Table of contents

```
                                        .TITLE  TSEXC2  -- Misc. TSX-Plus Executive Routines
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
                                        .IF     NDF,MAXJOB
                                MAXJOB  =       0               ;Maximum number of primary lines allowed
                                                                ;(0 ==> don't care; >0 for Micro and PRO
                                        .ENDC   ;NDF,MAXJOB
000000                                  .CSECT  TSEXC2
000000  021440                 TSEXC2: .RAD50  /EX2/
                                ;
                                ;-----------------------------------------------------------------------------
                                ;  TSEXC2 is a TSX-Plus virtual system overlay containing misc. routines.
                                ;
                                ;  Copyright (c) 1980, 1981, 1982, 1983, 1984, 1985.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee USA
                                ;
                                ;  All rights reserved
                                ;
                                ;-----------------------------------------------------------------------------
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  LOGOFF, TRPCOM, TSXTX, FPTRPX, EXCINI,NSIP
                                        .GLOBL  SYSDIE, ABORT, STOP, NEWUSR,TRPBPT
                                        .GLOBL  INITLN,CLKRUN,DZSSPD,DLSBRK,DZSBRK
                                        .GLOBL  DLGDSS,DLSDSS,DZGDSS,DZSDSS,DHGDSS,DHSDSS
                                        .GLOBL  DLCLOK,DZCLOK,VHGDSS,VHSDSS,DLSSPD
                                        .GLOBL  DHSSPD,VHSSPD,DHSBRK,VHSBRK,DHCLOK,VHCLOK
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  R$SWPC,C.NUMQ,VUSPHN,$RDSAV,DOTRMP,$GEMAR
                                        .GLOBL  LSW11,$PWKEY,AF$NPW,LNSPAC,SUCF2,IB$SF2,IB$IJ,SPIJ
                                        .GLOBL  LTTCR,EMTBLK,AF$DUP,AF$IND,AF$UCL,AF$SET,LCXTBL
                                        .GLOBL  CINFLG,DIEARG,DIEMSG,DIEPC,DIESP,VSYDMP,DODUMP,SYSHL1
                                        .GLOBL  VPAR5,TRPAR5,VDMKTP,DMPOVL,DMPHND,DMPTXT,PLSINI
                                        .GLOBL  VSWPSL,SWPJOB,SWPPOS,SLTSIZ,CSHDEV,CSHDVN
                                        .GLOBL  $DETCH,LSW,$DILUP,LOTSPC,LOTSIZ,CDSSPD,$DHCDO
                                        .GLOBL  LSTPL,PVON,LNPRIM,LINSWT,LSECPT,MAXSEC,NUMON,NEDCLO
                                        .GLOBL  PO$SPV,PO$NAM,PO$SYS,PO$BYP,LPARNT,CL$EPS
                                        .GLOBL  PO$DBG,SYNAME,PO$MEM,PO$LOK,II$PRV,PO$NFR,PO$NFW
                                        .GLOBL  INSTBL,INSTBN,II$$SZ,II$NAM,II$FLG,II$NPV,AF$PLK
                                        .GLOBL  AF$SCA,AF$NOW,AF$HIE,AF$NOI,AF$IOP,AF$MEM,AF$BYA
                                        .GLOBL  LNMAP,$DISCN,FORCEX,LBASE,LNBLKS,VH$LCR,CLTOTL
                                        .GLOBL  LP$SPD,LP$7BT,LP$PAR,LP$ODD,SYSXIT,VDBFLG,CXBOWN
                                        .GLOBL  DZ$LEN,DZ$8BT,DZ$7BT,DZ$PAR,DZ$ODD,SP$LNK
                                        .GLOBL  HF$LEN,HF$8BT,HF$7BT,HF$PAR,HF$ODD,WINREL
                                        .GLOBL  VF$LEN,VF$8BT,VF$7BT,VF$PAR,VF$EVN
                                        .GLOBL  $AUTO,S9600,$NABRS,SETSPD,LABTIM,PIDPTR,$CTRLO,CSHFIN
                                        .GLOBL  MXTYPE,CDX$DZ,INTMX1,VF$RIE,VF$TIE,VH$CSR,MH$SCR,FP$IOF
                                        .GLOBL  HF$TSR,MH$LPR,MH$BRK,VF$SC,VH$LPR,VF$BC,$NOUCR,JOBCCB
                                        .GLOBL  FRKINI,FQ$$SZ,NUMFRK,FRKGEN,FREFRK,FQ$LNK,SCHED,NPCCB
                                        .GLOBL  LMEMIN,DEQ,LSW2,LSW4,LSW5,$1STLG,LSW6,$CARUP,TOTON,PMUSER
                                        .GLOBL  PMRUN,SS,CORUSR,EXEC,JCDB,LSW7,CANIOT,S150,LIOHLD,NSCP
                                        .GLOBL  UMODE,INTLVL,UTRPAD,CHKABT,UPMODE,UFPTRP,CW$FPU,SCPFHD
```

```
                        .GLOBL  CONFIG,TRRDY,CTTSR,CTTBR,$DBGMD,$NOLF,LCXPAR,$RNMLK
                        .GLOBL  FREIOQ,NUMIOQ,IOQSIZ,Q.LINK,USRINI,LSW9,$SUCF,SYPNCR
                        .GLOBL  LSTMX,MXLNT,LSTPL,LMXNUM,MXLNT,CDCLOK,LCDTYP,LINSIZ
                        .GLOBL  SNMSHD,NMSNMB,SB$$SZ,SB$LNK,CLKINT,$CARMN,LINSPC,SP$$SZ
                        .GLOBL  $START,ILSW2,$DEAD,$PHONE,FSTDL,LSTDL,LSUCF,$TDEAD
                        .GLOBL  INITFL,LSW3,PSW,INTPRI,SYSHLT,CHKUSP,MSGABT
                        .GLOBL  $INKMN,ABRTAD,ABRTCD,STOP,LMXLN,OVRHC,MAPUSR,MONABT
                        .GLOBL  $INIT,LQUAN,LJSW,LINBUF,LINNXT,LSTACT,LINPNT,LINCNT
                        .GLOBL  LOTBUF,LOTNXT,LOTPNT,LACTIV,$LOFCF,CSHALC,NUMCCB,CCBHD
                        .GLOBL  LCOL,LAFSIZ,LPROJ,LPROG,LSCCA,LBRKCQ,LBRKCH,LCPUHI
                        .GLOBL  LCPULO,LCONTM,LINCUR,KPAR6,CURRDB,RPAR,RPDR,CXTRMN
                        .GLOBL  VECBAS,MVWDS,ITRMTP,LTRMTP,MSGINI,VMAXMC,S$HICP,KILJOB
                        .GLOBL  $CTRLS,$VNOTT,SPLINI,NSPLDV,LOKINI,VMXSF,LITIME,QNSPND
                        .GLOBL  SETMAP,DFJMEM,MAXMEM,EMTCAD,EMTRAD,RCBBAS,RCBEND,PLSXIT
                        .GLOBL  SPCPS,JSTKND,JSTK,EMTLEV,UERSEV,$DOOFF,$NOIN,ERRLOC
                        .GLOBL  VSWPFL,KMNTOP,KMNBAS,SUTOP,UHIMEM,JSWLOC,R$CH17
                        .GLOBL  KMNCHN,CSIARE,KMNSTKM,LQUAN,KMNSTR,$CTRLC,LSLEPH,LSLEPL
                        .GLOBL  LSPND,$IOMAP,$MLOCK,LRDTIM,IOHALT,IOSTOP,RTSTOP
                        .GLOBL  USRJOB,FREUSR,FRESPD,CANMKT,QFREE,LCMPL,CQ$LNK
                        .GLOBL  CLSCDB,HF$TIE,HF$RIE,DLSTRT,DZSTRT,CQ$CP
                        .GLOBL  LNSBLK,RCBBAS,RCBEND,FREMEM,VPLAS,TRNSTR,$HARD
                        .GLOBL  CXTPAG,KMNPGS,LIOCNT,VPRIDF,LBSPRI,LPRI,LSTHL
                        .GLOBL  LHIPCT,QHIPRI,INVEC,INRECV,RSR,STPFLG,$VIRJB
                        .GLOBL  RING,TRMRDY,MXRING,MXDTR,VTMOUT,LCDTIM,CARDET
                        .GLOBL  MXCAR,RCVDON,$IITIM,RDINT,RDONE,MXCSR,RIE,NEDCDO,NEDCDI
                        .GLOBL  $XCHAR,LOGCHR,LOGCR,LOGFLG,LF$IN,FPUUSE,$MAPOK,R$MFMV
                        .GLOBL  SPSTAT,SS$RUN,SS$PRT,LPRG1,LPRG2,$NOABT
                        .GLOBL  CURVC,SYSDAT,LSTATE,LSTSL,SWPCOT,DOSCHD,S$INWT,$FPUEX
                        .GLOBL  LRTCHR,MRKTHD,CQ$LNK,S$IOWT,PMBASE,PMTOP,PMNBPC,VPAR6
                        .GLOBL  PMPAR,$SOTFN,S$OTFN,$DEFER,$DODFR,$QCECO,S$OTLO,NEDSOT
                        .GLOBL  CLKRUN,TIKCNT,CLKCNT,LCPULO,LCPUHI,TK1VAL,CQ$JOB
                        .GLOBL  TK1CNT,TK5CNT,SYTIML,SYTIMH,DATIML,DATIMH,JM$LNK
                        .GLOBL  $DHBF1,$DHBF2,LSW10,VF$RIE,HF$RIE,VOFFTM,LOFFTM
                        .GLOBL  VMXWIN,WININI
                        .GLOBL  LMINQ,MINCTR,MINTIM,DTLX,UIOCNT,STRACT,$DEBUG
                        .GLOBL  VPRIHI,VPRILO,VQUAN0,VQUAN3,$SGQ0,$SGQ3,LCLUNT
                        .GLOBL  TMTOTL,TMTOTH,TMIOL,TMIOH,INBSY,OUTBSY,DBGTRP
                        .GLOBL  TMSWPL,TMSWPH,TMUSRL,TMUSRH,TMIOWL,TMIOWH
                        .GLOBL  TMSWTL,TMSWTH,TMIDLL,TMIDLH,SYSHLT,GETMEM,EMTCAS
                        .GLOBL  S$$HIP,S$$RT,S$TMWT,S$TWFN,CQ$PRI,$OITIM
                        .GLOBL  VQUAN1,VQUN1A,VQUAN2,QCPU,MBFFLG,UREGO,S$WSMB
                        .GLOBL  LSLEPL,LSLEPH,ENQTL,QCOMPL,VQUN1B,MONFQH,VMXMON,JM$$SZ
                        .GLOBL  CQ$HOT,CQ$LOT,CQ$LNK,CQ$RNS,CSHINI,FRKPRI
                        .GLOBL  PF$SYS,PF$IOW,PF$OVF,CLKPS,CLKPC,CURCP,SHRRCB,SHRRCN
                        .GLOBL  PMFLGS,UMODE,LEMTPC,$INCOR,EM$DTL,CP$STD
                        .GLOBL  LITIME,VINTIO,VQUN1C,QUNSIG,MS$BRK,TRBRK
                        .GLOBL  LSW8,$SGQ1,$SGQ1A,$SGQ1B,$SGQ1C,$SGQ2
                        .GLOBL  CC$$SZ,CC$LNK,CXTBAS,CXTWDS,PCCCR2,PROFLG
                        .GLOBL  CDGDSS,CDSDSS,MS$DTR,MS$CAR,MS$RNG,PROODC,PLSOFF
                        .GLOBL  CDX$VH,VH$CSR,VH$LSR,VH$LCR,$DBGBK
                        .GLOBL  VF$RNG,VF$DCD,VF$DTR,D.FLAG,D$RUN
                        .GLOBL  MF$LIN,DM$CSR,DM$LSR,MF$RNG,MF$CAR,MF$DTR
                        .GLOBL  FP$CK1,FRKGET,FORKQ,FQ$PRI,FQ$RTN
                        .GLOBL  CDIRTN,CDIFLG,FP$CDI,CDORTN,CDOFLG,FP$CDO
                        .GLOBL  TSR,MXBRK,LMXPRM,MXLPR,MXSBRK,FRECXT
                        .GLOBL  OVRADD,D.ADR,D.PAR,NUMDEV,HANPAR,PNAME
                ;
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Symbolic equates
                                ;
000015                  CR      =       15              ;Carriage-return
000012                  LF      =       12              ;Line-feed
000007                  BELL    =       7               ;Bell
```

```
                        ;+-----------------------------------------------------------------------------
                        ;  Macro calls
                        ;
                              .MCALL  .READW,.PURGE
                        ;
                        ;+-----------------------------------------------------------------------------
                        ;  Macro definitions
                        ;
                        ;  Macro to call a routine in another system overlay.
