externally defined run states.
                                        ;
007606  0000  001   JIRNST:  .BYTE   S$TWFN,1          ;Timed wait completion
007610  0000  001           .BYTE   S$OTLO,1          ;Terminal output buffer almost empty
007612  0000  001           .BYTE   S$IOFN,1          ;I/O completion
007614  0000  002           .BYTE   S$CPU,2           ;Normal priority execution
007616  0000  003           .BYTE   S$LOW,3           ;Fixed low priority execution
007620  0000  004           .BYTE   S$INWT,4          ;Waiting for terminal input
007622  0000  005           .BYTE   S$OTWT,5          ;Waiting for terminal output buffer space
007624  0000  006           .BYTE   S$TMWT,6          ;Waiting for timed interval
007626  0000  007           .BYTE   S$SPND,7.         ;Doing .SPND
007630  0000  010           .BYTE   S$SFWT,8.         ;Shared file wait
007632  0000  011           .BYTE   S$MSWT,9.         ;Waiting for message
007634  0000  012           .BYTE   S$QUSR,10.        ;Waiting for USR for file operation
007636  0000  013           .BYTE   S$IOWT,11.        ;Waiting for I/O operation
007640  0000  013           .BYTE   S$NEDQ,11.        ;Waiting for I/O queue element
007642  0000  013           .BYTE   S$QMIO,11.        ;Waiting for mapped I/O buffer
007644  0000  013           .BYTE   S$QCCB,11.        ;Waiting for cache control block
007646  0000  013           .BYTE   S$QSPD,11.        ;Waiting for special device data base
007650  0000  013           .BYTE   S$WSMB,11.        ;Waiting for system message buffer
007652  0000  014           .BYTE   S$SPDB,12.        ;Waiting for spooled device control block
007654  0000  014           .BYTE   S$SPCB,12.        ;Waiting for spooled device control block
007656  0000  015           .BYTE   S$TTSC,13.        ;Single char terminal input complete
007660  0000  015           .BYTE   S$TTFN,13.        ;Non single char terminal input complete
007662  0000  015           .BYTE   S$OTFN,13.        ;Terminal buffer empty
007664  0000  015           .BYTE   S$HICP,13.        ;Interactive job execution
007666  0000  016           .BYTE   S$RT,14.          ;Fixed high priority
007670  0000  017           .BYTE   S$WFM,15.         ;Waiting for memory expansion
007672                JIRNND:                         ;End of table
                            .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.
                                ;
007672  012702  007756'         TRMEMT: MOV     #LOTRM,R2       ;POINT TO START OF TERMINAL TYPE TABLE
007676  016104  000000G                 MOV     LTRMTP(R1),R4   ;GET TERMINAL TYPE FLAGS FOR OUR JOB
007702  116100  000000G                 MOVB    LNPRIM(R1),R0   ;Get out primary line number
007706  032760  000000G 000000G         BIT     #$V52EM,LSW11(R0);Are we emulating a VT52?
007714  001402                         BEQ     1$              ;Br if not
007716  012704  000000G                 MOV     #VT52,R4        ;Say terminal is a VT52
007722  020422                 1$:     CMP     R4,(R2)+        ;SEARCH FOR MATCHING PATTERN IN TABLE
007724  001405                         BEQ     2$              ;BR IF FOUND
007726  020227  010002'                 CMP     R2,#HITRM       ;HAVE WE SEARCHED ALL OF TABLE?
007732  101773                         BLOS    1$              ;BR IF NOT
007734  005002                         CLR     R2              ;SAY UNKNOWN TERMINAL TYPE
007736  000403                         BR      3$
007740  162702  007756'         2$:     SUB     #LOTRM,R2       ;CONVERT TABLE OFFSET TO TERMINAL CODE NUMBER
007744  006202                         ASR     R2
007746  010237  000000G         3$:     MOV     R2,UR0          ;RETURN TERMINAL TYPE CODE # IN USER'S R0
007752  000137  000000G                 JMP     EMTXIT          ;FINISHED
                                ;
                                ;  Table to translate terminal type flags into code number.
                                ;
007756  000000G                 LOTRM:  .WORD   VT52            ;   1 - VT52
007760  000000G                         .WORD   VT100           ;   2 - VT100
007762  000000G                         .WORD   HAZEL           ;   3 - Hazeltine
007764  000000G                         .WORD   ADM3A           ;   4 - ADM3A
007766  000000G                         .WORD   LA36            ;   5 - LA36
007770  000000G                         .WORD   LA120           ;   6 - LA120
007772  000000G                         .WORD   DIABLO          ;   7 - Diablo
007774  000000G                         .WORD   QUME            ;   8 - Qume
007776  000000G                         .WORD   VT2007          ;   9 - VT200 with 7 bit control codes
010000  000000G                         .WORD   VT2008          ;  10 - VT200 with 8 bit control codes
010002                          HITRM:

                                        .SBTTL  Get Project-Programmer number
                                ;-------------------------------------------------------------------------
                                ;  This emt is used to obtain the current Project-Programmer number
                                ;  that a job is running under.
                                ;
010002  013700  000020G         GETPPN: MOV     EMTBLK+2,R0     ;GET ADDRESS OF CELL TO RECEIVE PPN
010006  004737  000000G                 CALL    VALADW          ;VALIDATE THE ADDRESS
010012  016146  000000G                 MOV     LPROJ(R1),-(SP) ;MOVE PROJECT # TO USER'S BUFFER
010016  106620                         MTPD    (R0)+
010020  016146  000000G                 MOV     LPROG(R1),-(SP)
010024  106620                         MTPD    (R0)+           ;MOVE PROGRAMMER # TO USER'S BUFFER
010026  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
                                ;
 010032  105737  0000000        XGTLN:  TSTB    EMTBLK          ;WANT CURRENT LINE OR RELATIVE
 010036  001440                  BEQ     1$              ;BR IF CURRENT
                                ;
                                ;  THIS IS A REQUEST TO RETURN THE REAL LINE NUMBER OF A SUBPROCESS
                                ;
 010040  020127  0000000                CMP     R1,#LSTPL       ;ARE WE A PRIMARY LINE?
 010044  101406                         BLOS    4$              ;BR IF OK
 010046  020127  0000000                CMP     R1,#LSTDL       ;ARE WE A VIRTUAL LINE?
 010052  101003                         BHI     4$              ;BR IF OK
 010054  012700  000002                 MOV     #2,R0           ;DETACHED JOB'S DON'T HAVE SUBPROCESSES
 010060  000425                         BR      6$              ;GO COMPLAIN ABOUT IT
 010062  016101  0000000        4$:     MOV     LNPRIM(R1),R1   ;GET OUR PARENT INDEX
 010066  013705  0000020                MOV     EMTBLK+2,R5     ;WANTS LINE NUMBER FOR THIS SUBPROCESS
 010072  001002                         BNE     2$              ;BR IF NOT ASKING FOR PARENT
 010074  010105                         MOV     R1,R5           ;COPY PARENT'S PROCESS INDEX
 010076  000411                         BR      3$              ;DON'T NEED SUBPROCESS MAPPING
 010100  020527  0000000        2$:     CMP     R5,#MAXSEC      ;VALID SUBPROCESS NUMBER?
 010104  101403                         BLOS    5$              ;BR IF OK
 010106  012700  000001                 MOV     #1,R0           ;SIGNAL INVALID SUBPROCESS #
 010112  000410                         BR      6$              ;AND GO REPORT ERROR
 010114  066105  0000000        5$:     ADD     LSECPT(R1),R5   ;POINT TO PRIMARY LINE'S SUBPROCESS TABLE
 010120  114505                         MOVB    -(R5),R5        ;GET SPECIFIED LINE'S INDEX (^1 IS OFFSET 0)
 010122  006205                 3$:     ASR     R5              ;CONVERT INDEX TO LINE #
 010124  010537  0000000                MOV     R5,URO          ;PASS TO USER
 010130  001012                         BNE     XJNOP           ;IF IN USE, RETURN LINE # TO USER
 010132  005000                         CLR     R0              ;ELSE REPORT LINE NOT ACTIVE
 010134  000137  0000000        6$:     JMP     SETERR          ;RETURN WITH ERROR
                                ;
                                ;  THIS IS A REQUEST TO GET OUR CURRENT LINE NUMBER
                                ;
 010140  006201                 1$:     ASR     R1              ;CONVERT TO LINE #
 010142  010137  0000000                MOV     R1,URO          ;RETURN TO USER IN R0
 010146  000403                         BR      XJNOP
                                ;
                                ;-------------------------------------------------------------
                                ;  THE .SPLSP EMT RETURNS IN R0 THE NUMBER OF FREE BLOCKS
                                ;  IN THE SPOOL FILE.
                                ;
 010150  013737  0000000 0000000 XSPLSP: MOV     NFRESB,URO      ;RETURN # OF BLOCKS TO USER IN R0
 010156  000137  0000000        XJNOP:  JMP     EMTXIT
                                ;
                                ;-------------------------------------------------------------
                                ;  SET/RESET ODT CHARACTER ACTIVATION MODE
                                ;
 010162  105737  0000000        XODTMD: TSTB    EMTBLK          ;SET OR RESET MODE?
 010166  001407                         BEQ     1$              ;BR IF RESET
 010170  052761  0000000 0000000       BIS     #$ODTMD,LSW4(R1);TURN ON ODT MODE
 010176  052761  0000000 0000000       BIS     #$DBGMD,LSW6(R1);SAY WE ARE IN DEBUGGER TERMINAL CONTROL MODE
 010204  000764                         BR      XJNOP
 010206  042761  0000000 0000000 1$:     BIC     #$ODTMD,LSW4(R1);TURN OFF ODT MODE
```

```
      58 010214  042761  000000G 000000G          BIC     #$DBGMD,LSW6(R1);RESET DEBUGGER TERMINAL MODE
      59 010222  000755                          BR      XJNOP
      60                                ;
      61                                ;-------------------------------------------------------------------------
      62                                ;  Check to see if any input activation characters are pending.
      63                                ;  Return with c-flag set if not.
      64                                ;
      65 010224  005761  000000G        CKINAC: TST     LACTIV(R1)      ;ANY PENDING ACTIVATION CHARS?
      66 010230  001002                          BNE     1$              ;BR IF YES
      67 010232  000137  000000G                 JMP     SETC            ;RETURN WITH C-FLAG SET
      68 010236  000137  000000G        1$:     JMP     EMTXIT          ;RETURN WITH C-FLAG CLEAR
      69                                ;
      70                                ;-------------------------------------------------------------------------
      71                                ;  Return in R0 the TSX license number of this site.
      72                                ;
      73 010242  013737  000000G 000000G GTLICN: MOV     TSXSIT,UR0      ;PUT LICENSE # IN USER'S R0
      74 010250  000742                          BR      XJNOP           ;RETURN
```