                        ;
                              .MACRO  OCALL   ENTADD
                              .IF     B, ENTADD
                                      .ERROR  ;OCALL without entry address
                              .ENDC
                              CALL    OVRHC   ;call the low-core overlay handler
                              .WORD   ENTADD
                              .ENDM   OCALL
                        ;
                        ;  Macro to disable interrupts
                        ;
                              .MACRO  DISABL
                              BIS     #340,@#PSW
                              .ENDM   DISABL
                        ;
                        ;  Macro to enable interrrupts
                        ;
                              .MACRO  ENABL
                              BIC     INTPRI,@#PSW
                              .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.
                        ;
                              .MACRO  DIE     MSG,ARG
                              MOV     MSG,@#DIEMSG
                              .IF     NB,ARG
                              MOV     ARG,@#DIEARG
                              .ENDC
                              CALL    @#SYSHLT
                              .ENDM   DIE
                        ;
                        ;  Macro to define a system abort error message
                        ;
                              .MACRO  SATXT   NAME,TEST
                              .GLOBL  EM$'NAME
                        EM$'NAME = . - DIEBAS
                              .ASCIZ  \'NAME'-'TEST'\
                              .ENDM   SATXT
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Fatal system abort error messages:
                                ;
                                        .NLIST  BEX
000002                          DIEBAS:
000002                          SATXT   DTL,<Demonstration system time limit reached>
000056                          SATXT   FRK,<No free FORK blocks>
000106                          SATXT   JMO,<Jump occurred to location 0>
000146                          SATXT   KRE,<KMON read error>
000172                          SATXT   KTP,<Kernel mode trap>
000217                          SATXT   LMF,<Job lock mem failure>
000250                          SATXT   MIO,<Need to increase value of MIONWB sysgen parameter>
000336                          SATXT   MPR,<Memory parity error>
000366                          SATXT   NQE,<Ran out of free I/O queue elements>
000435                          SATXT   NSP,<No free swap command packets>
000476                          SATXT   PFT,<Power-fail trap>
000522                          SATXT   RIT,<Trap in real-time interrupt service routine>
000602                          SATXT   SFO,<Job swap file overflow>
000635                          SATXT   SIE,<Swap file I/O error>
000665                          SATXT   SSE,<PLAS region swap file I/O error>
000731                          SATXT   SJN,<Job # 0 at STOP>
000755                          SATXT   UEI,<Interrupt occurred at unexpected location>
001033                          SATXT   SOF,<Stack overflow>
                                ;
                                ;  Other related text strings.
                                ;
001056      015     012     012 TXFSE:  .ASCII  <CR><LF><LF><BELL>/?TSX-F-Fatal system error at /<200>
001120      101     162     147 TXARG:  .ASCII  /Arg. value = /<200>
001136      000                 TXNUL:  .BYTE   0
001137      120     101     122 TXPAR5: .ASCII  /PAR5 value = /<200>
001155      123     145     147 TXSEG:  .ASCII  /Seg. value = /<200>
001173      117     166     145 TXOID:  .ASCII  /Overlay: /<200>
001205      104     145     166 TXDEV:  .ASCII  /Device name: /<200>
001223      123     120     040 SPTXT:  .ASCII  /SP at time of crash = /<200>
                                        .EVEN
                                ;
                                ;  Line select bits for a DH11 mux.
                                ;
001252  000001  000002  000004 DHLBIT: .WORD   1,2,4,10,20,40,100,200,400
001274  001000  002000  004000         .WORD   1000,2000,4000,10000,20000,40000,100000
                                ;
                                ;  Line select bits for DZ11 mux.
                                ;
001312      001     002     004 MXLBIT: .BYTE   1,2,4,10,20,40,100,200
                                ;
                                ;  Number of days in each month
                                ;
001322      037     034     037 MONDAY: .BYTE   31.,28.,31.,30.,31.,30.,31.,31.,30.,31.,30.,31.
                                        .EVEN
001336  000000                  FORKIT: .WORD   0                ;Flag to create fork process
001340  0000000                 PROTIM: .WORD   PROODC           ;Call PI driver after this many ticks
001342  177777                  TIKO1S: .WORD   -1               ;# pending 0.1 second clock ticks
001344  063337                  R50PRO: .RAD50  /PRO/
001346  063344                  R50PRT: .RAD50  /PRT/
001350  073376                  R50SAV: .RAD50  /SAV/
                                ;
                                ;  Table to convert normal TSX-Plus speed codes into DH11 speed codes
```

```
                                        ;
001352      001     002     003 DHSPCT: .BYTE   1,2,3,4,5,7,10,11,12,0,13,0,14,0,15,16
                                        ;
                                        ;  Table to convert normal TSX-Plus speed codes into DHV11 speed codes
                                        ;
001372      000     001     002 VHSPCT: .BYTE   0,1,2,3,4,5,6,7,10,11,12,0,13,14,15,16
                                        .EVEN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Table of programs that have automatic switch assignment when they
                                ;  are started by TSKMON.  The following flags may be specified following
                                ;  the program name words:
                                ;
                                ;       AF$NOW = Allow non-wait .TTYIN operation.
                                ;       AF$HIE = Run program in high-efficiency mode.
                                ;       AF$NOI = Run program in non-interactive mode.
                                ;       AF$IOP = Map user PAR 7 to I/O page (requires operator priv.)
                                ;       AF$SCA = Enable single character activation.
                                ;       AF$MEM = Lock program in low memory.
                                ;       AF$PLK = RUN/LOCK program
                                ;       AF$DBG = RUN/DEBUG program
                                ;       AF$BYA = Bypass user ASSIGNs
                                ;       AF$TPO = Use transparent terminal output
                                ;       AF$DUP = Program is DUP
                                ;       AF$IND = Program is IND
                                ;       AF$UCL = Program is TSXUCL
                                ;       AF$SET = Program is SETUP
                                ;       AF$CCA = Suppress ctrl-C abort
                                ;       AF$NPW = No windows during program
                                ;
                                ;  Each program entry must consist of two words containing the 6 character
                                ;  program name followed by a word with the flags.
                                ;
000000                          NSIP    =       0                       ;No system installed programs yet
001412                          SRFPR@:
001412  016130  000000                  .RAD50  /DUP    /
001416  000000                          .WORD   AF$DUP
001420  000000                          .WORD   0
000001                  NSIP    =       NSIP+1
001422  035164  000000                  .RAD50  /IND    /
001426  000000C                         .WORD   AF$NOW!AF$IND
001430  000000                          .WORD   0
000002                  NSIP    =       NSIP+1                  ;Count another system program
001432  042614  000000                  .RAD50  /KED    /
001436  000000C                         .WORD   AF$SCA!AF$HIE!AF$NOW
001440  000000                          .WORD   0
000003                  NSIP    =       NSIP+1                  ;Count another system program
001442  042640  000000                  .RAD50  /KEX    /
001446  000000C                         .WORD   AF$SCA!AF$HIE!AF$NOW
001450  000000                          .WORD   0
000004                  NSIP    =       NSIP+1                  ;Count another system program
001452  045130  000000                  .RAD50  /K52    /
001456  000000C                         .WORD   AF$SCA!AF$HIE!AF$NOW
001460  000000                          .WORD   0
000005                  NSIP    =       NSIP+1                  ;Count another system program
001462  046537  057760                  .RAD50  /LOGON /
001466  000000C                         .WORD   AF$PLK!AF$BYA
001470  000001  000000C                 .WORD   +1,PO$SPV!PO$NAM!PO$SYS!PO$BYP
001474  000000                          .WORD   0
000006                  NSIP    =       NSIP+1
001476  062074  012000                  .RAD50  /PATCH /
001502  0000006                         .WORD   AF$SCA
001504  000000                          .WORD   0
000007                  NSIP    =       NSIP+1                  ;Count another system program
001506  073634  102700                  .RAD50  /SETUP /
```

```
     58 001512  000000C                        .WORD   AF$IOP!AF$SET
     59 001514  000000                         .WORD   0
     60         000010              NSIP       =       NSIP+1                  ;Count another system program
     61 001516  075273  051646                .RAD50  /SYSMON/
     62 001522  000000                         .WORD   0
     63 001524  000001  0000006                .WORD   +1,PO$MEM
     64 001530  000000                         .WORD   0
     65         000011              NSIP       =       NSIP+1                  ;Count another system program
     66 001532  076713  056700                .RAD50  /TECO  /
     67 001536  000000C                        .WORD   AF$SCA!AF$NOW
     68 001540  000000                         .WORD   0
     69         000012              NSIP       =       NSIP+1                  ;Count another system program
     70 001542  077721  055176                .RAD50  /TRANSF/
     71 001546  000000C                        .WORD   AF$SCA!AF$NOI!AF$NOW!AF$NPW
     72 001550  000000                         .WORD   0
     73         000013              NSIP       =       NSIP+1
     74 001552  077771  103150                .RAD50  /TSAUTH/
     75 001556  0000008                        .WORD   AF$BYA
     76 001560  000000                         .WORD   0
     77         000014              NSIP       =       NSIP+1                  ;Count another system program
     78 001562  100020  101704                .RAD50  /TSXUCL/
     79 001566  0000008                        .WORD   AF$UCL
     80 001570  000000                         .WORD   0
     81         000015              NSIP       =       NSIP+1
     82 001572  106243  057710                .RAD50  /VTCOM /
     83 001576  0000006                        .WORD   AF$SCA!AF$NOW!AF$MEM
     84 001600  000001  0000000               .WORD   +1,PO$LOK
     85 001604  000000                         .WORD   0
     86         000016              NSIP       =       NSIP+1                  ;Count another system program
     87 001606                      SRFEND:                                    ;End of special program flag list
```

```
                                        .SBTTL  INITLN -- Initialize a line
                                ;-----------------------------------------------------------------------
                                ;  INITLN is called to initiate (logon) a line.
                                ;  It initializes a number of line control tables and then places
                                ;  the line in a high-priority execution state.
                                ;  If the system is generated with job swapping turned off (SWAPFL=0)
                                ;  a check is made to see if there is sufficient free memory available
                                ;  for the job before it is initiated.  If there is not enough free memory
                                ;  available, the job is not initiated.
                                ;
                                ;  Inputs:
                                ;   R1 = Number of line to be initiated
                                ;   R0 = Pointer to I/O queue element with name of secondary
                                ;        start-up command file for the job (0=none).
                                ;
                                ;  Outputs:
                                ;   C-flag set if swapping is disabled and there is insufficient free
                                ;   memory space available to start job.
                                ;   The queue element with the secondary start-up command file name is
                                ;   freed if the job cannot be started.  Otherwise it is freed after the
                                ;   job is initialized.
                                ;
001606  010246                  INITLN: MOV     R2,-(SP)
001610  010346                          MOV     R3,-(SP)
001612  010003                          MOV     R0,R3           ;Get pointer to Q element with CF name
                                ;
                                ;  Never start a line that is being used as a CL unit
                                ;
001614  005761  0000000                 TST     LCLUNT(R1)      ;Is this a CL line?