```
                                ;-------------------------------------------------------------------------
                                ;  Set interactive or non-interactive mode for current program.
                                ;
010252  005737  0000020       INTEMT: TST     EMTBLK+2        ;Set or reset interactive mode?
010256  001404                        BEQ     1$              ;Reset interactive move
                                ;
                                ;  Set interactive mode
                                ;
010260  042761  0000000 0000000       BIC     #$NOINT,LSW7(R1);Clear non-interactive mode flag
010266  000405                        BR      9$
                                ;
                                ;  Reset interactive mode
                                ;
010270  052761  0000000 0000000 1$:   BIS     #$NOINT,LSW7(R1);Run job non-interactively
010276  005061  0000000               CLR     LITIME(R1)      ;Clear interactive timer
                                ;
                                ;  Finished
                                ;
010302  000137  0000000       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
                                ;
010306  113705  000000G        SYFEMT: MOVB    EMTBLK,R5       ;GET SYSTEM FILE INDEX
010312  020527  000004                CMP     R5,#MXSYFN      ;VALID FILE INDEX NUMBER?
010316  101403                        BLOS    1$              ;BRANCH IF OK
010320  005000                        CLR     R0              ;ERROR CODE 0 IF BAD INDEX
010322  000137  000000G               JMP     SETERR
                                ;
                                ;  Convert system file index number to table offset
                                ;
010326  006305                 1$:     ASL     R5              ;CONVERT INDEX TO TABLE OFFSET
010330  016504  010364'               MOV     SYFTBL(R5),R4   ;GET POINTER TO FILE NAME
                                ;
                                ;  Get and validate user's return buffer
                                ;
010334  013705  000002G               MOV     EMTBLK+2,R5     ;GET USER'S VIRTUAL ADDRESS
010340  010500                        MOV     R5,R0
010342  004737  000000G               CALL    VALADW          ;MAKE SURE ADDRESS IS OK
                                ;
                                ;  Now return 4 RAD50 words of dev:filspc.ext to user buffer
                                ;
010346  012700  000004                MOV     #4,R0           ;MOVE 4 WORDS
010352  012446                 2$:     MOV     (R4)+,-(SP)     ;GET NEXT RAD50 WORD
010354  106625                        MTPD    (R5)+           ;MOVE TO USER BUFFER
010356  077003                        SOB     R0,2$
010360  000137  000000G               JMP     EMTXIT          ;FINISHED
                                ;
                                ;  System file specification pointers
                                ;
010364  000000G                 SYFTBL: .WORD   SWDBLK           ; 0 SWAP FILE
010366  000000G                         .WORD   SPLBLK          ; 1 SPOOL FILE
010370  000000G                         .WORD   RSFBLK          ; 2 PLAS REGION SWAP FILE
010372  000000G                         .WORD   UCLDAT          ; 3 USER DEFINED COMMAND (UCL) FILE
010374  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
                                ;
010376                          EMTSPD:
                                ;
                                ;  Get line number and see if it is valid
                                ;
010376  013702  000020                  MOV     EMTBLK+2,R2     ;Get line number
010402  006302                          ASL     R2              ;Convert to line index number
010404  001001                          BNE     1$              ;Br if line number specified
010406  010102                          MOV     R1,R2           ;Use current line
010410  020227  000000          1$:     CMP     R2,#LSTSL       ;Primary or virtual line?
010414  101002                          BHI     10$             ;Br if not
010416  016202  000000                  MOV     LNPRIM(R2),R2   ;Map virtual line # to real line #
010422  020227  000000          10$:    CMP     R2,#LSTHL       ;Is this a valid line number?
010426  101010                          BHI     5$              ;Br if not
010430  032762  000000  000000          BIT     #$HARD,LSW3(R2) ;Line connected to hardware?
010436  001404                          BEQ     5$              ;Br if not
010440  032762  000000  000000          BIT     #$DEAD,LSW3(R2) ;Is line installed?
010446  001404                          BEQ     4$              ;Br if yes
010450  012700  000002          5$:     MOV     #2,R0           ;Return error code 2 for invalid line #
010454  000137  000000                  JMP     SETERR
                                ;
                                ;  See if we are privileged to perform this operation
                                ;
010460  032737  000000  000000  4$:     BIT     #P2$TRM,PRIVC2  ;Do we have privilege to change speed?
010466  001011                          BNE     11$             ;Br if yes
010470  105737  000000                  TSTB    EMTBLK          ;EMT to reset XOFF status or line speed EMT?
010474  001402                          BEQ     12$             ;Br if changing line speed
010476  020201                          CMP     R2,R1           ;Resetting XOFF for our own line?
010500  001404                          BEQ     11$             ;Br if yes
010502  012700  000001          12$:    MOV     #1,R0           ;Return error code 1
010506  000137  000000                  JMP     SETERR
                                ;
                                ;  See if this is actually the EMT to reset terminal XOFF status
                                ;
010512  010201                  11$:    MOV     R2,R1           ;Get line index number to R1
010514  123727  000000  000001          CMPB    EMTBLK,#1       ;Set line speed, reset XOFF or DTR?
010522  002416                          BLT     2$              ;Br if want to set speed
010524  001422                          BEQ     20$             ;Br if want to reset XOFF
                                ;
                                ;  Want to raise or lower DTR, get current data set status
                                ;
010526  004737  000000                  CALL    GETDSS          ;Get data set status into R0
010532  042700  000000                  BIC     #MS$DTR,R0      ;Assume we want to lower DTR
010536  123727  000000  000002          CMPB    EMTBLK,#2       ;Should we raise DTR?
010544  001002                          BNE     13$             ;Br if not
010546  052700  000000                  BIS     #MS$DTR,R0      ;Set DTR up bit
010552  004737  000000          13$:    CALL    SETDSS          ;Reset data set status
010556  000413                          BR      30$             ;And exit
```

Set transmit/receive speed for a line

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

```
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ;  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.
                                ;
010612  012702  000400          XGTTAB: MOV     #400,R2         ;STORE VALUES HERE
010616  012703  000372                  MOV     #372,R3         ;STORE VECTOR HERE
                                ;  TSLICH -- TSX command lead-in character.
010622  113746  000000G                MOVB    VTSLCH,-(SP)    ;GET LEAD-IN CHARACTER
010626  106623                         MTPD    (R3)+           ;MOVE TO JOB SPACE
                                ;  Name of running program.
010630  010246                         MOV     R2,-(SP)        ;SET POINTER TO PROGRAM NAME
010632  106623                         MTPD    (R3)+           ;IN VECTOR
010634  012700  000004                MOV     #4,R0           ;GET # WORDS TO MOVE
010640  012704  000000G                MOV     #RUNDEV,R4      ;POINT TO NAME OF RUNNING PROGRAM
010644  012446                  1$:    MOV     (R4)+,-(SP)     ;MOVE NAME TO JOB SPACE
010646  106622                         MTPD    (R2)+
010650  077003                         SOB     R0,1$
                                ;
                                ;  Tell user that vector is located at 350.
                                ;
010652  012737  000350  000000G        MOV     #350,URO        ;RETURN VECTOR ADDRESS IN USER'S R0
010660  000137  000000G                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.
                                ;
010664  113705  0000000         TBLEMT: MOVB    EMTBLK,R5       ;GET SUB-FUNCTION CODE
010670  001412                         BEQ     1$              ;BR IF IT IS GET-VALUE
                                ;
                                ;  Operator privilege is required to modify a table.
                                ;
010672  032737  0000000 0000000         BIT     #PO$SYS,PRIVCO  ;IS THIS USER PRIVILEGED?
010700  001004                         BNE     2$              ;BR IF YES
010702  012700  000001                 MOV     #1,R0           ;RETURN ERROR CODE OF 1 IF NOT
010706  000137  0000000                JMP     SETERR
010712  013703  0000040         2$:     MOV     EMTBLK+4,R3     ;GET VALUE FOR STORE
010716  013702  0000020         1$:     MOV     EMTBLK+2,R2     ;GET TABLE INDEX
                                ;
                                ;  If subfunction code = 4 then branch off to special processing routine
                                ;
010722  020527  000004                 CMP     R5,#4           ;IS THIS A SPECIAL TABLE OPERATION?
010726  001006                         BNE     11$             ;BR IF NOT
010730  006302                         ASL     R2              ;GET WORD TABLE INDEX FOR OPERATION
010732  020227  000000                 CMP     R2,#MXSPTF      ;MAKE SURE RANGE IS OK
010736  101043                         BHI     9$              ;BR IF INVALID
010740  000172  011136'                JMP     @SPLTF(R2)      ;ENTER PROCESSING ROUTINE
                                ;
                                ;  This is not a special table function
                                ;
010744  006302                 11$:    ASL     R2              ;CONVERT TO WORD TABLE VALUE
010746  002007                         BGE     4$              ;BR IF POS TABLE
                                ;
                                ;  Negative table index
                                ;
010750  005402                         NEG     R2              ;MAKE TABLE INDEX POSITIVE
010752  020227  000006                 CMP     R2,#MXNGTX      ;IS VALUE TOO BIG?
010756  103033                         BHIS    9$              ;BR IF TOO BIG
010760  016204  011164'                MOV     NEGTAB(R2),R4   ;GET TABLE ADDRESS
010764  000406                         BR      3$
                                ;
                                ;  Positive table index
                                ;
010766  020227  000024         4$:     CMP     R2,#MXPSTX      ;IS TABLE INDEX TOO BIG?
010772  103025                         BHIS    9$              ;BR IF TOO BIG
010774  016204  011140'                MOV     POSTAB(R2),R4   ;GET TABLE ADDRESS
011000  060104                         ADD     R1,R4           ;INDEX BY LINE NUMBER
                                ;
                                ;  Perform requested operation on table
                                ;
011002  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
011012  011437  000000G       MOV     @R4,UR0         ;RETURN TABLE VALUE IN USER'S R0
011016  000411                  BR      10$
                ;  Store value into table.
011020  010314          6$:     MOV     R3,@R4          ;STORE INTO TABLE
011022  000407                  BR      10$
011024  020527  000002  5$:     CMP     R5,#2           ;BIT SET OR RESET?
011030  001002                  BNE     7$              ;BR IF BIT CLEAR
                ;  Set bits in table
011032  050314                  BIS     R3,@R4          ;SET BITS
011034  000402                  BR      10$
                ;  Clear bits in table
011036  040314          7$:     BIC     R3,@R4          ;CLEAR BITS IN TABLE
011040  000400                  BR      10$
                ;
                ;  Finished
                ;
011042  000137  000000G 10$:     JMP     EMTXIT
                ;
                ;  Invalid table index
                ;
011046  005000          9$:     CLR     R0              ;RETURN ERROR CODE 0
011050  000137  000000G         JMP     SETERR
                ;
                ;  Special table function routines
                ;
                ;  1 -- Set maximum execution priority for job
                ;
011054  005703  SETMXP: TST     R3              ;PRIORITY MUST BE > 0
011056  003403          BLE     1$              ;BR IF INVALID
011060  020327  000000G CMP     R3,#MAXPRI      ;MAY NOT EXCEED THIS
011064  101402          BLOS    2$              ;BR IF OK
011066  113703  000000G 1$:     MOVB    VPRIDF,R3       ;IF BAD, SET TO NORMAL PRIORITY
011072  110337  000000G 2$:     MOVB    R3,MXJPRI       ;SET MAX PRIORITY FOR JOB
011076  120361  000000G         CMPB    R3,LBSPRI(R1)   ;COMPARE WITH CURRENT JOB PRIORITY
011102  103013                  BHIS    3$              ;BR IF CURRENT PRIORITY IS OK
011104  110361  000000G         MOVB    R3,LBSPRI(R1)   ;SET BASE PRIORITY FOR JOB
011110  110361  000000G         MOVB    R3,LPRI(R1)     ;SET CURRENT PRIORITY FOR JOB
011114  032761  000000G 000000G BIT     #$VNOTT,LSW(R1) ;Is this a subprocess disconnected from term?
011122  001403                  BEQ     3$              ;Br if not
011124                          OCALL   SPPRED          ;Reduce prio of disconnected subprocess
011132  000137  000000G 3$:     JMP     EMTXIT          ;FINISHED
                ;
                ;  Jump table for special table functions (subfunction code = 4)
                ;
011136  011054' SPLTF:  .WORD   SETMXP           ; 0 -- Set maximum job priority
000000  MXSPTF  =       <.-SPLTF>-2
                ;
                ;  Table of table addresses for positive offsets.
                ;
000000G POSTAB: .WORD   LCONTM          ;+00 -- CONNECT TIME
000000G         .WORD   LSW             ;+01 -- LSW
000000G         .WORD   LSW2            ;+02 -- LSW2
000000G         .WORD   LSW3            ;+03 -- LSW3
000000G         .WORD   LSW4            ;+04 -- LSW4
```

```
     115 011152  000000                      .WORD   LSW5            ;+05 -- LSW5
     116 011154  000000                      .WORD   LSW9            ;+06 -- LSW9
     117 011156  000000                      .WORD   LPROJ           ;+07 -- LPROJ
     118 011160  000000                      .WORD   LPROG           ;+10 -- LPROG
     119 011162  000000                      .WORD   LSW2S           ;+11 -- LSW2S
     120         000024              MXPSTX  =       .-POSTAB
     121                             ;
     122                             ;  Table of addresses for negative offsets.
     123                             ;
     124 011164  000000              NEGTAB: .WORD   0               ;-00
     125 011166  000000                      .WORD   UPPN            ;-01 -- PROJECT #
     126 011170  000000                      .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).
                                ;
011172  010105                  EMUNAM: MOV     R1,R5           ;GET JOB INDEX NUMBER
011174  013700  0000020                MOV     EMTBLK+2,R0     ;GET ADDRESS OF USER'S BUFFER
011200  004737  0000000G               CALL    VALADW          ;VALIDATE THE ADDRESS
011204  123727  0000000G 000002        CMPB    EMTBLK,#2       ;GET OR SET PROGRAM NAME?
011212  002030                        BGE     3$              ;BR IF GET/SET PROGRAM NAME
011214  012702  000006                MOV     #6,R2           ;6 WORDS PER USER NAME
011220  070502                        MUL     R2,R5           ;* 12 BYTES PER USER NAME
011222  062705  0000000G               ADD     #LUNAME,R5      ;POINT TO USER NAME TABLE ENTRY
011226  105737  0000000G               TSTB    EMTBLK          ;IS THIS A REQUEST TO GET OR SET USER NAME
011232  001414                        BEQ     1$              ;BR IF GETTING THE NAME
                                ;
                                ;  This is a request to set the user name.
                                ;  SETNAME privilege is required to do this.
                                ;
011234  032737  0000000G 0000000G       BIT     #PO$NAM,PRIVC0  ;IS THIS JOB PRIVILEGED?
011242  001004                        BNE     2$              ;BR IF YES
011244  012700  000001                MOV     #1,R0           ;RETURN ERROR IF NOT
011250  000137  0000000G               JMP     SETERR
011254  106520                  2$:     MFPD    (R0)+           ;GET NEXT WORD OF USER NAME
011256  012625                        MOV     (SP)+,(R5)+     ;MOVE TO SYSTEM TABLE
011260  077203                        SOB     R2,2$           ;LOOP TO MOVE ALL OF NAME
011262  000432                        BR      9$
                                ;
                                ;  This is a request to get the user name.
                                ;
011264  012546                  1$:     MOV     (R5)+,-(SP)     ;STACK A WORD OF THE USER NAME
011266  106620                        MTPD    (R0)+           ;MOVE TO USER'S BUFFER
011270  077203                        SOB     R2,1$
011272  000426                        BR      9$
                                ;
                                ;  This is a request to get or set the program name
                                ;
011274  003007                  3$:     BGT     4$              ;BR IF SETTING
011276  016146  0000000G               MOV     LPRG1(R1),-(SP) ;GET FIRST 3 CHARS OF PROG NAME
011302  106620                        MTPD    (R0)+           ;MOVE TO USER'S BUFFER
011304  016146  0000000G               MOV     LPRG2(R1),-(SP) ;GET SECOND 3 CHARS OF PROG NAME
011310  106620                        MTPD    (R0)+           ;MOVE TO USER'S BUFFER
011312  000416                        BR      9$
                                ;
                                ;   This is a request to set the program name
                                ;
011314  106520                  4$:     MFPD    (R0)+           ;GET FIRST 3 CHARS
011316  012605                        MOV     (SP)+,R5        ;OF NEW PGM NAME
011320  020527  175000                CMP     R5,#175000      ;IS THIS A VALID RAD50 VALUE
011324  103013                        BHIS    5$              ;BR IF NOT
011326  010561  0000000G               MOV     R5,LPRG1(R1)    ;SET FIRST 3 CHARS OF PROG NAME
011332  106520                        MFPD    (R0)+           ;GET SECOND 3 CHARS
011334  012605                        MOV     (SP)+,R5        ;OF NEW PGM NAME
011336  020527  175000                CMP     R5,#175000      ;IS THIS A VALID RAD50 VALUE
011342  103004                        BHIS    5$              ;BR IF YES
```

Get or set user name

```
     58 011344  010561  0000000                  MOV     R5,LPRG2(R1)    ;SET SECOND 3 CHARS OF PROG NAME
     59 011350  000137  0000000          9$:     JMP     EMTXIT          ;FINISHED
     60
     61 011354  012700  000002          5$:     MOV     #2,R0           ;SIGNAL ERROR CODE
     62 011360  000137  0000000                  JMP     SETERR
```

```
                                        .SBTTL  Request operator privilege
                                ;-------------------------------------------------------------------------
                                ;  Request operator privilege.
                                ;  This is only legal if user has SYSPRV privilege.
                                ;
011364  123727  0000000 000001  REQPRV: CMPB    EMTBLK,#1          ;Request to release locked program?
011372  001417                         BEQ     2$                 ;Br if yes
011374  032737  0000000 0000000        BIT     #PO$SPV,PRIVCO     ;Does job have SETPRV privilege?
011402  001410                         BEQ     1$                 ;Br if not
011404  052737  000000C 0000000        BIS     #<PO$SPV!PO$SYS!PO$NAM!PO$BYP>,PRIVCO ;ENABLE PRIVILEGES
011412  052761  0000000 0000000        BIS     #$PRGLK,LSW5(R1);Lock program to line
011420  000137  0000000                JMP     EMTXIT
011424  005000                 1$:     CLR     R0                 ;RETURN ERROR CODE OF 0
011426  000137  0000000                JMP     SETERR
                                ;
                                ;  Release program lock
                                ;
011432  042761  0000000 0000000 2$:     BIC     #$PRGLK,LSW5(R1);Program no longer locked to line
011440  000137  0000000                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.
                                ;
011444                          GETBRK:
                                ;
                                ;  See if we are connecting a completion routine to a break character
                                ;  or to any input activation condition.
                                ;
011444  105737  0000000                 TSTB    EMTBLK          ;Connecting to break character?
011450  001402                          BEQ     3$              ;Br if yes
011452  000137  0000000                 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.
                                ;
011456  005061  0000000         3$:     CLR     LBRKCH(R1)      ;NO BREAK CHARACTER
011462  005037  0000000                 CLR     UR0             ;ASSUME NO BREAK ROUTINE NOW
011466  016100  0000000                 MOV     LBRKCQ(R1),R0   ;GET ADDRESS OF BREAK COMPLETION QUEUE ENTRY
011472  001410                          BEQ     1$              ;BR IF NONE
011474  016037  0000000 0000000         MOV     CQ$RTN(R0),UR0  ;RETURN ADDRESS OF CURRENT BREAKPOINT ROUTINE
011502  005061  0000000                 CLR     LBRKCQ(R1)      ;SAY NO COMPLETION ENTRY
011506  010001                          MOV     R0,R1           ;GET ADDRESS OF QUEUE ELEMENT TO R1 FOR QFREE
011510  004737  0000000                 CALL    QFREE           ;RELEASE THE QUEUE ELEMENT
                                ;
                                ;  See if we need to establish a completion routine.
                                ;
011514  005737  0000040         1$:     TST     EMTBLK+4        ;DID HE SPECIFY THE ADDRESS OF A COMPL RTN?
011520  001437                          BEQ     2$              ;BR IF NOT
                                ;
                                ;  Get a queue element and set up as a completion queue element.
                                ;
011522  004737  0000000                 CALL    GETQ            ;GET A QUEUE ELEMENT
011526  113702  0000000                 MOVB    CORUSR,R2       ;GET USER INDEX NUMBER
011532  110261  0000000                 MOVB    R2,CQ$JOB(R1)   ;SET JOB NUMBER IN QUEUE ELEMENT
011536  112761  0000000 0000000         MOVB    #S$TWFN,CQ$RNS(R1);SET EXECUTION STATE
011544  116261  0000000 0000000         MOVB    LPRI(R2),CQ$PRI(R1);SET EXECUTION PRIORITY
011552  112761  0000000 0000000         MOVB    #CP$STD,CQ$CP(R1);Set compl routine class priority
011560  013700  0000040                 MOV     EMTBLK+4,R0     ;GET ADDRESS OF COMPL ROUTINE
011564  004737  0000000                 CALL    UACHKW          ;MAKE SURE ADDRESS IS VALID
011570  103415                          BCS     9$              ;BR IF INVALID
011572  010061  0000000                 MOV     R0,CQ$RTN(R1)   ;SET COMPL ROUTINE ADDRESS IN QUEUE ELEMENT
011576  013761  0000000 0000000         MOV     EMTMAP,CQ$PA5(R1);SET EMT ENTRY MAPPING FOR PAR 5
011604  010162  0000000                 MOV     R1,LBRKCQ(R2)   ;SET ADDRESS OF BREAK QUEUE ELEMENT
                                ;
                                ;  Remember special user-defined break character.
                                ;
011610  113700  0000020                 MOVB    EMTBLK+2,R0     ;GET USER-DEFINED BREAK CHARACTER
011614  010062  0000000                 MOV     R0,LBRKCH(R2)   ;SAVE FOR TT INTERRUPT PROCESSING
                                ;
                                ;  Finished
                                ;
011620  000137  0000000         2$:     JMP     EMTXIT
                                ;
                                ;  Error -- Invalid argument address.
```

Establish Break key sentinal control

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