001620  002160                          BGE     9$              ;Br if yes
                                ;
                                ;  See if we are constrained by max # jobs allowed to be on
                                ;
001622  012700  000000                  MOV     #MAXJOB,R0      ;Get max # jobs allowed to be on
001626  001403                          BEQ     4$              ;Br if no limit on # logged on jobs
001630  123700  0000000                 CMPB    NUMON,R0        ;Max # jobs already logged on?
001634  103152                          BHIS    9$              ;Br if yes
                                ;
                                ;  Initialize some line control tables
                                ;
001636  005061  0000000         4$:     CLR     LSW(R1)         ;Reset job status flags
001642  005061  0000000                 CLR     LNBLKS(R1)      ;JOB HAS NO ASSIGNED MEMORY NOW
001646  005061  0000000                 CLR     LIOCNT(R1)      ;JOB HAS NO ACTIVE I/O
001652  005061  0000000                 CLR     LCMPL(R1)       ;JOB HAS NOT PENDING COMPLETION ROUTINES
                                ;
                                ;  Determine how much memory the line needs to be initiated.
                                ;
001656  013700  0000000                 MOV     DFJMEM,R0       ;GET DEFAULT # KB FOR JOB
001662  006300                          ASL     R0              ;CVT TO # PAGES
001664  063700  0000000                 ADD     CXTPAG,R0       ;ADD # PAGES NEEDED FOR JOB CONTEXT BLOCK
001670  020037  0000000                 CMP     R0,KMNPGS       ;COMPARE TO # PAGES NEEDED TO RUN KMON
001674  103002                          BHIS    1$              ;BR IF JOB SPACE > KMON SIZE
001676  013700  0000000                 MOV     KMNPGS,R0       ;MUST HAVE AT LEAST ENOUGH MEMORY FOR KMON
                                ;
                                ;  If this is a non-swapping system, make sure enough free pages are
                                ;  available for this job.
                                ;
```

```
      58 001702  010061  0000000          1$:     MOV     R0,LMEMIN(R1)   ;SET # MEMORY PAGES NEEDED FOR JOB
      59 001706  105737  0000000                  TSTB    VSWPFL          ;IS THIS A SWAPPING SYSTEM?
      60 001712  001010                          BNE     3$              ;BR IF YES
      61 001714  010061  0000000                  MOV     R0,LNBLKS(R1)   ;SET SIZE OF ROOT OF JOB REGION
      62 001720  004737  0000000                  CALL    GETMEM          ;TRY TO ALLOCATE MEMORY FOR THIS JOB
      63 001724  103516                          BCS     9$              ;BR IF NOT ENOUGH MEMORY AVAILABLE
      64 001726  052761  0000000 0000000        BIS     #$INCOR,LSW(R1) ;SET FLAG SAYING JOB IS IN MEMORY
      65                                ;
      66                                ;  Initialize the line control tables for the job
      67                                ;
      68 001734  016102  0000000          3$:     MOV     LINBUF(R1),R2   ;SET UP INFO ABOUT INPUT BUFFER
      69 001740  010261  0000000                  MOV     R2,LINNXT(R1)
      70 001744  010261  0000000                  MOV     R2,LSTACT(R1)
      71 001750  010261  0000000                  MOV     R2,LINPNT(R1)
      72 001754  016161  0000000 0000000        MOV     LINSIZ(R1),LINSPC(R1)
      73 001762  005061  0000000                  CLR     LINCNT(R1)
      74 001766  052761  0000000 0000000        BIS     #$DILUP,LSW(R1) ;REMEMBER LINE IS ACTIVE
      75 001774  052761  0000000 0000000        BIS     #$NOIN,LSW3(R1) ;DON'T ACCEPT TT CHARS FOR LINE YET
      76 002002  012761  0000000 0000000        MOV     #$SUCF,LSW9(R1) ;Say we are in startup command file
      77 002010  012761  0000000 0000000        MOV     #$PWKEY,LSW11(R1);Initialize LSW11
      78 002016  113761  0000000 0000000        MOVB    VPRIDF,LBSPRI(R1);SET BASE PRIORITY FOR JOB
      79 002024  113761  0000000 0000000        MOVB    VPRIDF,LPRI(R1);SET CURRENT PRIORITY FOR JOB
      80 002032  013761  0000000 0000000        MOV     VINTIO,LHIPCT(R1);INIT HIGH-PRIORITY HITS FOR JOB
      81 002040  013761  0000000 0000000        MOV     VQUAN1,LITIME(R1);SET INTERACTIVE JOB TIME SLICE
      82 002046  010361  0000000                  MOV     R3,LPROJ(R1)    ;Store ptr to command file buffer in LPROJ
      83                                ;
      84                                ;  If this is a dial-up line and carrier is up, set flag that will
      85                                ;  cause us to log off the line if carrier is lost.
      86                                ;
      87 002052  020127  0000000                  CMP     R1,#LSTPL       ;Is this a primary line?
      88 002056  101020                          BHI     2$              ;Br if not
      89 002060  005061  0000000                  CLR     LOFFTM(R1)      ;Don't have to time logoff time any more
      90 002064  042761  0000000 0000000        BIC     #$CARMN,LSW5(R1);Assume we do not need to monitor carrier
      91 002072  032761  0000000 0000000        BIT     #$PHONE,ILSW2(R1);Is this a dial-up line?
      92 002100  001407                          BEQ     2$              ;Br if not
      93 002102  032761  0000000 0000000        BIT     #$CARUP,LSW3(R1);Is carrier up now?
      94 002110  001403                          BEQ     2$              ;Br if not
      95 002112  052761  0000000 0000000        BIS     #$CARMN,LSW5(R1);Set flag saying to monitor carrier
      96                                ;
      97                                ;  Put job in high priority execution state
      98                                ;
      99 002120  004737  0000000          2$:     CALL    QHIPRI          ;PUT JOB IN HIGH PRIORITY STATE
     100                                ;
     101                                ;  Count number of logged on jobs
     102                                ;
     103 002124  105237  0000000                  INCB    TOTON           ;Total number of logged on jobs
     104 002130  020127  0000000                  CMP     R1,#LSTPL       ;Is this a primary line?
     105 002134  101003                          BHI     5$              ;Br if not
     106 002136  105237  0000000                  INCB    NUMON           ;Count another real line on
     107 002142  000403                          BR      6$
     108 002144  020127  0000000          5$:     CMP     R1,#LSTDL       ;Is this a virtual line?
     109 002150  101402                          BLOS    8$              ;Br if not
     110 002152  105237  0000000          6$:     INCB    PVON            ;Count # primary & virtual lines on
     111                                ;
     112                                ;  Successful finish
     113                                ;
     114 002156  000241                  8$:     CLC                     ;Signal successful return
```

```
002160  000407                          BR      10$
                                ;
                                ;  We were not able to start the job
                                ;  Free the I/O queue element used to pass name of start-up command file
                                ;
002162  010146                  9$:     MOV     R1,-(SP)        ;Save job number
002164  010301                          MOV     R3,R1           ;Get pointer to Q element
002166  001402                          BEQ     11$             ;Br if no Q element to free
002170  004737  0000000                 CALL    QFREE           ;Free the Q element
002174  012601                  11$:    MOV     (SP)+,R1        ;Restore job number
002176  000261                          SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
002200  012603                  10$:    MOV     (SP)+,R3
002202  012602                          MOV     (SP)+,R2
002204  000207                          RETURN
```

```
                                        .SBTTL  NEWUSR -- Start a new time-sharing job
                                ;-----------------------------------------------------------------------
                                ;  Do initialization for start-up of a new job.
                                ;
002206                          NEWUSR:
                                ;
                                ;  Initialize LSW tables.
                                ;
002206  052761  000000G 000000G        BIS     #$INIT,LSW(R1)  ;SAY JOB INITIALIZATION IS BEING DONE
002214  005061  000000G                CLR     LQUAN(R1)       ;INITIALIZE JOB'S TIME QUANTUM
002220  005061  000000G                CLR     LJSW(R1)        ;JOB STATUS WORD
002224  005061  000000G                CLR     LSW7(R1)
002230  005061  000000G                CLR     LSW8(R1)
002234  005061  000000G                CLR     LNSBLK(R1)      ;NO SPACE FOR ANY PLAS REGIONS
002240  012761  000000G 000000G        MOV     #$INKMN,LSW4(R1);START OUT RUNNING KMON
002246  016102  000000G                MOV     LOTBUF(R1),R2   ;SET UP INFO ABOUT OUTPUT BUFFER
002252  010261  000000G                MOV     R2,LOTNXT(R1)
002256  010261  000000G                MOV     R2,LOTPNT(R1)
002262  016161  000000G 000000G        MOV     LOTSIZ(R1),LOTSPC(R1)
002270  005061  000000G                CLR     LACTIV(R1)      ;SAY NO ACTIVATION CHARS RECEIVED YET
002274  005061  000000G                CLR     LCOL(R1)
002300  005061  000000G                CLR     LAFSIZ(R1)      ;NO ACTIVATION FIELD SIZE
002304  005061  000000G                CLR     LPROG(R1)       ;OR PROGRAMMER NUMBER
002310  005061  000000G                CLR     LSCCA(R1)       ;NO .SCCA DONE YET
002314  005061  000000G                CLR     LBRKCQ(R1)      ;NO BREAK KEY CONNECTION
002320  005061  000000G                CLR     LBRKCH(R1)      ;CLEAR BREAK CHARACTER
002324  005061  000000G                CLR     LTTCR(R1)       ;No TT activation completion routine
002330  005061  000000G                CLR     LCPUHI(R1)      ;CLEAR RUN-TIME ACCUMULATOR
002334  005061  000000G                CLR     LCPULO(R1)
002340  005061  000000G                CLR     LCONTM(R1)      ;CLEAR CONNTECT-TIME
002344  005061  000000G                CLR     LINCUR(R1)
                                ;
                                ;  Map kernel mode PAR 6 to job context block.
                                ;
002350  016137  000000G 000000G        MOV     LCXPAR(R1),@#KPAR6;MAP KERNEL PAGE 6 TO CONTEXT BLOCK FOR JOB
                                ;
                                ;  Zero the job's context block
                                ;
002356  012702  000000G                MOV     #CXTBAS,R2      ;Get address of base of context area
002362  013703  000000G                MOV     CXTWDS,R3       ;Get # words in context area
002366  006203                        ASR     R3              ;Get # of doublewords
002370  005022                  7$:   CLR     (R2)+           ;Clear first word of pair
002372  005022                        CLR     (R2)+           ;Clear second word of pair
002374  077303                        SOB     R3,7$           ;Loop if more doublewords left to clear
002376  032737  000001  000000G        BIT     #1,CXTWDS       ;Is there an odd word at end to clear?
002404  001401                        BEQ     8$              ;Br if not
002406  005012                        CLR     (R2)            ;Clear last word of context block
                                ;
                                ;  Initialize some cells in job context block
                                ;
002410  012737  177777  000000G 8$:   MOV     #-1,EMTLEV      ;Say job is not executing an EMT
002416  012737  000000G 000000G        MOV     #EMTCAS,EMTCAD  ;Say completion routine return stack is empty
002424  012737  123456  000000G        MOV     #123456,JSTKND  ;Set value used to check for stack overflow
                                ;
                                ;  Set up simulated RMON fixed offset table.
                                ;
002432  013702  000000G                MOV     CXTRMN,R2       ;Get addr of RMON in job context area
```

```
     58 002436  012703  0000000                    MOV     #VECBAS,R3      ;POINT TO SYSTEM CHANNEL BLOCK
     59 002442  012700  0000000                    MOV     #MVWDS,R0       ;GET # WORDS TO MOVE
     60 002446  012322                     4$:     MOV     (R3)+,(R2)+     ;SET UP JOB'S SIMULATED RMON IN CONTEXT BLK
     61 002450  077002                             SOB     R0,4$
     62                                     ;
     63                                     ;  Zero the I/O count in the channel used to access system swap file.