```
                                        .SBTTL  CHKTT -- CHECK I/O TO TT DEVICE
                                ;-------------------------------------------------------------------
                                ;  CHKTT is called to see if the current I/O operation is being directed
                                ;  to device TT or some other device.
                                ;
                                ;  Inputs:
                                ;   CHNADR = Address of current channel block
                                ;
                                ;  Outputs:
                                ;   C-flag cleared if I/O is directed to TT
                                ;   C-flag set if I/O not directed to TT
                                ;
011640  017700  0000000         CHKTT:  MOV     @CHNADR,R0      ;GET DEV TABLE INDEX
011644  002005                          BGE     1$              ;BRANCH IF NOT OPEN
011646  032700  000076                  BIT     #76,R0          ;IS THIS TT DEVICE (INDEX # = 0)?
011652  001002                          BNE     1$              ;BR IF NOT TT
011654  000241                          CLC                     ;SAY THIS IS TT
011656  000207                          RETURN
011660  000261                  1$:     SEC
011662  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.
                ;
011664  010146          CKIOST: MOV     R1,-(SP)
011666  032737  0000000 0000000         BIT     #UMODE,EMTPS    ;Was this EMT executed in kernel mode?
011674  001434                  BEQ     2$              ;Br if yes -- Don't stop system EMT's
011676  105737  0000000         TSTB    CURCP           ;Was EMT executed by a completion routine?
011702  001031                  BNE     2$              ;Br if yes -- I/O hold flag stops cpl rtns
011704  113701  0000000         MOVB    CORUSR,R1       ;Get current job index #
011710  005761  0000000 1$:     TST     LIOHLD(R1)      ;Is I/O hold flag set for job?
011714  001424                  BEQ     2$              ;Br if not
011716                          DISABL                  ;;;** Disable interrupts **
011724  105761  0000000         TSTB    LIOCNT(R1)      ;;;Is any I/O in progress now?
011730  001407                  BEQ     3$              ;;;Br if not
011732  012700  0000000         MOV     #S$IOWT,R0      ;;;Get I/O wait state code
011736  004737  0000000         CALL    QNSPND          ;Enqueue in I/O wait state (enable ints)
011742  004737  0000000         CALL    CHKABT          ;See if we were aborted while asleep
011746  000760                  BR      1$              ;See if we should continue waiting
011750  005061  0000000 3$:     CLR     LIOHLD(R1)      ;;;Reset I/O hold flag
011754                          ENABL                   ;** Enable interrupts **
011762  004737  0000000         CALL    SCHED           ;Call scheduler to probably swap job
                ;
                ;  Finished
                ;
011766  012601          2$:     MOV     (SP)+,R1
011770  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.
                                ;
011772  010246                 SETQ:   MOV     R2,-(SP)
011774  010346                         MOV     R3,-(SP)
011776  010446                         MOV     R4,-(SP)
                                ;
                                ;  Set up misc. parameters
                                ;
012000  013761  0000000 0000000         MOV     CHNNUM,Q.CHAN(R1);SET USER'S CHANNEL #
012006  013703  0000000                 MOV     CHNADR,R3       ;GET ADDRESS OF CSW FOR CHANNEL
012012  010361  0000000                 MOV     R3,Q.UCSW(R1)   ;SET POINTER TO CSW
012016  013761  0000000 0000000         MOV     @#KPAR6,Q.PA6(R1);Save context blk mapping to access channel
012024  016302  0000000                 MOV     C.CSW(R3),R2    ;GET CHANNEL STATUS WORD
012030  042702  177701                 BIC     #^C76,R2        ;EXTRACT DEVICE INDEX NUMBER
012034  110261  0000000                 MOVB    R2,Q.DEVX(R1)   ;SAVE IN QUEUE ELEMENT CELL
012040  116361  0000000 0000000         MOVB    C.DEVQ(R3),Q.UNIT(R1);SET UNIT NUMBER
012046  113702  0000000                 MOVB    CORUSR,R2       ;GET USER INDEX #
012052  006202                         ASR     R2              ;Convert to job number
012054  110261  0000000                 MOVB    R2,Q.JOB(R1)    ;Store job # into queue element
012060  020227  000037                  CMP     R2,#31.         ;Will number fit in old Q.JNUM field?
012064  101402                         BLOS    12$             ;Br if yes
012066  012702  000037                  MOV     #31.,R2         ;Signal that Q.JNUM field not valid
012072  072227  000003          12$:    ASH     #3,R2           ;Position for Q.JNUM field
012076  150261  0000000                 BISB    R2,Q.JNUM(R1)   ;STORE JOB #
                                ;
                                ;  Copy original channel block into internal channel block that is
                                ;  part of I/O queue element.
                                ;
012102  010100                         MOV     R1,R0           ;Get address of I/O queue block
012104  062700  0000000                 ADD     #Q.ICSW,R0      ;Get address of CSW within queue block
012110  010061  0000000                 MOV     R0,Q.CSW(R1)    ;Set pointer to internal CSW
012114  010302                         MOV     R3,R2           ;Get pointer to original channel block
012116  012220                         MOV     (R2)+,(R0)+     ;Copy original channel block to internal one
012120  012220                         MOV     (R2)+,(R0)+
012122  012220                         MOV     (R2)+,(R0)+
012124  012220                         MOV     (R2)+,(R0)+
012126  011210                         MOV     (R2),(R0)
                                ;
                                ;  Set up buffer address
                                ;
012130  013702  0000060                 MOV     EMTBLK+6,R2     ;GET # WORDS TO BE TRANSFERED
012134  001004                         BNE     5$              ;BR IF WORD COUNT SPECIFIED
012136  123727  0000010 000032          CMPB    EMTBLK+1,#32    ;IS THIS A .SPFUN REQUEST?
012144  001104                         BNE     1$              ;BR IF NOT -- THIS MUST BE A SEEK REQUEST
```

```
                                        5$:     MOV     EMTBLK+4,R0     ;BASE ADDRESS OF BUFFER AREA
                                                BIT     #UMODE,EMTPS    ;WAS EMT DONE IN USER OR KERNEL MODE?
                                                BEQ     2$              ;BR IF IN KERNEL MODE (ASSUME ADDRESS IS OK)
                                                CALL    UACHKB          ;MAKE SURE IT IS LEGAL
                                                BCS     9$              ;BR IF ILLEGAL
                                                CMPB    EMTBLK+1,#32    ;IS THIS A .SPFUN OPERATION?
                                                BEQ     2$              ;BR IF YES -- DON'T CHECK WORDCOUNT VALUE
                                                ASL     R2              ;CVT # WORDS TO # BYTES
                                                DEC     R2              ;GET ADDRESS OF LAST BYTE TRANSFERED
                                                ADD     R2,R0
                                                CALL    UACHKB          ;CHECK HIGH RANGE ADDRESS OF BUFFER
                                                BCS     9$              ;BR IF ILLEGAL
                                                MOV     EMTBLK+4,R0     ;GET BACK BASE ADDRESS OF BUFFER
                                        2$:     CALL    CVTUAD          ;CONVERT USER ADDRESS TO PHYSICAL ADDRESS
                                                MOV     R0,Q.BUFF(R1)   ;BUFFER ADDRESS RELATIVE TO Q.PAR
                                                MOV     R2,Q.PAR(R1)    ;RELOCATION BASE ADDRESS
                                        ;
                                        ;  Set up word count and block number for transfer.
                                        ;
                                                CLR     UR0             ;SAY # WORDS TRANSFERED = 0 (IN CASE OF ERROR)
                                                MOV     EMTBLK+6,R4     ;GET WORD COUNT
                                                ADD     #377,R4         ;CONVERT TO BLOCK COUNT
                                                CLRB    R4
                                                SWAB    R4
                                                MOV     EMTBLK+2,R0     ;GET BLOCK # FOR REQUEST
                                                MOV     (R3),R2         ;GET CSW
                                                BIC     #^C76,R2        ;GET DEVICE INDEX
                                                BIT     #DS$DIR,DVSTAT(R2);IS THIS A FILE STRUCTURED DEVICE?
                                                BNE     10$             ;BR IF YES
                                                ADD     R4,R0           ;COMPUTE HIGH BLOCK NUMBER FOR TRANSFER
                                                BIT     #CS$EOF,(R3)    ;HAS END OF FILE BEEN HIT?
                                                BEQ     1$              ;BR IF NOT
                                                BIC     #CS$EOF,(R3)    ;ACKNOWLEDGE EOF
                                                BR      7$              ;RETURN EOF STATUS CODE
                                        10$:    BIC     #CS$EOF,(R3)    ;CLEAR END OF FILE FLAG
                                                TST     C.SBLK(R3)      ;STARTING BLOCK FOR FILE = 0?
                                                BEQ     1$              ;DIR OP IF YES
                                                MOV     C.LENG(R3),R2   ;GET # BLOCKS ALLOCATED FOR FILE
                                                CMP     R0,R2           ;IS THIS A VALID STARTING BLOCK #?
                                                BHIS    8$              ;BR IF INVALID STARTING BLOCK #
                                                ADD     R4,R0           ;COMPUTE ENDING BLOCK NUMBER + 1
                                                CMP     R0,R2           ;WILL TRANSFER PASS END OF FILE?
                                                BLOS    1$              ;BR IF WITHIN FILE
                                                MOV     R2,R0           ;DON'T PASS END OF FILE
                                                SUB     EMTBLK+2,R2     ;CALCULATE # BLOCKS ACTUALLY TO BE TRANSFERRED
                                                SWAB    R2              ;GET ACTUAL WORD COUNT FOR TRANSFER
                                                BR      3$
                                        1$:     MOV     EMTBLK+6,R2     ;GET REQUEST WORD COUNT
                                        3$:     MOV     R2,UR0          ;RETURN ACTUAL WORD COUNT TO USER IN R0
                                                MOV     R2,Q.WCNT(R1)   ;SET WORD COUNT IN Q ELEMENT
                                                MOV     EMTBLK+2,R2     ;GET REQUEST BLOCK #
                                                ADD     C.SBLK(R3),R2   ;ADD BASE BLOCK # OF FILE
                                                MOV     R2,Q.BLKN(R1)   ;SET PHYSICAL BLOCK #
                                        ;
                                        ;  See if this is an I/O operation to a logical disk
                                        ;
                                                CMPB    Q.DEVX(R1),LDDEVX;I/O TO LOGICAL DISK?
```

```
    115 012414  001006                          BNE     11$             ;BR IF NOT
    116 012416  010046                          MOV     R0,-(SP)        ;SAVE HIGH BLOCK NUMBER
    117 012420                                  OCALL   LDIO            ;ALTER QUEUE ELEMENT FOR LOGICAL DISK I/O
    118 012426  012600                          MOV     (SP)+,R0        ;RESTORE HIGH BLOCK NUMBER
    119 012430  103406                          BCS     8$              ;BR IF NOT VALID LOGICAL DISK
    120 012432  000241                  11$:    CLC                     ;SIGNAL SUCCESS ON RETURN
    121 012434  000407                          BR      4$
    122                                 ;  Invalid buffer address
    123 012436  112737  177766  0000000 9$:     MOVB    #-12,INTERR     ;RETURN ERROR CODE 12
    124 012444  000402                          BR      6$
    125                                 ;  Invalid block number (return EOF status)
    126 012446                          8$:
    127                                 ;       BIS     #CS$EOF,(R3)    ;SET END-OF-FILE FLAG IN CSW
    128 012446  105037  0000000         7$:     CLRB    INTERR          ;RETURN ERROR CODE OF 0
    129 012452  000261                  6$:     SEC                     ;SIGNAL ERROR ON RETURN
    130                                 ;
    131                                 ;  Copy Channel Status Word from user's channel status block into
    132                                 ;  internal CSW word in I/O queue element (Q.ICSW).
    133                                 ;  (Note, Do not alter C-flag)
    134                                 ;
    135 012454  016361  0000000 0000000 4$:     MOV     C.CSW(R3),Q.ICSW(R1);Copy CSW into Q element
    136                                 ;
    137                                 ;  Finished
    138                                 ;
    139 012462  012604                          MOV     (SP)+,R4
    140 012464  012603                          MOV     (SP)+,R3
    141 012466  012602                          MOV     (SP)+,R2
    142 012470  000207                          RETURN
```

```
                                        .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.
                                ;
012472  020027  000001          SETCR:  CMP     R0,#1                   ;COMPL RTN ADDR OF 1 ==> .READW/.WRITW
012476  101403                          BLOS    2$                      ;BR IF YES
012500  004737  000000G                 CALL    UACHKW                  ;VALIDATE ADDRESS OF COMPLETION ROUTINE
012504  103406                          BCS     1$                      ;BR IF ILLEGAL
012506  010061  000000G         2$:     MOV     R0,Q.COMP(R1)           ;STORE ADDRESS INTO QUEUE ELEMENT
012512  013761  000000G 000000G         MOV     EMTMAP,Q.PA5(R1)        ;SET EMT ENTRY PAR 5 MAPPING VALUE
012520  000241                          CLC                             ;SIGNAL GOOD RETURN
012522  000207                  1$:     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.
                                ;
012524  010146                  CVTUAD: MOV     R1,-(SP)
012526  010346                          MOV     R3,-(SP)
012530  113701  000000G                 MOVB    CORUSR,R1       ;GET JOB INDEX NUMBER
                                ;
                                ;  Determine if the EMT was done in user or kernel mode.
                                ;
012534  032737  000000G 000000G         BIT     #UMODE,EMTPS    ;WAS EMT DONE IN USER OR KERNEL MODE?
012542  001443                          BEQ     1$              ;BR IF IN KERNEL MODE
                                ;
                                ;  EMT was done in user mode.
                                ;  Map user virtual address to physical address.
                                ;
012544  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON RUNNING?
012552  001405                          BEQ     8$              ;BR IF NOT
012554  020027  000000G                 CMP     R0,#CXTBAS      ;IS KMON DOING I/O TO JOB CONTEXT AREA?
012560  103037                          BHIS    5$              ;BR IF YES
012562  162700  000000G                 SUB     #KMNBAS,R0      ;SUBTRACT KMON ADDRESS BIAS
012566  010003                  8$:     MOV     R0,R3           ;GET VIRTUAL ADDRESS
012570  005002                          CLR     R2              ;CLEAR FOR SHIFT
012572  073227  000003                 ASHC    #3,R2           ;GET VIRTUAL PAR # FROM ADDRESS
012576  006302                          ASL     R2              ;*2 TO GET WORD TABLE INDEX
012600  005762  000000G                 TST     RPDR(R2)        ;IS THIS PAGE SPECIALLY MAPPED?
012604  001405                          BEQ     6$              ;BR IF NOT
012606  016202  000000G                 MOV     RPAR(R2),R2     ;GET BASE 64-BYTE BLOCK # FOR THIS PAGE
012612  042700  160000                 BIC     #160000,R0      ;CLEAR PAR # FROM VIRTUAL ADDRESS
012616  000402                          BR      7$
```

```
     58 012620  016102  0000000         6$:     MOV     LPARBS(R1),R2   ;GET BASE 64-BYTE BLOCK # FOR JOB
     59 012624  010003                 7$:     MOV     R0,R3           ;GET VIRTUAL ADDRESS
     60 012626  000241                         CLC                     ;SHIFT RIGHT 6-BITS WITHOUT SIGN EXTENSION
     61 012630  006003                         ROR     R3
     62 012632  072327  177773               ASH     #-5,R3
     63 012636  060302                       ADD     R3,R2           ;GET 64-BYTE BLOCK # OF PHYSICAL ADDRESS
     64 012640  042700  177700               BIC     #^C77,R0        ;GET BYTE-WITHIN-BLOCK FROM VIRTUAL ADDRESS
     65 012644  062700  0000000              ADD     #VPAR6,R0       ;MAP INTO PAR6 REGION
     66 012650  000440                       BR      9$
     67                                 ;
     68                                 ;  EMT was done in kernel mode.
     69                                 ;  Determine which region to map to.
     70                                 ;
     71 012652  020027  0000000        1$:     CMP     R0,#CXTBAS      ;IS ADDRESS IN JOB CONTEXT BLOCK?
     72 012656  103405                       BLO     2$              ;BR IF NOT
     73                                 ;
     74                                 ;  Map to Job context block.
     75                                 ;
     76 012660  016102  0000000        5$:     MOV     LBASE(R1),R2    ;GET BASE 256-WORD BLOCK # OF CONTEXT BLOCK
     77 012664  072227  000003               ASH     #3,R2           ;CONVERT TO 32-WORD BLOCK #
     78                                 ;  Note: Virtual address in R0 is already in PAR 6 range (140000-157777).
     79 012670  000430                       BR      9$
     80                                 ;
     81                                 ;  See if accessing the system mapped region.
     82                                 ;
     83 012672  020027  0000000        2$:     CMP     R0,#VPAR5       ;IS ADDRESS IN MAPPED SYSTEM REGION?
     84 012676  103405                       BLO     12$             ;BR IF NOT
     85                                 ;
     86                                 ;  Map to the buffer using the current contents of EMTMAP which indicates
     87                                 ;  the mapped value for kernel PAR 5 when the EMT was done.
     88                                 ;
     89 012700  013702  0000000              MOV     EMTMAP,R2       ;Get mapping value that PAR 5 had on entry
     90 012704  062700  020000               ADD     #20000,R0       ;INCREASE VIRTUAL ADDRESS
     91 012710  000420                       BR      9$
     92                                 ;
     93                                 ;  Not in job context area.
     94                                 ;  See if accessing base of job region.
     95                                 ;
     96 012712  005700                 12$:    TST     R0              ;VIRTUAL ADDRESS = 0?
     97 012714  001005                       BNE     3$              ;BR IF NOT
     98                                 ;
     99                                 ;  Map to base of job space.
    100                                 ;
    101 012716  016102  0000000              MOV     LPARBS(R1),R2   ;GET PAR OF JOB BASE
    102 012722  062700  0000000              ADD     #VPAR6,R0       ;PUT VIRTUAL ADDRESS IN PAGE 6 REGION
    103 012726  000411                       BR      9$
    104                                 ;
    105                                 ;  Address is already physical.
    106                                 ;
    107 012730  010002                 3$:     MOV     R0,R2           ;GET ACTUAL ADDRESS
    108 012732  000241                         CLC                     ;CONVERT TO PAGE #
    109 012734  006002                         ROR     R2
    110 012736  072227  177773               ASH     #-5,R2
    111 012742  042700  177700               BIC     #^C77,R0        ;GET BYTE-IN-PAGE TO R0
    112 012746  062700  0000000              ADD     #VPAR6,R0       ;MAP THROUGH PAR 6
    113                                 ;
    114                                 ;  Finished
```

```
    115                                         ;
    116 012752  012603                          9$:     MOV     (SP)+,R3
    117 012754  012601                                  MOV     (SP)+,R1
    118 012756  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
                                ;
012760  010446                  LDIO:   MOV     R4,-(SP)
                                ;
                                ;  Change logical device number and unit number into physical
                                ;  device number and unit number.
                                ;
                                        MOVB    Q.UNIT(R1),R4   ;GET LOGICAL DISK UNIT NUMBER
012762  116104  000000G                 BIC     #^C7,R4         ;GET UNIT # ONLY
012766  042704  177770                  ASL     R4              ;CONVERT TO WORD TABLE INDEX
012772  006304                          MOV     LDPDEV(R4),R0   ;GET PHYSICAL UNIT # AND DEVICE #
012774  016400  000000G                 BEQ     2$              ;BR IF THIS LOGICAL UNIT NOT IN USE
013000  001456                          MOVB    R0,Q.DEVX(R1)   ;SET PHYSICAL DEVICE INDEX NUMBER
013002  110061  000000G                 BICB    #7,Q.UNIT(R1)   ;CLEAR OLD UNIT #
013006  142761  000007  000000G        SWAB    R0              ;GET PHYSICAL UNIT # TO LOW BYTE
013014  000300                          BISB    R0,Q.UNIT(R1)   ;SET PHYSICAL UNIT #
013016  150061  000000G
                                ;
                                ;  See if this is a .SPFUN to get device size or LD unit tables
                                ;
013022  116100  000000G                 MOVB    Q.FUNC(R1),R0   ;IS THIS A .SPFUN REQUEST?
013026  002023                          BGE     1$              ;BR IF NOT
013030  105061  000000G                 CLRB    Q.DEVX(R1)      ;TELL QIO NOT TO DO ANY ACTUAL I/O
013034  120027  000372                  CMPB    R0,#372         ;IS THIS FUNCTION TO GET DEVICE TABLE?
013040  001006                          BNE     5$              ;BR IF NOT
013042  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
013046  000240                          NOP                     ;Reserve room to patch BR back in if needed
013050  004737  013144'                 CALL    LDRST           ;PASS DEVICE TABLE INFO TO USER
013054  000426                          BR      4$              ;FINISHED
013056  120027  000373          5$:     CMPB    R0,#373         ;IS THIS FUNCTION TO GET DEVICE SIZE?
013062  001023                          BNE     4$              ;BR IF NOT -- IGNORE ALL OTHER SPECIAL FUNCTS
013064  016446  000000G                 MOV     LDSIZE(R4),-(SP);GET LOGICAL DISK SIZE
013070  106677  000004G                 MTPD    @EMTBLK+4       ;MOVE INTO USER'S BUFFER
013074  000416                          BR      4$              ;FINISHED
                                ;
                                ;  Check to make sure that the transfer is completely within the
                                ;  area of the logical disk.
                                ;
013076  016100  000000G         1$:     MOV     Q.WCNT(R1),R0   ;GET WORD COUNT FOR TRANSFER
013102  062700  000377                  ADD     #377,R0         ;CONVERT TO BLOCK COUNT
013106  105000                          CLRB    R0
013110  000300                          SWAB    R0
013112  066100  000000G                 ADD     Q.BLKN(R1),R0   ;ADD TRANSFER STARTING BLOCK NUMBER
013116  020064  000000G                 CMP     R0,LDSIZE(R4)   ;IS TRANSFER WITHIN LOGICAL DISK?
013122  101005                          BHI     2$              ;BR IF NOT
```