     64                                     ;  (Count could be non-zero if swap was in progress when we copied RMON data)
     65                                     ;
     66 002452  013700  0000000                    MOV     CXTRMN,R0       ;Get pointer to base of simulated RMON
     67 002456  062700  0000000                    ADD     #R$SWPC,R0      ;Point to swap file channel in context block
     68 002462  105060  0000000                    CLRB    C.NUMQ(R0)      ;Zero I/O count in swap file channel block
     69                                     ;
     70                                     ;  Change instruction in .MFPS routine in simulated RMON to return a
     71                                     ;  value of 0 rather than trying to access the I/O page
     72                                     ;
     73 002466  013700  0000000                    MOV     CXTRMN,R0       ;Get pointer to base of simulated RMON
     74 002472  062700  0000000                    ADD     #R$MFMV,R0      ;Point to MOV @#PSW,(SP) instruction
     75 002476  012720  012716                     MOV     #012716,(R0)+   ;Store MOV #0,(SP) instruction
     76 002502  005010                             CLR     (R0)            ;Store 0 value following instruction
     77                                     ;
     78                                     ;  See if we need to set up the name of a secondary start-up command
     79                                     ;  file for the job or other info provided by initiating job.
     80                                     ;
     81 002504  016104  0000000                    MOV     LPROJ(R1),R4    ;Get pointer to buffer with file name
     82 002510  001421                             BEQ     10$             ;Br if no secondary start-up file
     83 002512  116437  0000000 0000000            MOVB    IB$IJ(R4),SPIJ  ;Save index # of initiating job
     84 002520  010402                             MOV     R4,R2
     85 002522  062702  0000000                    ADD     #IB$SF2,R2      ;Point to start of name string
     86 002526  012703  0000000                    MOV     #SUCF2,R3       ;Point to cell in context block
     87 002532  112223                     11$:    MOVB    (R2)+,(R3)+     ;Move name into context block
     88 002534  001376                             BNE     11$             ;Loop till asciz null reached
     89 002536  010146                             MOV     R1,-(SP)        ;Save job number
     90 002540  010401                             MOV     R4,R1           ;Get pointer to Q element used as buffer
     91 002542  004737  0000000                    CALL    QFREE           ;Free the Q element
     92 002546  012601                             MOV     (SP)+,R1        ;Restore job number
     93 002550  005061  0000000                    CLR     LPROJ(R1)       ;Say no project number
     94                                     ;
     95                                     ;  Determine if this is a primary, virtual, or detached line
     96                                     ;
     97 002554  020127  0000000            10$:    CMP     R1,#LSTDL       ;REAL OR VIRTUAL LINE?
     98 002560  003021                             BGT     1$              ;BR IF VIRTUAL
     99 002562  020127  0000000                    CMP     R1,#LSTPL       ;REAL OR DETACHED LINE?
    100 002566  003404                             BLE     2$              ;BR IF REAL
    101                                     ;
    102                                     ;  Initialization for detached jobs only.
    103                                     ;
    104 002570  052761  0000000 0000000            BIS     #$DETCH,LSW(R1) ;REMEMBER THIS IS A DETACHED JOB
    105 002576  000430                             BR      3$
    106                                     ;
    107                                     ;  Initialization for primary lines only.
    108                                     ;
    109 002600  016161  0000000 0000000    2$:     MOV     ILSW2(R1),LSW2(R1);INIT SOME LSW TABLES
    110 002606  016161  0000000 0000000            MOV     ITRMTP(R1),LTRMTP(R1);SET DEFAULT TERMINAL TYPE
    111 002614  042761  0000000 0000000            BIC     #$CTRLS!$CTRLO,LSW3(R1);Enable terminal output
    112 002622  000416                             BR      3$
    113                                     ;
    114                                     ;  Initialization for virtual lines only.
```

```
                                        ;  Copy some information from parent line.
                                        ;
002624  016102  0000006          1$:    MOV     LNPRIM(R1),R2   ;GET PRIMARY LINE #
002630  016261  0000006 0000006        MOV     LSW2(R2),LSW2(R1)
002636  016261  0000006 0000006        MOV     LTRMTP(R2),LTRMTP(R1)
002644  016261  0000006 0000006        MOV     LPROJ(R2),LPROJ(R1)
002652  016261  0000006 0000006        MOV     LPROG(R2),LPROG(R1)
                                        ;
                                        ;  See if we should start line with deferred echo
                                        ;
002660  032761  0000006 0000006  3$:    BIT     #$$DEFER,LSW2(R1);Is deferred echo wanted?
002666  001403                        BEQ     6$                ;Br if not
002670  052761  000000C 0000006        BIS     #<$DODFR!$GCECO>,LSW3(R1) ;Start deferring now
                                        ;
                                        ;  Set up mapping registers for job.
                                        ;
002676  004737  0000006          6$:    CALL    SETMAP            ;SET UP MAPPING REGISTERS FOR THE JOB
002702  013700  0000006                 MOV     DFJMEM,R0         ;GET DEFAULT # K-BYTES OF MEMORY FOR JOB
002706  072027  000012                  ASH     #10.,R0           ;CONVERT TO ADDRESS
002712  010037  0000006                 MOV     R0,MAXMEM         ;SET AS DEFAULT UPPER LIMIT FOR JOB
                                        ;
                                        ;  Switch to stack in job's context area.
                                        ;
002716  012706  0000006                 MOV     #JSTK,SP          ;SWITCH TO CONTEXT-BLOCK STACK
                                        ;
                                        ;  Enter code to load KMON.
                                        ;
002722  000466                          BR      LDKMON            ;GO LOAD KMON
```

```
                                        .SBTTL  STOP   -- Stop program execution & enter KMON
                                ;-------------------------------------------------------------------
                                ;  STOP is jumped to when the job wants to terminate its execution and
                                ;  enter KMON.  This is usually caused by .EXIT, .CHAIN or CTRL-C.
                                ;
002724  113701  000000G        STOP:   MOVB    CORUSR,R1       ;GET JOB # OF CURRENT JOB
002730  001007                        BNE     1$              ;IF SHOULD NOT BE ZERO
002732                                  DIE     #EM$SJN,(SP)    ;DIE IF JOB # = 0
002750  012706  000000G        1$:     MOV     #JSTK,SP        ;RUN ON JOB'S CONTEXT-BLOCK STACK
002754                                  ENABL                   ;MAKE SURE INTERRUPTS ARE ENABLED
002762  052761  000000G 000000G        BIS     #$NOUCR,LSW9(R1);Tell DOCMPL not to run user compl routines
002770  012737  177777  000000G        MOV     #-1,EMTLEV      ;SAY JOB IS NOT EXECUTING AN EMT
002776  052737  000000G 000000G        BIS     #UPMODE,@#PSW   ;MAKE SURE PREVIOUS-MODE = USER
003004  032761  000000G 000000G        BIT     #$INKMN,LSW4(R1);IS KMON RUNNING NOW?
003012  001010                        BNE     2$              ;BR IF YES
003014  106537  000052                MFPD    @#52            ;GET JOB'S ERROR CELLS
003020  000316                        SWAB    (SP)            ;PUT (53) IN LOW-ORDER BYTE
003022  112637  000000G               MOVB    (SP)+,UERSEV    ;SAVE USER SPECIFIED ERROR SEVERITY LEVEL
003026  005046                        CLR     -(SP)           ;NOW CLEAR USER ERROR SEVERITY
003030  106637  000052                MTPD    @#52
                                ;
                                ;  Do general cleanup on exiting job.
                                ;
003034  004737  003366'       2$:     CALL    CLENUP          ;CLEAN UP STATUS OF JOB
003040  042761  000000G 000000G        BIC     #$NOUCR,LSW9(R1);Reenable user completion routine processing
                                ;
                                ;  See if we should force logoff of this job.
                                ;
003046  032761  000000G 000000G        BIT     #$DISCN,LSW(R1) ;DID A LINE DISCONNECT OCCUR?
003054  001411                        BEQ     LDKMON          ;BR IF NOT
003056  052761  000000G 000000G        BIS     #$DOOFF,LSW(R1) ;FORCE LOGOFF
003064  052761  000000G 000000G        BIS     #$NOIN,LSW3(R1) ;IGNORE INPUT FROM LINE DURING LOGOFF
003072  042761  000000G 000000G        BIC     #$DISCN,LSW(R1) ;ACKNOWLEDGE DISCONNECT
                                ;
                                ;  Read KMON into memory and enter it
                                ;
003100  052737  000000G 000000G LDKMON: BIS     #UPMODE,@#PSW   ;SET USER-PREVIOUS-MODE IN PS
003106  042761  173330  000000G        BIC     #173330,LJSW(R1);CLEAN OUT SOME FLAGS IN JOB STATUS WORD
003114  016146  000000G               MOV     LJSW(R1),-(SP)  ;GET CURRENT JOB STATUS WORD
003120  106537  000000G               MFPD    @#ERRLOC        ;GET JOB'S ERROR CELLS
003124  052761  000000G 000000G        BIS     #$INKMN,LSW4(R1);SAY KMON IS RUNNING
003132  042761  000000G 000000G        BIC     #$MAPOK,LSW7(R1);SAY CONTEXT BLOCK MAPPING DATA IS INVALID
003140  042761  000000G 000000G        BIC     #$VIRJB,LSW9(R1);SAY THIS IS NOT A VIRTUAL JOB
003146  105737  000000G               TSTB    VSWPFL          ;IS THIS A NON-SWAPPING SYSTEM?
003152  001406                        BEQ     3$              ;BR IF YES -- DON'T CHANGE MEMORY ALLOCATION
003154  013700  000000G               MOV     KMNTOP,R0       ;GET ADDRESS ABOVE TOP OF KMON
003160  162700  000000G               SUB     #KMNBAS,R0      ;GET AMT OF MEMORY NEEDED FOR KMON
003164  004737  000000G               CALL    SUTOP           ;SET TOP OF MEMORY FOR JOB
003170  004737  000000G        3$:     CALL    SETMAP          ;MAKE SURE MEMORY MAPPING SET FOR KMON
003174  013737  000000G 000000G        MOV     KMNTOP,UHIMEM   ;SAY JOB CAN ACCESS UP TO TOP OF KMON
003202  012637  000000G               MOV     (SP)+,@#ERRLOC  ;SET ERROR CELLS
003206  011637  000000G               MOV     (SP),@#JSWLOC   ;SET JSW
003212  012661  000000G               MOV     (SP)+,LJSW(R1)
                                ;
                                ;  Set up status in user channel # 17 to allow us to access kmon file.
                                ;
003216  012703  000017                MOV     #17,R3          ;USE USER CHANNEL # 17
```

```
     58 003222                                          .PURGE  R3              ;MAKE SURE CHANNEL 17 IS FREE
     59 003232  013703  0000000                         MOV     CXTRMN,R3       ;BASE OF JOB CHANNEL AREA
     60 003236  062703  0000000                         ADD     #R$CH17,R3      ;POINT TO AREA FOR CHANNEL # 17
     61 003242  012700  000005                          MOV     #5,R0           ;# WORDS TO MOVE
     62 003246  012704  0000000                         MOV     #KMNCHN,R4      ;POINT TO BLOCK WITH KMON SAVED STATUS
     63 003252  012423                          1$:     MOV     (R4)+,(R3)+     ;SET UP INFO IN USER CHANNEL 17 BLOCK
     64 003254  077002                                  SOB     R0,1$
     65                                         ;
     66                                         ;  Now read Kmon into user's program space.