```
                                        ;
                                        ;  Bias logical block number by base block of logical disk file
                                        ;
013124  066461  0000006 0000006        ADD     LDBASE(R4),Q.BLKN(R1) ;BIAS BLOCK NUMBER
                                        ;
                                        ;  Transfer is ready to go
                                        ;
013132  000241                  4$:     CLC                     ;SIGNAL SUCCESS ON RETURN
013134  000401                          BR      3$
                                        ;
                                        ;  End of file hit by transfer
                                        ;
013136  000261                  2$:     SEC                     ;SIGNAL ERROR ON RETURN
                                        ;
                                        ;  Finished
                                        ;
013140  012604                  3$:     MOV     (SP)+,R4
013142  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.
                                ;
013144  010246                  LDRST:  MOV     R2,-(SP)
013146  010346                          MOV     R3,-(SP)
013150  010446                          MOV     R4,-(SP)
                                ;
                                ;  Pass flag table info to user
                                ;
013152  013702  0000040                 MOV     EMTBLK+4,R2     ;Get address of user's buffer
013156  123727  0000000 000001          CMPB    LDVERS,#1       ;LD version format 5.3& earlier?
013164  001406                          BEQ     11$             ;Skip first two words if old format
013166  012746  045640                  MOV     #^RLD ,-(SP)    ;Pass LD device name to user
013172  106622                          MTPD    (R2)+
013174  012746  0000000                 MOV     #MAXLD,-(SP)    ;Pass number of LD units to user
013200  106622                          MTPD    (R2)+
013202  005003                  11$:    CLR     R3              ;Init LD index
013204  016300  0000000         1$:     MOV     LDPDEV(R3),R0   ;Get real disk device # and unit #
013210  001417                          BEQ     2$              ;Br if unit is not allocated
013212  052700  100000                  BIS     #100000,R0      ;Set unit-allocated flag in status
013216  032763  0000000 0000000         BIT     #LD$RON,LDFLAG(R3);Is the unit read-only?
013224  001411                          BEQ     2$              ;Br if not read-only
013226  123727  0000000 000001          CMPB    LDVERS,#1       ;LD version format 5.3& earlier?
013234  001403                          BEQ     22$             ;Use old mask for old format
013236  052700  040000                  BIS     #040000,R0      ;Set read-only status flag (5.4&later)
013242  000402                          BR      2$
013244  052700  020000          22$:    BIS     #020000,R0      ;Set read-only status flag (5.3&earlier)
013250  010046                  2$:     MOV     R0,-(SP)        ;Pass status flags to user
013252  106622                          MTPD    (R2)+
013254  062703  000002                  ADD     #2,R3           ;Increment unit index
013260  020327  0000000                 CMP     R3,#MAXLD*2     ;Done all units?
013264  103747                          BLO     1$              ;Br if more to do
                                ;
                                ;  Offset info table
                                ;
013266  012704  0000000                 MOV     #LDBASE,R4      ;Point to table with base values for disks
013272  012700  0000000                 MOV     #MAXLD,R0       ;Get number of units
013276  012446                  3$:     MOV     (R4)+,-(SP)     ;Pass offset info to user
013300  106622                          MTPD    (R2)+
013302  077003                          SOB     R0,3$
                                ;
                                ;  Size info table
```

```
                                        ;
013304  012704  0000000                 MOV     #LDSIZE,R4      ;Point to table with size info
013310  012700  0000000                 MOV     #MAXLD,R0       ;Get # LD units
013314  012446                  4$:     MOV     (R4)+,-(SP)     ;Pass size info to user
013316  106622                          MTPD    (R2)+
013320  077003                          SOB     R0,4$
                                        ;
                                        ;  File name table
                                        ;
013322  012704  0000000                 MOV     #LDNAME,R4      ;Point to table with file name info
013326  012700  000000C                 MOV     #MAXLD*4,R0     ;Get # words to move
013332  012446                  5$:     MOV     (R4)+,-(SP)     ;Pass info to user
013334  106622                          MTPD    (R2)+
013336  077003                          SOB     R0,5$
                                        ;
                                        ;  Finished
                                        ;
013340  012604                          MOV     (SP)+,R4
013342  012603                          MOV     (SP)+,R3
013344  012602                          MOV     (SP)+,R2
013346  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.
                                ;
013350  010546                 INDIO:  MOV     R5,-(SP)
                                ;
                                ;  Determine if this I/O is directed to the IND data segment overlay
                                ;
013352  023727  0000000 000017          CMP     CHNNUM,#17      ;IS I/O BEING DONE TO IND.SAV FILE?
013360  001034                          BNE     9$              ;BR IF NOT
013362  023737  0000020 0000000         CMP     EMTBLK+2,INDDBL ;ARE WE ACCESSING DATA SEGMENT OVERLAY?
013370  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.
                                ;
013372  032761  0000000 0000006       BIT     #$INDDW,LSW6(R1);HAS DATA SEGMENT BEEN WRITTEN TO TEMP FILE?
013400  001007                          BNE     1$              ;BR IF YES
013402  123727  0000010 000010          CMPB    EMTBLK+1,#10    ;IS THIS A READ OR WRITE REQUEST?
013410  001420                          BEQ     9$              ;BR IF READ -- GO TO SAV FILE TILL 1ST WRITE
013412  052761  0000000 0000006       BIS     #$INDDW,LSW6(R1);REMEMBER 1ST WRITE HAS BEEN DONE TO TEMP FILE
                                ;
                                ;  Change info to cause I/O to be directed to SY:INDTMP.TSX file.
                                ;  The position within the INDTMP file is a function of the Job index #.
                                ;
013420  010105                  1$:     MOV     R1,R5           ;GET JOB #
013422  006205                          ASR     R5              ;DIVIDE BY 2
013424  005305                          DEC     R5              ;MAKE RELATIVE TO 0
013426  070537  0000000               MUL     INDDBS,R5       ;TIMES # OF BLOCKS IN THE OVERLAY SEGMENT
013432  010537  0000020               MOV     R5,EMTBLK+2     ;REPLACE BLOCK # FOR I/O OPERATION
013436  013703  0000000               MOV     CXTRMN,R3       ;GET ADDRESS OF JOB'S SIMULATED RMON
013442  062703  0000000               ADD     #R$INTC,R3      ;POINT TO CHANNEL BLOCK FOR INDTMP FILE
013446  010337  0000000               MOV     R3,CHNADR       ;DO I/O TO INDTMP FILE (SET NEW CSW ADDRESS)
                                ;
                                ;  Finished
                                ;
013452  012605                  9$:     MOV     (SP)+,R5
013454  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.
                                ;
013456  010246                  EMTTRC: MOV     R2,-(SP)
013460  010346                          MOV     R3,-(SP)
013462  010446                          MOV     R4,-(SP)
013464  010546                          MOV     R5,-(SP)
                                ;
                                ;  Determine if we should trace this EMT
                                ;
013466  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON RUNNING?
013474  001101                          BNE     9$              ;BR IF YES -- DON'T TRACE KMON
013476  032761  000000G 000000G         BIT     #$CCLRN,LSW5(R1);IS CCL RUNNING?
013504  001075                          BNE     9$              ;BR IF YES -- DON'T TRACE CCL
013506  032737  000000G 000000G         BIT     #UMODE,EMTPS    ;WAS EMT EXECUTED IN USER MODE?
013514  001471                          BEQ     9$              ;BR IF NOT -- DON'T TRACE KERNEL EMT'S
                                ;
                                ;  We want to trace this EMT
                                ;  Begin display line
                                ;
013516  012700  000015                  MOV     #15,R0          ;PRINT CR
013522                                  .TTYOUT
013526  012700  000012                  MOV     #12,R0          ;PRINT LF
013532                                  .TTYOUT
                                ;  Print EMT address
013536  013705  000000G                 MOV     EMTADR,R5       ;GET EMT ADDRESS
013542  012703  000005                  MOV     #5,R3           ;PRINT 16-BIT VALUE
013546  004737  013712'                 CALL    PRTOCT
                                ;  Print EMT instruction low-order byte
013552  113705  000000G                 MOVB    CUREMT,R5       ;GET INSTRUCTION ARGUMENT BYTE
013556  012703  000003                  MOV     #3,R3           ;PRINT 8-BITS
013562  004737  013712'                 CALL    PRTOCT          ;PRINT IT
                                ;  Print function code
013566  113705  000001G                 MOVB    EMTBLK+1,R5     ;GET FUNCTION CODE
013572  123727  000000G 000374         CMPB    CUREMT,#374     ;IS THIS AN EMT 374?
013600  001003                          BNE     3$              ;BR IF NOT
013602  162705  000044                  SUB     #E375MX,R5      ;GET 374 FUNCTION CODE
013606  000405                          BR      2$              ;GO PRINT IT
013610  020527  000060          3$:     CMP     R5,#RT11EX      ;IS THIS A TSX EMT?
013614  103402                          BLO     2$              ;BR IF NOT
013616  062705  000020                  ADD     #<100-RT11EX>,R5;GET ORIGINAL FUNCTION CODE VALUE
013622  004737  013712'          2$:     CALL    PRTOCT
                                ;  Print channel number
013626  113705  000000G                 MOVB    EMTBLK,R5       ;GET CHANNEL #
013632  004737  013712'                 CALL    PRTOCT          ;PRINT IT
                                ;  Print EMT argument values
013636  012704  000002G                 MOV     #EMTBLK+2,R4    ;POINT TO ARGUMENT BLOCK
013642  012702  000005                  MOV     #5,R2           ;PRINT 5 ARGUMENT VALUES
013646  010203                          MOV     R2,R3           ;PRINT 16-BITS EACH
013650  012405                  1$:     MOV     (R4)+,R5        ;GET ARGUMENT VALUE
013652  004737  013712'                 CALL    PRTOCT          ;PRINT IT
013656  077204                          SOB     R2,1$           ;LOOP IF MORE TO PRINT
                                ;  End of line
013660  012700  000015                  MOV     #15,R0          ;PRINT CR
```

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