     67                                         ;
     68 003256  013703  0000000                         MOV     KMNTOP,R3       ;GET TOP OF MEMORY ADDRESS FOR KMON
     69 003262  162703  0000000                         SUB     #KMNBAS,R3      ;SUBTRACT BASE ADDRESS OF KMON
     70 003266  000241                                  CLC
     71 003270  006003                                  ROR     R3              ;CVT TO # WORDS TO READ
     72 003272                                          .READW  #CSIARE,#17,#0,R3,#32.  ;READ IN KMON
     73 003326  103005                                  BCC     2$              ;BR IF READ OK
     74 003330                                          DIE     #EM$KRE         ;SYSTEM HALT IF KMON READ ERROR
     75                                         ;
     76                                         ;  Set up Kmon user-mode stack pointer.
     77                                         ;
     78 003342  013746  0000000                 2$:     MOV     KMNSTK,-(SP)    ;GET KMON STACK POINTER
     79 003346  106606                                  MTPD    SP              ;SET IN USER-MODE SP
     80                                         ;
     81                                         ;  Give Kmon a full time-slice.
     82                                         ;
     83 003350  005061  0000000                         CLR     LQUAN(R1)       ;CLEAR JOB TIME QUANTUM
     84                                         ;
     85                                         ;  Use RTI to enter Kmon in user mode.
     86                                         ;
     87 003354  012746  0000000                         MOV     #UMODE,-(SP)    ;USER-MODE PS
     88 003360  013746  0000000                         MOV     KMNSTR,-(SP)    ;STARTING ADDRESS IN KMON
     89 003364  000002                                  RTI                     ;ENTER KMON IN USER MODE
```

```
                                        .SBTTL  CLENUP -- Do general cleanup when a job stops
                                ;-------------------------------------------------------------------
                                ;  CLENUP is called to do I/O rundown and other general cleanup
                                ;  operations when a job stops.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
003366  010146                  CLENUP: MOV     R1,-(SP)
003370  010246                          MOV     R2,-(SP)
003372  010446                          MOV     R4,-(SP)
003374  010546                          MOV     R5,-(SP)
003376  042761  000000C 000000G          BIC     #<$RDSAV!$GEMAR>,LSW11(R1);No longer reading in SAV file
003404  005061  000000G                  CLR     LSLEPH(R1)      ;CLEAR .TWAIT SLEEP TIME
003410  005061  000000G                  CLR     LSLEPL(R1)
003414  005061  000000G                  CLR     LSPND(R1)       ;CLEAR .SPND COUNT FOR JOB
003420  042761  000000C 000000G          BIC     #<$IOMAP!$MLOCK!$DBGMD!$NOLF>,LSW6(R1) ;CLEAN OUT LSW6
003426  042761  000000C 000000G          BIC     #$DBGBK!$NOABT!$RNMLK,LSW9(R1);Clear LSW9 flags
003434  005061  000000G                  CLR     LRDTIM(R1)      ;CLEAR ANY TT-READ TIMEOUT
003440  005061  000000G                  CLR     LSCCA(R1)       ;REMOVE CONTROL-C TRAP CONTROL
003444  005061  000000G                  CLR     LBRKCH(R1)      ;CLEAR BREAK CHARACTER
003450  005037  000000G                  CLR     EMTRAD
003454  005037  000000G                  CLR     SPCPS
003460  012737  000000G 000000G          MOV     #EMTCAS,EMTCAD  ;Reset return stack for completion routines
003466  005037  000000G                  CLR     D.FLAG          ;Clear debugger control flags
                                ;
                                ;  If double control-C was typed, echo this to the log file
                                ;
003472  032761  000000G 000000G          BIT     #$CTRLC,LSW(R1) ;Were we aborted by 2 ctrl-c's?
003500  001424                          BEQ     12$             ;Br if not
003502  105037  000000G                  CLRB    CINFLG          ;Reset chain-in-progress flag
003506  032737  000000G 000000G          BIT     #LF$IN,LOGFLG   ;Are we logging input characters?
003514  001413                          BEQ     11$             ;Br if not
003516  012700  000003                  MOV     #3,R0           ;Echo ^c
003522                                  OCALL   LOGCHR
003530                                  OCALL   LOGCHR          ; twice
003536                                  OCALL   LOGCR           ;Log Cr Lf
003544  042761  000000G 000000G  11$:    BIC     #$CTRLC,LSW(R1) ;Clear control-C abort flag
                                ;
                                ;  Abort all I/O for this job
                                ;
003552  105737  000000G          12$:    TSTB    CINFLG          ;Are we doing a .CHAIN?
003556  001004                          BNE     16$             ;Br if yes
003560  004737  000000G                  CALL    CANIOT          ;Cancel any pending .TIMIO requests for job
003564  004737  000000G                  CALL    IOHALT          ;Abort pending I/O operations for this job
                                ;
                                ;  Do any real-time associated cleanup
                                ;
003570                          16$:    OCALL   RTSTOP          ;DO REAL-TIME CLEANUP
                                ;
                                ;  Free the USR
                                ;
003576  120137  000000G          4$:     CMPB    R1,USRJOB       ;ARE WE HOLDING THE USR?
003602  001004                          BNE     15$             ;BR IF NOT
003604                                  OCALL   FREUSR          ;RELEASE IT
003612  000771                          BR      4$              ;WE MAY HAVE LOCKED IT MORE THAN ONCE
                                ;
```

```
                                        ;  Free job context block buffer
                                        ;
003614  120137  0000000G       15$:     CMPB    R1,CXBOWN       ;Are we holding context block buffer?
003620  001003                         BNE     3$              ;Br if not
003622                                   OCALL   FRECXT          ;Free context block buffer
                                        ;
                                        ;  Free the Special Device data base
                                        ;
003630                           3$:     OCALL   FRESPD          ;FREE SPECIAL DEVICE DATA BASE
                                        ;
                                        ;  Cancel any pending mark-time requests for this job
                                        ;
003636                                   OCALL   CANMKT          ;CANCEL ALL MARK-TIME REQUESTS FOR JOB
                                        ;
                                        ;  Cancel any monitoring requests for this job
                                        ;
003644                                   OCALL   MONABT          ;Cancel any monitoring requests
                                        ;
                                        ;  Cancel any pending message requests for this job
                                        ;
003652  005737  0000000G                 TST     VMAXMC          ;Is message system included in system?
003656  001403                           BEQ     14$             ;Br if not
003660                                   OCALL   MSGABT          ;Cleanup message system
                                        ;
                                        ;  Undo Break key connection.
                                        ;
003666  016100  0000000G       14$:     MOV     LBRKCQ(R1),R0   ;GET ADDRESS OF BREAK QUEUE ENTRY
003672  001407                           BEQ     17$             ;BR IF NONE
003674  005061  0000000G                 CLR     LBRKCQ(R1)      ;SAY NO BREAK KEY CONNECTION
003700  010146                           MOV     R1,-(SP)        ;SAVE JOB INDEX NUMBER
003702  010001                           MOV     R0,R1           ;GET QUEUE ENTRY ADDRESS FOR QFREE
003704  004737  0000000G                 CALL    QFREE           ;FREE THE QUEUE ENTRY
003710  012601                           MOV     (SP)+,R1        ;GET BACK JOB NUMBER
                                        ;
                                        ;  Undo TT input completion routine connection
                                        ;
003712  016100  0000000G       17$:     MOV     LTTCR(R1),R0    ;GET ADDRESS OF QUEUE ENTRY
003716  001407                           BEQ     1$              ;BR IF NONE
003720  005061  0000000G                 CLR     LTTCR(R1)       ;SAY NO COMPL ROUTINE
003724  010146                           MOV     R1,-(SP)        ;SAVE JOB INDEX NUMBER
003726  010001                           MOV     R0,R1           ;GET QUEUE ENTRY ADDRESS FOR QFREE
003730  004737  0000000G                 CALL    QFREE           ;FREE THE QUEUE ENTRY
003734  012601                           MOV     (SP)+,R1        ;GET BACK JOB NUMBER
                                        ;
                                        ;  Clean up all pending completion routine requests for this job
                                        ;
003736  004737  004060'          1$:     CALL    CANCPL          ;Cancel all pending completion routines
                                        ;
                                        ;  Close all shared files for this job.
                                        ;
003742  013704  0000000G       5$:      MOV     JCDB,R4         ;GET ADDRESS OF NEXT CDB FOR JOB
003746  001405                           BEQ     6$              ;BR IF NO MORE
003750  013705  0000020G                 MOV     JCDB+2,R5       ;Get par pointer for CDB
003754  004777  0000000G                 CALL    @CLSCDB         ;CLOSE THE CDB
003760  000770                           BR      5$              ;SEE IF THERE ARE MORE TO DO
                                        ;
                                        ;  Release any PLAS regions created for the job
```

```
     115                                        ;
     116 003762  005737  0000000        6$:     TST     VPLAS           ;IS PLAS SUPPORT GENNED INTO SYSTEM?
     117 003766  001403                         BEQ     13$             ;BR IF NOT
     118 003770                                 OCALL   PLSXIT          ;FREE ANY PLAS REGIONS AND WINDOWS
     119                                        ;
     120                                        ;  Release trap control for job
     121                                        ;
     122 003776  005037  0000000        13$:    CLR     UTRPAD          ;Undo .TRPSET
     123 004002  005037  0000000                CLR     UFPTRP          ;Undo .SFPA
     124 004006  105037  0000000                CLRB    FPUUSE          ;Tell system that FPU is not in use
     125                                        ;
     126                                        ;  Release any associated shared run-time systems
     127                                        ;
     128 004012  005037  0000000                CLR     CURRDB          ;DISASSOCIATE RUN-TIME SYSTEM
     129 004016  105037  0000000                CLRB    DOTRMP          ;Disable fast TRAP mapping
     130 004022  005002                         CLR     R2              ;INIT PAR INDEX
     131 004024  005062  0000000        9$:     CLR     RPAR(R2)        ;RESET PAR MAPPING INFO
     132 004030  005062  0000000                CLR     RPDR(R2)
     133 004034  062702  000002                 ADD     #2,R2           ;ADVANCE INDEX
     134 004040  020227  000016                 CMP     R2,#2*7         ;DONE ALL?
     135 004044  101767                         BLOS    9$              ;LOOP IF MORE TO DO
     136                                        ;
     137                                        ;  Finished
     138                                        ;
     139 004046  012605                 10$:    MOV     (SP)+,R5
     140 004050  012604                         MOV     (SP)+,R4
     141 004052  012602                         MOV     (SP)+,R2
     142 004054  012601                         MOV     (SP)+,R1
     143 004056  000207                         RETURN
```

```
 1                                              .SBTTL  CANCPL -- Cancel all pending completion routines
 2                                      ;-----------------------------------------------------------------------
 3                                      ;  CANCPL is called during job exit cleanup to cancel any pending completion
 4                                      ;  routines.  User-mode completion routines are removed from the pending
 5                                      ;  list, system-mode completion routines are forced to be called.
 6                                      ;
 7                                      ;  Inputs:
 8                                      ;   R1 = Job index number.
 9                                      ;
10 004060  010546                      CANCPL: MOV     R5,-(SP)
11                                      ;
12                                      ;  Say we are not in a completion routine now
13                                      ;
14 004062  105037  0000000      1$:     CLRB    CURCP           ;Say not executing in completion rtn now
15                                      ;
16                                      ;  If any I/O is in progress for job, suspend execution while we
17                                      ;  wait for it to finish.
18                                      ;
19 004066                               DISABL                  ;;;** Disable interrupts **
20 004074  126137  0000000 0000000      CMPB    LIOCNT(R1),NPCCB;;;Is any I/O in progress for job?
21 004102  003404                       BLE     2$              ;;;Br if not
22 004104  012700  0000000              MOV     #S$IOWT,R0      ;;;Suspend job till I/O completes
23 004110  004737  0000000              CALL    QNSPND          ;;;Suspend execution of job
24                                      ;
25                                      ;  See if there are any pending completion routines
26                                      ;
27 004114                       2$:     ENABL                   ;** Enable interrupts **
28 004122  005761  0000000              TST     LCMPL(R1)       ;Is there a pending completion routine?