```
                                        .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
                                ;
013712  010346                 PRTOCT: MOV     R3,-(SP)
013714  010446                         MOV     R4,-(SP)
013716  010546                         MOV     R5,-(SP)
                                ;
                                ;  If this is a 16-bit value, print 1st digit now
                                ;
013720  020327  000005                 CMP     R3,#5           ;IS THIS A 16-BIT VALUE?
013724  001007                         BNE     1$              ;BR IF NOT
013726  012700  000030                 MOV     #30,R0          ;GET 60 SHIFTED
013732  006105                         ROL     R5              ;GET 1ST BIT OF VALUE
013734  006100                         ROL     R0              ;INTO R0
013736                                 .TTYOUT                 ;PRINT IT
013742  000404                         BR      2$
                                ;
                                ;  This is an 8-bit value, left justify the 8 bit quantity
                                ;
013744  042705  177400         1$:     BIC     #^C<377>,R5     ;CLEAR ALL BUT 8 BITS
013750  072527  000007                 ASH     #7,R5           ;LEFT JUSTIFY THE VALUE
                                ;
                                ;  Begin loop to print the value
                                ;
013754  012704  000006         2$:     MOV     #6,R4           ;WILL BECOME 60 WHEN SHIFTED
013760  073427  000003                 ASHC    #3,R4           ;MOVE 3 BITS FROM R5 TO R4
013764  010400                         MOV     R4,R0           ;GET TO R0 FOR TTYOUT
013766                                 .TTYOUT                 ;SEND THE CHARACTER
013772  077310                         SOB     R3,2$           ;LOOP IF MORE DIGITS TO PRINT
                                ;
                                ;  Put in 2 spaces following value
                                ;
013774  112700  000040                 MOVB    #' ,R0          ;GET SPACE
014000                                 .TTYOUT                 ;SEND IT
014004                                 .TTYOUT                 ; "    "
                                ;
                                ;  Finished
                                ;
014010  012605                         MOV     (SP)+,R5
014012  012604                         MOV     (SP)+,R4
014014  012603                         MOV     (SP)+,R3
014016  000207                         RETURN
        000001                         .END
Errors detected:  0

*** Assembler statistics


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

PRTOCT -- Print an octal value

Operating system: RT-11

Elapsed time: 00:01:24.40
DK:TSEM2,LP:TSEM2=DK:TSEM2.MAC/C/N:SYM

Cross reference table (CREF V05.04)

```
$CCLRN    1-102     57-16
$CFOPN    1-83      14-21
$CTRLO    1-106      1-125    14-23
$CTRLS    1-84      43-67
$DBGMD    1-119     40-55     40-58
$DEAD     1-126     43-26
$DETCH    1-85      31-34
$DILUP    1-150     31-32
$EMTTR    1-91       4-14
$FORM     1-118      1-161
$HARD     1-126     43-24
$INDDW    1-121     56-33     56-37
$INDRN    1-109      8-75      9-91
$INKMN    1-103      4-36     31-38     53-44     57-14
$IOMAP    1-162     26-84
$KINIT    1-165     37-34
$MLOCK    1-165     37-67
$NOINT    1-132     41-9      41-14
$ODTMD    1-130     40-54     40-57
$PRGLK    1-103     47-11     47-18
$TTGAG    1-136     31-36
$V52EM    1-158     39-11
$VIRJB    1-96      12-90
$VNOTT    1-84      45-98
...V1    15-40      57-25     57-27     57-58     57-60     58-20     58-33     58-39     58-40
...V2    15-40      15-40#
ABORT     1-129
ABRTAD    1-101     36-20*
ABRTCD    1-101     36-21*
ABTIO     1-87       2-58
ADM3A     1-159     39-30
ALCEMT    1-89       2-110
ASTXIT    3-18      34-5#
BADEMT    1-75       2-19      2-23      2-29      2-30      2-39      2-56      4-25      34-12     36-16
C.CSW     1-142      8-7       8-16      8-48      8-53      8-67      8-69*     8-148     8-150*    9-7       9-16      9-51
          9-63       9-65*     9-71      9-73*     9-128     9-130*    9-136     9-138*    11-7      11-74     11-76*    11-79
         11-81*     16-6      16-9      16-36     16-38*    17-9      17-26     17-51*    18-6      18-20     18-27*    19-9
         19-20      19-22*    25-9      25-22     25-38     51-26     51-135
C.DEVQ    1-144     17-32     25-35     51-29
C.LENG    1-145     25-28     51-95
C.NUMQ    1-143     17-31*
C.SBLK    1-145      8-83      8-108    25-25     51-93     51-109
C.USED    1-142      8-104     8-112*    8-113*   25-31
CANMKT    1-73      15-21     23-56     24-9#
CAPOPN    2-85       5-12#
CDFN      2-21      25-52#
CFLAG     1-109      8-121
CHKABT    1-133     21-14     32-24     50-21
CHKERR    7-9        7-18      7-27#
CHKSF     5-6        5-12      5-19      5-26      5-33      5-39      5-45      5-53      5-61#
CHKTT     8-42       9-43     49-13#
CHKUSP    1-90
CHNADR    1-124      4-51*     8-6       9-6       11-6      16-5      17-5      17-25     18-5      19-5      25-5      49-13
          51-23     56-49*
CHNERR    4-40       4-70#
CHNNUM    1-120      4-42*     18-9      51-22     56-25
```

Cross reference table (CREF V05.04)

```
CHNSIZ   1-117    4-49
CHSTAT   2-31    25-5#
CINDAT   1-142   14-10    28-18
CJNFLG   1-70    14-5*
CKINAC   2-83    40-65#
CKIOST   8-5      9-5     11-5     50-8#
CKWIN    2-113    5-75#
CLASN    1-71     2-109
CLKOPN   2-64     5-6#
CLOSE    1-121    2-15     2-53     18-26
CLRDIR   1-107    8-91
CMDB     1-174   30-5
CMDC     1-174   30-6
CMDE     1-174   30-8
CMDF     1-174   30-9
CMDG     1-174   30-10
CMDH     1-174   30-11
CMDI     1-175   30-12
CMDJ     1-175   30-13
CMDK     1-175   30-14
CMDL     1-175   30-15
CMDM     1-175   30-16
CMDN     1-175   30-17
CMDO     1-175   30-18
CMDR     1-176   30-21
CMDS     1-176   30-22
CMDT     1-176   30-23
CMDU     1-176   30-24
CMDW     1-176   30-26
CMDX     1-176   30-27
CMDY     1-176   30-28
CMDZ     1-176   30-29
CMKT     2-27    23-5#
CONFIG   1-163   12-29
CORUSR   1-125    4-13    22-14    24-15    48-36    50-13    51-30    53-34
CP$STD   1-118   48-40
CPYEMT   1-89     2-112
CQ$CP    1-118   48-40*
CQ$FLG   1-108   23-17    24-17
CQ$HOT   1-155   22-20*   23-35
CQ$JOB   1-98    22-14*   23-15    24-15    48-37*
CQ$LNK   1-98    22-36*   23-11    23-13    23-26    23-26*   24-11    24-13    24-19    24-19*
CQ$LOT   1-155   22-22*   23-37
CQ$PA5   1-98    22-29*   48-45*
CQ$PRI   1-128   48-39*
CQ$RO    1-98    22-31*   23-19
CQ$RNS   1-128   48-38*
CQ$RTN   1-98    22-27*   48-23    48-44*
CS$ENT   1-140    8-106   16-9     17-29
CS$EOF   1-141    9-71     9-73     9-136    9-138    11-79    11-81    51-88    51-90    51-92
CS$ERR   1-140    8-67     8-69     8-148    8-150     9-63     9-65     9-128    9-130    11-74    11-76    19-20
        19-22
CS$NMX   1-140    1-141   17-34    18-21
CS$OPN   1-141    8-7      9-7     11-8     16-6     16-36    17-9     17-27    18-6     19-9     25-9
CS$RON   1-109    8-54
CS$SPL   1-147    8-48
```

Cross reference table (CREF V05.04)

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

Cross reference table (CREF V05.04)

```
        51-56   51-58   51-63   51-70   51-78   51-82   51-102  51-105  51-108  54-35   54-45*  55-25
        56-27   56-35   56-46*  57-37   57-47   57-50
EMTCAD   1-124  34-9    34-14*
EMTCAS   1-125  34-10
EMTERR   1-139
EMTLST   2-8#    2-44    2-60    2-117   4-58
EMTMAP   1-120  22-29   48-45   52-18   53-89
EMTMAX   2-117#  4-23
EMTPLS   1-92    6-8
EMTPLX   2-38    6-6#
EMTPS    1-102   4-34    8-89    8-121*  50-9    51-59   53-38   57-18
EMTRAD   1-124  34-13*
EMTSP    1-90
EMTSPD   2-108  43-11#
EMTTRC   4-18   57-7#
EMTWIN   1-92    5-79
EMTXIT   1-74    2-33    2-35    2-54    3-10    3-11    5-55    7-27    8-38    8-156   8-174   9-39
         9-145   10-5    10-11   11-18   11-87   12-18   12-43   12-61   12-94   13-23   14-24   15-47
        16-39   17-46   18-28   19-28   20-21   20-53   21-26   22-42   23-46   23-57   25-45   25-53
        26-27   26-40   27-11   27-24   27-42   29-27   31-76   34-15   35-18   37-106  37-139  37-155
        39-23   39-50   40-47   40-68   41-19   42-32   43-73   44-25   45-77   45-101  46-59   47-12
        47-19   48-55
EMUNAM   2-103  46-8#
ENTER    1-86    2-10
ESPCPS   2-41   35-7#
EXCJOB   1-106  12-75
EXIT     3-12   28-5#
FAKCMP   1-110   8-37    8-170   9-38
FETCH    1-89    3-7
FPUUSE   1-130  12-31*  12-33*
FRECXT   1-83   37-82
GETBRK   2-91   48-6#
GETCXT   1-84   37-72
GETDSS   1-178  43-51
GETPPN   2-98   39-44#
GETQ     1-70    8-95    9-97    11-41   22-9    48-35
GETTIM   2-25   20-5#
GETUCH   1-71   31-61
GFINFO   1-88    2-101
GTJB     2-24   13-5#
GTJJ    13-23#
GTLICN   2-84   40-73#
GTMSBF  31-47   32-9#   32-25
GTSPAC   1-174  30-7
GVAL     2-36   26-5#
GVBAD   27-5    27-7    27-16   27-27   27-33   27-45   27-50#
GVJMP   26-13   27-55#
HAZEL    1-159  39-29
HERR     2-52   10-10#
HITRM   39-16   39-37#
HRESET   3-19   15-6#
INDDBL   1-121  56-27
INDDBS   1-121  56-45
INDFIL   1-177  42-40
INDIO    8-77    9-93   56-21#
INTEMT   2-107  41-4#
```

Cross reference table (CREF V05.04)

```
INTERR    1-110     9-104*    9-151     51-123*   51-128*
INTPRI    1-99      22-38     23-27     23-50     24-20     24-28     32-31     33-25     50-24
IOABFL    1-127     18-24
IOABRT    8-160     9-100     9-105     9-150#    11-91
IOHALT    1-104     15-6
IOWAIT    1-73      8-144     9-124     11-70     16-16     18-8      19-16
JBINFO    2-100     37-15#
JCDB      1-126     8-136     9-80      9-114     16-20     17-40     18-13
JIBASE   37-116#    37-163
JICONT   37-110#    37-162
JICPU    37-133#    37-166
JIDLN     1-164     37-65
JIJMPX    37-53     37-159#   37-169
JIMEM    37-102#    37-161
JIMLOK    1-164     37-69
JINAME   37-148#    37-168
JIPPN    37-127#    37-165
JIPRIO   37-143#    37-167
JIPRIV    1-164     37-81
JIPROG   37-121#    37-164
JIRET1    37-84     37-96     37-98     37-104#   37-112    37-117    37-144
JIRET2    37-123    37-129    37-135#
JIRNND    37-93     38-31#
JIRNST    37-88     38-5#
JIRUN     37-88#    37-160
JISTAT    37-57#    37-159
JIVLN     1-164     37-63
JNOP      12-9      12-18#    12-34
JSWLOC    1-135     12-89*
KMNBAS    1-112     53-48
KMNEMT    1-76       2-86
KPAR6     1-85      51-25
LA120     1-159     39-32
LA36      1-159     39-31
LACTIV    1-99      40-65
LBASE     1-147     37-116    53-76
LBRKCH    1-127     48-19*    48-51*
LBRKCQ    1-127     48-21     48-24*    48-46*
LBSPRI    1-128     45-94     45-96*
LCONTM    1-151     37-111    45-110
LCPUHI    1-166     37-133
LCPULO    1-166     37-134
LD$RON    1-97      55-36
LDBASE    1-97      54-61     55-51
LDDEVX    1-112     11-34     51-114
LDFLAG    1-97      55-36
LDIO     51-117     54-13#
LDMEMT    1-84       2-115
LDNAME    1-104     55-67
LDPDEV    1-97      54-21     55-33
LDRST    54-40      55-19#
LDSIZE    1-97      54-44     54-56     55-59
LDVERS    1-97      55-26     55-38
LEMTPC    1-158     28-5*
LIOCNT    1-131     50-17
LIOHLD    1-131     50-14     50-23*
```

Cross reference table (CREF V05.04)

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

Cross reference table (CREF V05.04)

```
MSGWT     1-93      7-26
MXJADR    1-130    12-77
MXJIFN   37-16     37-169#
MXJPRI    1-128    26-111    45-93*
MXNGTX   45-43     45-127#
MXPSTX   45-50     45-120#
MXSPTF   45-31     45-106#
MXSYFN   42-10     42-41#
MXTSXV   26-32     26-62#
NEGTAB   45-45     45-124#   45-127
NFRESB    1-99     40-46
NLCHN     1-91      4-33
NMUMB     1-137    32-10     32-29*
NSPLDV    1-133    18-10
NUCHN     1-101     4-38     15-35
NUMDCD    1-110     9-82      9-116
NUMDEV    1-154    17-35
ODTBAS    1-107    12-54     12-56     12-92*    13-10     13-21
OVLBIT    1-146    15-38
OVRHC     1-180     7-8       7-17      7-26      8-91      8-164     15-25     15-31     18-12     29-25     37-72     37-78
         37-82     45-100    51-117
PO$BYP    1-95     47-10
PO$DBG    1-91      4-16
PO$MEM    1-131    27-26     27-44
PO$NAM    1-94     46-22     47-10
PO$OPR    1-94     20-27     31-20
PO$SND    1-95     31-16
PO$SPF    1-95     11-32
PO$SPV    1-96     47-8      47-10
PO$SYS    1-94     26-77     37-79     45-19     47-10
P2$MSG    1-93      7-35
P2$RLK    1-92      5-63
P2$TRM    1-94     43-33
PASLIN    1-123    28-9
PEEK     27-15#    27-56
PKGET    27-23#    27-28
PKLOW    27-18     27-20     27-26#
PMEMT     1-77      2-94
PNAME     1-101    25-40
POKE     27-32#    27-58
POLOW    27-35     27-37     27-44#
POSTAB   45-52     45-110#   45-120
POVAL    27-40#    27-46
PRINT     1-90      3-13
PRIVCO    1-102     4-16      11-32     20-27     26-77     27-26     27-44     31-16     31-20     37-76     45-19     46-22
         47-8      47-10*
PRIVC2    1-95      5-63      7-35      43-33
PRTOCT   57-31     57-35     57-45     57-48     57-54     58-9#
PS        1-23#    22-35*    22-38*    23-12*    23-27*    23-50*    24-12*    24-20*    24-28*    32-9*     32-31*    33-13*
         33-25*    50-16*    50-24*
PURGE     2-50     18-5#
PVAL     27-4#     27-57
Q.BLKN    1-143    51-110*   54-55     54-61*
Q.BUFF    1-111    51-72*
Q.CHAN    1-111    51-22*
Q.COMP    1-112    52-17*
```

Cross reference table (CREF V05.04)

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

Cross reference table (CREF V05.04)

```
S$QMIO   1-113    38-19
S$QSPD   1-113    38-21
S$QUSR   1-168    38-16
S$RT     1-114    38-29
S$SFWT   1-167     1-168    38-14
S$SPCB   1-168    38-24
S$SPDB   1-168    38-23
S$SPND   1-167    38-13
S$TMWT   1-153    21-20     38-12
S$TTFN   1-114    38-26
S$TTSC   1-113    38-25
S$TWFN   1-113    38-5      48-38
S$WFM    1-114    38-30
S$WSMB   1-149    32-22     38-22
SB$END   1-136    31-60
SB$LNK   1-137    32-30     33-17     33-19     33-21*    33-24*
SB$PNT   1-137    32-34*
SB$TXT   1-136    31-58     32-33
SCCA     2-37     12-5#
SCHAIN   1-123    28-9
SCHED    1-116    50-25
SDCLOS   1-70     18-12
SDMOVE   1-110     8-164
SDTTM    2-40     20-27#
SERFLG   1-105    10-4*     10-10*
SERR     2-51     10-4#
SETC     1-170    40-67
SETCR    8-126     9-102    11-56     52-13#
SETDSS   1-178    43-56
SETERR   1-74      4-71     5-70      5-78      6-14      7-42      8-10      8-21      8-29      8-57      8-63      8-71
         8-152     9-10     9-21      9-30      9-59      9-67      9-76      9-132     9-141     9-152     11-11     11-25
        11-37     11-78     11-83     16-12     16-41     17-12     17-53     19-12     19-24     20-30     22-48     23-52
        25-12     26-9      26-24     26-35     27-51     29-32     31-19     31-43     31-50     31-81     35-10     35-14
        37-19     37-30     37-37     40-34     42-13     43-29     43-40     45-22     45-82     46-25     46-62     47-14
        48-61
SETMXP  45-88#    45-105
SETPRI   1-88      2-104
SETQ     8-100     9-99     11-50     51-16#
SETRBF   1-174    30-4
SETSIZ   2-97     12-72#
SETSPD   1-100    43-62
SETTOP   3-16     12-49#
SETTP1  12-50#    12-74     12-76
SFCLS    1-142    18-15
SFDATE   1-86      2-42
SFPA     2-32     12-24#
SFPROT   1-86      2-43
SFRSST   1-147    17-42
SFSPND   2-82      5-53#
SFSVST   1-136     5-54     16-22
SFTIME   1-88      2-102
SFWAC    1-175    30-20
SFWL     1-176    30-25
SFWRIT   1-109     8-138
SNDMSG   2-87     31-16#
SNMSHD   1-137    32-14     32-30*
```

Cross reference table (CREF V05.04)

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

Cross reference table (CREF V05.04)

```
UACHKW    1-70      22-17     22-25     23-33     48-42     52-15
UCHAN     1-146     15-46*    25-52*
UCLBLK    1-105     26-126
UCLDAT    1-177     42-39
UFPTRP    1-126     12-26*
ULK1BK    2-75       5-39#
UMODE     1-139      4-34      8-89     50-9      51-59     53-38     57-18
UPDATE   20-11      20-19#
UPPN      1-99      45-125    45-126
URO       1-133      6-13*     8-34*     8-61*     8-119     9-35*     9-57*     9-74*     9-86*     9-139*    12-5*     12-42*
          12-49     12-58*    12-82*    20-17*    25-20*    26-26*    26-39*    27-9*     27-23*    27-40*    39-22*    40-31*
          40-39*    40-46*    40-73*    44-24*    45-61*    48-20*    48-23*    51-77*    51-106*
UTRPAD    1-105     12-17*
VALADB    1-103     12-25     31-56
VALADW    1-103     12-7      12-16     13-6      16-28     17-17     20-6      20-36     21-6      25-18     35-16     37-43
          39-45     42-24     46-10
VIMAGE    1-96      12-87
VLSEMT    1-85       2-114
VMAXMC    1-100      7-33     15-29
VMXSF     1-92       5-61
VMXWIN    1-92       5-75
VPAR5     1-114     53-83
VPAR6     1-99      53-65     53-102    53-112
VPLAS     1-98       6-6
VPRIDF    1-129     45-92
VPRIHI    1-129     26-121
VPRILO    1-129     26-116
VSWPFL    1-129     12-73
VT100     1-159     39-28
VT2007    1-160     39-35
VT2008    1-160     39-36
VT52      1-159     39-13     39-27
VTSLCH    1-127     26-72     44-11
WRITE     1-69       2-17      8-5#
WRITTT    1-108      8-44
WRTCPL    8-168#     9-87
WRTSPL    8-49       8-164#
XGTLN     2-72      40-8#
XGTTAB    2-74      44-8#
XHIIN     1-87       2-77
XHIOUT    1-87       2-76
XHISET    1-88       2-80
XJNOP    40-32      40-40     40-47#    40-56     40-59     40-74
XMSEND    2-68       7-6#
XMSGCK    2-69       7-15#
XMSGWT    2-70       7-24#
XODTMD    2-73      40-52#
XRDTIM    1-87       2-79
XSPLSP    2-71      40-46#
XTERCK    1-87       2-78
```

Cross reference table (CREF V05.04)

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ...CM3 | 15-40 | | | | | | | | | | | |
| ...CM5 | 57-25 | 57-27 | 57-58 | 57-60 | 58-20 | 58-33 | 58-39 | 58-40 | | | | |
| .PURGE | 1-64# | 15-40 | | | | | | | | | | |
| .TTYOU | 1-64# | 57-25 | 57-27 | 57-58 | 57-60 | 58-20 | 58-33 | 58-39 | 58-40 | | | |
| DIE | 1-42# | | | | | | | | | | | |
| DISABL | 1-27# | 22-35 | 23-12 | 24-12 | 32-9 | 33-13 | 50-16 | | | | | |
| ENABL | 1-31# | 22-38 | 23-27 | 23-50 | 24-20 | 24-28 | 32-31 | 33-25 | 50-24 | | | |
| OCALL | 1-53# | 7-8 | 7-17 | 7-26 | 8-91 | 8-164 | 15-25 | 15-31 | 18-12 | 37-72 | 37-78 | 37-82 |
| | 45-100 | 51-117 | | | | | | | | | | |