29 004126  001005                       BNE     3$              ;Br if yes
30 004130  126137  0000000 0000000      CMPB    LIOCNT(R1),NPCCB;Is all I/O finished too?
31 004136  003351                       BGT     1$              ;If not, then wait till it is
32 004140  000403                       BR      5$              ;All I/O is finished and no compl rtns pend
33                                      ;
34                                      ;  There is at least one pending completion routine.
35                                      ;  Now call SCHED to force pending system-mode routines to be run
36                                      ;
37 004142  004737  0000000      3$:     CALL    SCHED           ;Force completion routine execution
38                                      ;
39                                      ;  Go back and make sure all completion routines have been taken care of
40                                      ;
41 004146  000745                       BR      1$              ;Loop till all completion routines finished
42                                      ;
43                                      ;  All I/O has terminated and there are no pending completion routines.
44                                      ;  Free any cache control blocks that were pending when job aborted.
45                                      ;
46 004150  005737  0000000      5$:     TST     JOBCCB          ;Any pending cache control blocks
47 004154  001406                       BEQ     9$              ;Br if not
48 004156  013705  0000000      6$:     MOV     JOBCCB,R5       ;Get pointer to 1st pending control block
49 004162  001737                       BEQ     1$              ;Br if none left pending
50 004164  004777  0000000              CALL    @CSHFIN         ;Free the cache control block
51 004170  000772                       BR      6$              ;Loop to free any others
52                                      ;
53                                      ;  Finished
54                                      ;
55 004172  012605                   9$: MOV     (SP)+,R5
56 004174  000207                       RETURN
```

```
                                        .SBTTL  LOGOFF -- Log off a job
                                ;-----------------------------------------------------------------------
                                ;  LOGOFF is jumped to log off the current job.
                                ;  All tables for the job are reset and then the scheduler is entered
                                ;  to look for another job to run.
                                ;
                                ;  Inputs:
                                ;   R1 = Job number of job being logged off.
                                ;
004176  032761  000000G 000000G LOGOFF: BIT     #$DETCH,LSW(R1) ;IS THIS A DETACHED JOB LOGGING OFF?
004204  001402                        BEQ     26$             ;Br if not
004206  000137  004666'                JMP     4$
                                ;
                                ;  Wait for all TT output for the job to complete.
                                ;
004212  032761  000000G 000000G 26$:    BIT     #$VNOTT,LSW(R1) ;IS JOB CONNECTED TO TERMINAL?
004220  001045                        BNE     12$             ;BR IF NOT (DON'T WAIT FOR OUTPUT)
004222  013704  000000G               MOV     SYTIMH,R4       ;Get high-order time-of-day
004226  013705  000000G               MOV     SYTIML,R5       ;Get low-order time-of day
004232  062705  000454                ADD     #5.*60.,R5      ;Add time to allow for logoff message
004236  005504                        ADC     R4              ;Propogate carry
004240  023704  000000G        10$:    CMP     SYTIMH,R4       ;Have we waited long enough?
004244  101033                        BHI     12$             ;Br if yes
004246  103403                        BLO     16$             ;Br if not
004250  023705  000000G               CMP     SYTIML,R5       ;Compare low-order time
004254  103027                        BHIS    12$             ;Br if have waited long enough
004256  116103  000000G        16$:    MOVB    LNPRIM(R1),R3   ;GET PRIMARY LINE NUMBER
004262  032763  000000G 000000G       BIT     #$DILUP,LSW(R3) ;IS PRIMARY LINE STILL LOGGED ON?
004270  001421                        BEQ     12$             ;BR IF NOT
004272  042763  000000G 000000G       BIC     #$CTRLS,LSW3(R3);CLEAR CTRL-S SUSPEND
004300  004777  000000G               CALL    @TRNSTR         ;MAKE SURE OUTPUT IS GOING
004304  026161  000000G 000000G       CMP     LOTSPC(R1),LOTSIZ(R1);ANY CHARS LEFT TO TRANSMIT?
004312  001352                        BNE     10$             ;WAIT FOR ALL OUTPUT TO COMPLETE
004314  032761  000000C 000000G       BIT     #<$DHBF1!$DHBF2>,LSW10(R1);DH11 buffer being transmitted?
004322  001346                        BNE     10$             ;Wait for DH11 to finish
004324  032761  000000G 000000G       BIT     #$XCHAR,LSW3(R1);Wait for last char to go out
004332  001342                        BNE     10$
                                ;
                                ;  See if this is a primary or virtual line logging off.
                                ;
004334  020127  000000G        12$:    CMP     R1,#LSTPL       ;PRIMARY OR VIRTUAL LINE?
004340  003466                        BLE     1$              ;BR IF PRIMARY LINE
                                ;
                                ;  Log off a virtual line.
                                ;
004342  105337  000000G               DECB    PVON            ;Count # primary & virtual lines on
004346  010102                        MOV     R1,R2           ;Save virtual line number
004350  113703  000000G               MOVB    SPIJ,R3         ;Get # of process that started us
004354  001422                        BEQ     22$             ;Br if unknown
004356  032763  000000G 000000G       BIT     #$DISCN,LSW(R3) ;Is that process terminating now?
004364  001016                        BNE     22$             ;Br if yes -- Switch to primary
004366  016100  000000G               MOV     LNPRIM(R1),R0   ;Get our primary process #
004372  020300                        CMP     R3,R0           ;Switching back to primary process?
004374  001412                        BEQ     22$             ;Br if yes
004376  016000  000000G               MOV     LSECPT(R0),R0   ;Get pointer to subprocess # table
004402  005005                        CLR     R5              ;Init subprocess #
004404  005205                24$:    INC     R5              ;Increment subprocess #
```

```
     58 004406  020527  0000000                    CMP     R5,#MAXSEC      ;Checked all subprocess entries?
     59 004412  101003                             BHI     22$             ;Br if yes -- Switch back to primary
     60 004414  120320                             CMPB    R3,(R0)+        ;Search for originating process in table
     61 004416  001372                             BNE     24$             ;Loop till found
     62 004420  000407                             BR      23$             ;Found subprocess to switch back to (R5=#)
     63 004422  005005                       22$:  CLR     R5              ;Say we are switching to primary process
     64 004424  016103  0000000                    MOV     LNPRIM(R1),R3   ;Get primary line number
     65 004430  032763  0000000 0000000            BIT     #$DILUP,LSW(R3) ;Is the primary line still logged on?
     66 004436  001423                             BEQ     2$              ;Br if not
     67 004440  016203  0000000              23$:  MOV     LNPRIM(R2),R3   ;Get primary process index
     68 004444  120163  0000000                    CMPB    R1,LNMAP(R3)    ;Are we running on line being logged off?
     69 004450  001003                             BNE     6$              ;Br if not -- Someone must have killed us
     70 004452                                     OCALL   LINSWT          ;Switch back to initiating process
     71                                       ;
     72                                       ;  At this point R3 has the primary line number.
     73                                       ;  R1 & R2 have the virtual line number.
     74                                       ;  Remove the virtual line from the primary line's ownership table.
     75                                       ;
     76 004460  005004                       6$:   CLR     R4              ;CHECK 1ST VIRTUAL LINE ENTRY
     77 004462  016305  0000000                    MOV     LSECPT(R3),R5   ;POINT TO TABLE OF VIRTUAL LINE #'S
     78 004466  120225                       5$:   CMPB    R2,(R5)+        ;IS THIS THE VIRTUAL LINE ENTRY?
     79 004470  001405                             BEQ     3$              ;BR IF YES
     80 004472  005204                             INC     R4              ;CHECK NEXT ENTRY
     81 004474  020427  0000000                    CMP     R4,#MAXSEC      ;CHECKED ALL?
     82 004500  002772                             BLT     5$              ;BR IF MORE TO CHECK
     83 004502  000401                             BR      2$              ;STRANGE -- COULDN'T FIND VIRTUAL LINE
     84 004504  105045                       3$:   CLRB    -(R5)           ;SAY VIRTUAL LINE NOT OWNED BY PRIMARY LINE
     85 004506  005062  0000000              2$:   CLR     LNPRIM(R2)      ;REMOVE PRIMARY LINE NUMBER FOR VIRTUAL LINE
     86 004512  010201                             MOV     R2,R1           ;GET BACK VIRTUAL LINE NUMBER
     87 004514  000464                             BR      4$
     88                                       ;
     89                                       ;  Disconnect a primary line.
     90                                       ;  This forces the disconnect of any associated virtual lines.
     91                                       ;
     92 004516  105337  0000000              1$:   DECB    NUMON           ;# PRIMARY LINES ON
     93 004522  105337  0000000                    DECB    PVON            ;# PRIMARY & VIRTUAL LINES ON
     94 004526  013700  0000040                    MOV     EMTBLK+4,R0     ;Get time to drop DTR
     95 004532  001002                             BNE     21$             ;Br if time parameter specified
     96 004534  013700  0000000                    MOV     VOFFTM,R0       ;Get sysgen specified time
     97 004540  010061  0000000              21$:  MOV     R0,LOFFTM(R1)   ;Drop DTR after this much time
     98 004544  016161  0000000 0000000            MOV     ILSW2(R1),LSW2(R1);Reset line status flags
     99 004552  010103                             MOV     R1,R3           ;SAVE PRIMARY LINE NUMBER
    100 004554  010161  0000000                    MOV     R1,LNMAP(R1)    ;REASSOCIATE TS LINE WITH PRIMARY LINE
    101                                       ;
    102                                       ;  If this is an autobaud line, start time which will reset the line
    103                                       ;  speed to 9600 baud after a short delay to get the logoff message out.
    104                                       ;
    105 004560  032761  0000000 0000000            BIT     #$AUTO,ILSW2(R1);Is autobaud selected for this line?
    106 004566  001406                             BEQ     14$             ;Br if not
    107 004570  012761  000012  0000000            MOV     #10.,LABTIM(R1) ;Start autobaud timer for this line
    108 004576  052761  0000000 0000000            BIS     #$NABRS,LSW9(R1);Set flag saying we need to reset the speed
    109                                       ;
    110                                       ;  Reset all character translation for the line
    111                                       ;
    112 004604  016104  0000000              14$:  MOV     LCXTBL(R1),R4   ;Get pointer to lines translation table
    113 004610  001401                             BEQ     25$             ;Br if no translation table
    114 004612  005014                             CLR     (R4)            ;Say no translation in effect
```

```
                                        ;
                                        ;  Force disconnect of all associated virtual lines.
                                        ;
004614  005004                  25$:    CLR     R4                  ;START WITH 1ST VIRTUAL LINE
004616  016305  0000000                 MOV     LSECPT(R3),R5       ;GET ADDRESS OF VIRTUAL LINE TABLE
004622  020427  0000000         9$:     CMP     R4,#MAXSEC          ;CHECKED ALL VIRTUAL LINE ENTRIES
004626  002016                          BGE     7$                  ;BR IF YES
004630  111501                          MOVB    (R5),R1             ;GET VIRTUAL LINE NUMBER
004632  001411                          BEQ     8$                  ;BR IF NO ASSOCIATED VIRTUAL LINE HERE
004634  032761  0000000 0000000         BIT     #$LOFCF,LSW9(R1);IS JOB DOING LOGOFF PROCESSING NOW?
004642  001005                          BNE     8$                  ;BR IF YES
004644  052761  0000000 0000000         BIS     #$DISCN,LSW(R1) ;FORCE LOG OFF OF THIS VIRTUAL LINE
004652  004737  0000000                 CALL    FORCEX              ;FORCE ITS EXECUTION
004656  105025                  8$:     CLRB    (R5)+               ;CLEAR ENTRY IN PRIMARY LINE'S TABLE
004660  005204                          INC     R4                  ;CHECK NEXT ENTRY
004662  000757                          BR      9$
004664  010301                  7$:     MOV     R3,R1               ;GET BACK PRIMARY LINE NUMBER
                                        ;
                                        ;  Release any display windows for job
                                        ;
004666  005737  0000000         4$:     TST     VMXWIN              ;Is window support included in system?
004672  001403                          BEQ     20$                 ;Br if not
004674                                  OCALL   WINREL              ;Release windows for job
                                        ;
                                        ;  Do cleanup of PLAS regions
                                        ;
004702  005737  0000000         20$:    TST     VPLAS               ;Is PLAS support included?
004706  001403                          BEQ     17$                 ;Br if not
004710                                  OCALL   PLSOFF              ;Do PLAS cleanup
                                        ;
                                        ;  Say we are no other job's parent
                                        ;
004716  012702  0000000         17$:    MOV     #LSTSL,R2           ;Get # of last job
004722  026201  0000000         18$:    CMP     LPARNT(R2),R1       ;Are we that job's parent?
004726  001002                          BNE     19$                 ;Br if not
004730  005062  0000000                 CLR     LPARNT(R2)          ;No longer its parent
004734  162702  000002          19$:    SUB     #2,R2               ;Check other jobs
004740  003370                          BGT     18$
                                        ;
                                        ;  Free all memory assigned to the job.
                                        ;
004742  016102  0000000                 MOV     LBASE(R1),R2        ;GET BASE PAGE # ASSIGNED TO JOB
004746  016100  0000000                 MOV     LNBLKS(R1),R0       ;GET # PAGES ASSIGNED TO JOB
004752  004737  0000000                 CALL    FREMEM              ;RELEASE THE MEMORY SPACE
004756  005061  0000000                 CLR     LNBLKS(R1)          ;SAY ALL MEMORY DEASSIGNED
004762  005061  0000000                 CLR     LMEMIN(R1)
004766  005061  0000000                 CLR     LBASE(R1)
                                        ;
                                        ;  Now clear line tables.
                                        ;
004772  004737  0000000                 CALL    DEQ                 ;REMOVE JOB FROM RUN QUEUE  ** DISABLE **
004776  005061  0000000                 CLR     LSW(R1)             ;CLEAR LINE STATUS TABLES
005002  005061  0000000                 CLR     LSW4(R1)
005006  005061  0000000                 CLR     LSW5(R1)
005012  042761  000000C 0000000         BIC     #^C<$1STLG>,LSW6(R1);CLEAR ALL BUT 1ST-LOGON FLAG
005020  005061  0000000                 CLR     LSW7(R1)
005024  012700  000000C                 MOV     #<$DEAD!$HARD!$CARUP!$XCHAR>,R0 ;FLAGS TO PRESERVE IN LSW3
```

```
    172 005030  005100                          COM     R0              ;MASK TO CLEAR ALL OTHERS
    173 005032  040061  0000000G                BIC     R0,LSW3(R1)
    174 005036  005061  0000000G                CLR     LPARNT(R1)      ;Say we have no parent job
    175 005042  005061  0000000G                CLR     LNSPAC(R1)      ;Say no user-defined activation characters
    176 005046  105337  0000000G                DECB    TOTON           ;TOTAL # JOBS
    177 005052  120137  0000000G                CMPB    R1,PMUSER       ;ARE WE DOING A PERFORMANCE ANALYSIS?
    178 005056  001004                          BNE     13$             ;BR IF NOT
    179 005060  005037  0000000G                CLR     PMUSER          ;SAY WE ARE DONE
    180 005064  005037  0000000G                CLR     PMRUN
    181                             ;
    182                             ;  Job is logged off.
    183                             ;  Enter scheduler to find another one to run.
    184                             ;
    185 005070  012706  0000000G    13$:        MOV     #SS,SP          ;SWITCH TO SYSTEM STACK
    186 005074  105037  0000000G                CLRB    CORUSR          ;NO USER RUNNING
    187 005100                                  ENABL                   ;** ENABLE **
    188 005106  105037  0000000G                CLRB    MAPUSR          ;SAY MEMORY MAPPING NOT SET UP FOR ANY JOB
    189 005112  000137  0000000G                JMP     EXEC            ;GO LOOK FOR ANOTHER JOB TO RUN
```

```
                                        .SBTTL  TSXTX  -- Trap Handler
                                ;-------------------------------------------------------------------------
                                ;  TSXTX is entered from the resident routines that catch traps to 4 and 10.
                                ;
                                ;  Inputs:
                                ;    The following items are on the stack:
                                ;       0(SP) = Saved R5
                                ;       2(SP) = Saved R4
                                ;       4(SP) = Trap PC
                                ;       6(SP) = Trap PS
                                ;       R5    = Trap code (1==>Trap 4, 2==>Trap 10)
                                ;
                                ;  See if trap occured in user or kernel mode.
                                ;
005116  016604  000004         TSXTX:  MOV     4(SP),R4        ;GET ADDRESS OF TRAP
005122  032766  000000G 000006         BIT     #UMODE,6(SP)    ;DID TRAP OCCUR IN USER OR KERNEL MODE?
005130  001036                        BNE     1$              ;BR IF TRAP IN USER MODE
005132  012705  000016                MOV     #16,R5          ;SET KERNEL-MODE-TRAP ERROR CODE
005136  105737  000000G               TSTB    CORUSR          ;IS A JOB RUNNING NOW?
005142  001416                        BEQ     6$              ;IF NOT THEN TRAP MUST BE IN SYSTEM
005144  005737  000000G               TST     CURVC           ;IN REAL-TIME INTERRUPT SERVICE ROUTINE?
005150  001013                        BNE     6$              ;BR IF YES
005152  105737  000000G               TSTB    FRKPRI          ;IN A FORK ROUTINE?
005156  001010                        BNE     6$              ;BR IF YES
005160  105737  000000G               TSTB    INTLVL          ;ARE WE RUNNING AT INTERRUPT LEVEL?
005164  002005                        BGE     6$              ;BR IF YES
005166  105737  000000G               TSTB    VDMKTP          ;SHOULD WE ALWAYS CRASH ON KERNEL TRAP?
005172  001002                        BNE     6$              ;BR IF YES
005174  000137  005460'               JMP     TRPCOM          ;ALWAYS ABORT IF KERNEL MODE TRAP
                                ;
                                ;  We had a trap within a critical system routine.
                                ;  Cause a system crash.
                                ;
005200  010437  000000G         6$:     MOV     R4,DIEAR@       ;Set address of trap location
005204  012737  000170  000000G        MOV     #EM$KTP,DIEMS@  ;Set address of abort message
005212  012605                        MOV     (SP)+,R5        ;Restore R5
005214  012604                        MOV     (SP)+,R4        ;Restore R4
005216  062706  000004                ADD     #4,SP           ;Pop trap PC and PS
005222  004737  000000G               CALL    SYSHL1          ;Die without changing TRPAR5
                                ;
                                ;  Trap in user mode.
                                ;  See if user was executing a real-time interrupt service routine.
                                ;
005226  005737  000000G         1$:     TST     CURVC           ;ARE WE EXECUTING IN A REAL-TIME INT ROUTINE?
005232  001407                        BEQ     7$              ;BR IF NOT
005234                                  DIE     #EM$RIT,R4      ;TRAP IN REAL-TIME INTERRUPT SERVICE ROUTINE
                                ;
                                ;  See if a stack overflow occured.
                                ;
005252  004737  000000G         7$:     CALL    CHKUSP          ;IS USER STACK POINTER OK?
005256  103002                        BCC     2$              ;BR IF OK
005260  000137  005644'               JMP     ABORT           ;ABORT JOB
                                ;
                                ;  See if user did a .TRPSET
                                ;
005264  005737  000000G         2$:     TST     UTRPAD          ;DID USER DO A .TRPSET?
005270  001010                        BNE     3$              ;BR IF YES
```

```
                                ;
                                ;  User did a .TRPSET
                                ;
005312  004737  0000000G  3$:   CALL    CHKABT          ;MAKE SURE JOB HASN'T BEEN ABORTED
                                ;  Move trap PC & PS from kernel stack to user's stack.
005316  106506                  MFPD    SP              ;GET USER'S SP
005320  012604                  MOV     (SP)+,R4
005322  016646  000006          MOV     6(SP),-(SP)     ;PUSH TRAP PS ON USER'S STACK
005326  106644                  MTPD    -(R4)
005330  016646  000004          MOV     4(SP),-(SP)     ;PUSH TRAP PC ON USER'S STACK
005334  106644                  MTPD    -(R4)
005336  010446                  MOV     R4,-(SP)        ;STORE UPDATED USER SP
005340  106606                  MTPD    SP
                                ;
                                ;  Enter user's .trpset routine.
                                ;
005342  012704  000000C         MOV     #UMODE!UPMODE,R4;GET USER-MODE PS
005346  020527  000002          CMP     R5,#2           ;WAS TRAP TO 4 OR 10?
005352  001001                  BNE     4$              ;BR IF TRAP TO 4
005354  005204                  INC     R4              ;SET C-FLAG IN PS TO SIGNAL TRAP TO 10
005356  010466  000006  4$:     MOV     R4,6(SP)        ;STORE NEW PS OVER TRAP PS
005362  013766  0000000G 000004 MOV     UTRPAD,4(SP)    ;SET ADDRESS OF USER'S ROUTINE
005370  005037  0000000G        CLR     UTRPAD          ;RESET .TRPSET
005374  012605                  MOV     (SP)+,R5
005376  012604                  MOV     (SP)+,R4
005400  000002                  RTI                     ;ENTER USER'S TRAP ROUTINE
```

```
                                ;+------------------------------------------------------------------------------
                                ;  TRPBPT is entered when a breakpoint (BPT) trap occurs to location 14
                                ;  and the system debugger is not connected to the user's job.
                                ;
                                ;  Inputs:
                                ;    R4 = Job index number
                                ;  Information that has been pushed on the stack:
                                ;    PS-PC-R4
                                ;
005402                          TRPBPT:
                                ;
                                ;  If breakpoint occurred in TSKMON, enter debugger
                                ;
005402  032764  0000000 0000000         BIT     #$INKMN,LSW4(R4);Is kmon running?
005410  001004                          BNE     1$              ;Br if yes
                                ;
                                ;  See if user provided a PC in location 14
                                ;
005412  106537  000014                  MFPD    @#14            ;Get contents of loc 14 from user's job
005416  005726                          TST     (SP)+           ;Did user provide a PC for trap?
005420  001011                          BNE     9$              ;Br if yes
                                ;
                                ;  User did not provide a PC for trap.
                                ;  Enter system debugger.
                                ;
005422  105737  0000000         1$:     TSTB    VDBFLG          ;Is system debugger included in system?
005426  001406                          BEQ     9$              ;Br if not
005430  052764  0000000 0000000         BIS     #$DBGBK,LSW9(R4);Set flag to force entry to debugger
005436  012604                          MOV     (SP)+,R4        ;Restore R4
005440  000137  0000000                 JMP     SYSXIT          ;Exit through routine that will test flag
                                ;
                                ;  User provided a PC for BPT trap, enter his routine.
                                ;
005444  010546                  9$:     MOV     R5,-(SP)
005446  012705  000012                  MOV     #12,R5          ;Get error code
005452  012704  000014                  MOV     #14,R4          ;Get trap location
005456  000400                          BR      TRPCOM          ;Enter trap processing routine
```

```
                                ;-----------------------------------------------------------------------
                                ;  General trap handling routine.
                                ;
                                ;  Inputs:
                                ;   R4 = Address of trap vector.
                                ;   R5 = Error message number.
                                ;
                                ;  If user provided a PC & PS in the user-mode trap vector, then we enter
                                ;  his routine.  Otherwise we abort the job.
                                ;
005460  010146                  TRPCOM: MOV     R1,-(SP)
                                ;
                                ;  At this point the stack contains PS, PC, R4, R5, and R1
                                ;
005462  032766  0000000 000010          BIT     #UMODE,8.(SP)   ;DID TRAP OCCUR IN USER OR KERNEL MODE?
005470  001005                         BNE     4$              ;Br if in user mode
005472  032737  0000000 0000000         BIT     #D$RUN,D.FLAG   ;Is the debugger program running now?
005500  001046                         BNE     3$              ;Br if yes -- reenter the debugger
005502  000456                         BR      2$              ;Trap occurred within the system
005504  004737  0000000         4$:     CALL    CHKUSP          ;SEE IF USER STACK POINTER IS OK
005510  103453                         BCS     2$              ;BR IF INVALID STACK POINTER
005512  113701  0000000                MOVB    CORUSR,R1       ;GET CURRENT JOB NUMBER
005516  032761  0000000 0000000         BIT     #$INKMN,LSW4(R1);DID TRAP OCCUR IN KMON?
005524  001045                         BNE     2$              ;ALWAYS ABORT IF YES
005526  106524                         MFPD    (R4)+           ;GET PC FROM USER SPACE
005530  005726                         TST     (SP)+           ;DID USER PROVIDE PC?
005532  001430                         BEQ     1$              ;BR IF NOT -- ABORT THE JOB
                                ;  User supplied a PC.
005534  004737  0000000                CALL    CHKABT          ;MAKE SURE JOB HASN'T BEEN ABORTED
                                ;  Move trap PC & PS from kernel stack to user stack.
005540  106506                         MFPD    SP              ;GET USER'S SP
005542  012605                         MOV     (SP)+,R5
005544  016646  000010                 MOV     8.(SP),-(SP)    ;GET PS FROM TRAP
005550  106645                         MTPD    -(R5)           ;PUSH ONTO USER'S STACK
005552  016646  000006                 MOV     6(SP),-(SP)     ;GET PC FROM TRAP
005556  106645                         MTPD    -(R5)           ;PUSH ONTO USER'S STACK
005560  010546                         MOV     R5,-(SP)        ;UPDATE USER'S SP
005562  106606                         MTPD    SP
                                ;
                                ;  Enter user's trap routine
                                ;
005564  106514                         MFPD    (R4)            ;GET PS FROM TRAP VECTOR
005566  052716  000000C                BIS     #UMODE!UPMODE,(SP);MAKE SURE USER-MODE IS SET
005572  012666  000010                 MOV     (SP)+,8.(SP)    ;STORE OVER TRAPPED PS
005576  106544                         MFPD    -(R4)           ;GET PC FROM TRAP VECTOR
005600  012666  000006                 MOV     (SP)+,6(SP)     ;STORE OVER TRAPPED PC
005604  012601                         MOV     (SP)+,R1
005606  012605                         MOV     (SP)+,R5        ;RESTORE REGISTERS
005610  012604                         MOV     (SP)+,R4
005612  000002                         RTI                     ;ENTER USER'S TRAP ROUTINE
                                ;
                                ;  User did not specify a trap routine.
                                ;  If program is running with the debugger, enter it.
                                ;
005614  005004                  1$:     CLR     R4              ;Set flag saying trap was not in debugger
005616  113701  0000000         3$:     MOVB    CORUSR,R1       ;Get job index number
005622  032761  0000000 0000000         BIT     #$DEBUG,LSW9(R1);Is program running with debugger?
```

```
     58 005630  001403                        BEQ     2$              ;Br if not
     59 005632  012601                        MOV     (SP)+,R1        ;Recover R1
     60 005634  000137  000000G               JMP     DBGTRP          ;Enter debugger
     61                              ;
     62                              ;  Abort the job
     63                              ;
     64 005640  016604  000006      2$:      MOV     6(SP),R4        ;GET PC WHERE TRAP OCCURED
     65                              ;
     66                              ;  Enter at ABORT to abort the current job.
     67                              ;  Inputs:
     68                              ;   R4 = Address of aborted instruction.
     69                              ;   R5 = Abort error code.
     70                              ;
     71 005644  010437  000000G     ABORT:   MOV     R4,ABRTAD       ;SAVE ADDRESS OF ABORT
     72 005650  110537  000000G              MOVB    R5,ABRTCD       ;SAVE ABORT ERROR CODE
     73 005654  004737  002724'              CALL    STOP            ;TERMINATE THE JOB
```

```
                                        .SBTTL  FPTRPX -- Floating point trap routine
                                ;-------------------------------------------------------------------
                                ;  FPTRPX processed Floating Point Unit (FPU) exception interrupts.
                                ;  This routine is jumped to when we are about to exit from an interrupt
                                ;  back to user mode.
                                ;  On entry, the current job index number is in R1.
                                ;  The saved convents of R1 are on the top of the stack followed by the
                                ;  interrupt PC and PS ready to do an RTI.
                                ;
005660  052737  000000G 000000G FPTRPX: BIS     #UPMODE,@#PSW   ;Make sure previous mode = user
005666  042761  000000G 000000G        BIC     #$FPUEX,LSW(R1) ;Reset FPU exception flag for job
005674  012601                  1$:   MOV     (SP)+,R1        ;Recover R1 contents
005676  023727  000000G 000001        CMP     UFPTRP,#1       ;Did user do a .SFPA?
005704  101004                        BHI     1$              ;Br if yes
                                ;
                                ;  User did not do a .SFPA,  Abort the job.
                                ;
005706  012705  000005                MOV     #5,R5           ;Set abort code
005712  011604                        MOV     (SP),R4         ;Get address of aborted instruction
005714  000753                        BR      ABORT           ;Abort the job
                                ;
                                ;  User gets trap control.
                                ;  Push trap PC & PS onto user's stack
                                ;
005716  004737  000000G         1$:   CALL    CHKUSP          ;IS USER'S STACK POINTER OK?
005722  103002                        BCC     2$              ;BR IF OK
005724  011604                        MOV     (SP),R4         ;GET ADDRESS WHERE TRAP OCCURED
005726  000746                        BR      ABORT           ;ABORT THE JOB
005730  010346                  2$:   MOV     R3,-(SP)        ;GET A WORK REGISTER
005732  106506                        MFPD    SP              ;GET USER'S STACK POINTER
005734  012603                        MOV     (SP)+,R3
005736  016646  000004                MOV     4(SP),-(SP)     ;GET TRAP PS
005742  106643                        MTPD    -(R3)           ;PUSH ONTO USER'S STACK
005744  016646  000002                MOV     2(SP),-(SP)     ;GET TRAP PC
005750  106643                        MTPD    -(R3)           ;PUSH ONTO USER'S STACK
                                ;
                                ;  See if hardware has a FPU
                                ;
005752  032737  000000G 000000G        BIT     #CW$FPU,CONFIG  ;DOES HARDWARE HAVE AN FPU UNIT?
005760  001407                        BEQ     3$              ;BR IF NOT
005762  170346                        STST    -(SP)           ;GET FPU STATUS
005764  106663  177774                MTPD    -4(R3)          ;MOVE FPU STATUS ONTO USER'S STACK
005770  106663  177776                MTPD    -2(R3)
005774  162703  000004                SUB     #4,R3           ;UPDATE USER'S SP
                                ;
                                ;  Reset user's SP
                                ;
006000  010346                  3$:   MOV     R3,-(SP)        ;NEW USER SP
006002  106606                        MTPD    SP              ;RESET USER SP
006004  012603                        MOV     (SP)+,R3        ;RESTORE WORK REGISTER
                                ;
                                ;  Enter user's trap routine
                                ;  Note, when user's trap routine does an RTI, it will transfer
                                ;  control to the point we would have exited to if the FPU trap hadn't
                                ;  have occurred.
                                ;
006006  013716  0000000                 MOV     UFPTRP,(SP)     ;SET PC FOR TRAP ROUTINE
```

```
     58 006012  012766  000000C 000002          MOV     #UMODE!UPMODE,2(SP);SET PS
     59 006020  012737  000001  0000000         MOV     #1,UFPTRP           ;RESET .SFPA TO AVOID REENTRENCY
     60 006026  000002                          RTI                         ;ENTER USER'S TRAP ROUTINE
```

```
                                        .SBTTL  CLKRUN -- Clock processing routine
                                ;-------------------------------------------------------------
                                ;  CLKRUN is the clock interrupt service routine entered from TSEXEC
                                ;  running at fork level.
                                ;
006030  013703  000000G       CLKRUN: MOV     TIKCNT,R3       ;GET # CLOCK TICKS THAT HAVE OCCURED
006034  005203                4$:     INC     R3              ;CONVERT TO ACTUAL NUMBER (STARTED AT -1)
006036  010337  000000G               MOV     R3,CLKCNT       ;ADVANCE ALL TIMERS BY THIS AMOUNT
                                ;
                                ;  Keep track of time of day
                                ;
006042  004737  006344'               CALL    CLKDAT          ;ADVANCE TIME-OF-DAY AND DATE
                                ;
                                ;  Keep track of time used by currently running job (if any)
                                ;
006046  113701  000000G               MOVB    CORUSR,R1       ;GET INDEX # OF CURRENTLY RUNNING JOB
006052  001405                        BEQ     3$              ;BR IF NO JOB RUNNING NOW
006054  063761  000000G 000000G       ADD     CLKCNT,LCPULO(R1);ACCUMULATE RUN-TIME FOR JOB
006062  005561  000000G               ADC     LCPUHI(R1)      ;PROPOGATE CARRY
                                ;
                                ;  Check on .MRKT and .TIMIO requests
                                ;
006066  004737  010176'        3$:     CALL    CKMRKT          ;Check on .MRKT and .TIMIO requests
                                ;
                                ;  Check on jobs doing .TWAIT's
                                ;
006072  004737  010076'               CALL    CKTWAT          ;Check on jobs doing .TWAIT's
                                ;
                                ;  Check on job output buffer scheduling requests
                                ;
006076  005737  000000G               TST     NEDSOT          ;Output scheduling needed?
006102  001404                        BEQ     8$              ;Br if not
006104  005037  000000G               CLR     NEDSOT          ;Say output scheduling done
006110  004737  010736'               CALL    CKSCHD          ;Do job scheduling for output buffer low
                                ;
                                ;  See if we need to do performance monitoring.
                                ;
006114  005737  000000G        8$:     TST     PMRUN           ;IS PERFORMANCE MONITORING TO BE DONE?
006120  001402                        BEQ     2$              ;BR IF NOT
006122  004737  010526'               CALL    CLKPM           ;DO PERFORMANCE MONITORING
                                ;
                                ;  If we are running on a Professional, call the PI output service
                                ;  routine every 20th of a second.
                                ;
006126  013700  000000G        2$:     MOV     PIDPTR,R0       ;Are we running on a Pro?
006132  001407                        BEQ     6$              ;Br if not
006134  005337  001340'               DEC     PROTIM          ;Time to call PI driver?
006140  003004                        BGT     6$              ;Br if not
006142  004710                        CALL    (R0)            ;Call PI output driver
006144  012737  000000G 001340'       MOV     #PROODC,PROTIM  ;Reset time counter
                                ;
                                ;  Do clock driven processing of serial lines.
                                ;  We do this as a lower priority fork request since this processing
                                ;  could be lengthy.
                                ;
006152  005737  000000G        6$:     TST     NEDCDI          ;Input character processing needed?
006156  001417                        BEQ     5$              ;Br if not
```

```
                                        TSTB    CDIFLG          ;Are we still doing input char processing?
                                        BNE     5$              ;Br if yes
                                        INCB    CDIFLG          ;Set flag saying processing is being done
                                        CALL    FRKGET          ;Get a fork request block
                                        MOVB    #FP$CDI,FQ$PRI(R4);Set low priority for fork request
                                        MOV     CDIRTN,FQ$RTN(R4);Set address of routine to call
                                        CALL    FORKQ           ;Queue the fork request
                                5$:     TST     NEDCDO          ;Output character processing needed?
                                        BEQ     7$              ;Br if not
                                        TSTB    CDOFLG          ;Are we still doing output char processing?
                                        BNE     7$              ;Br if yes
                                        INCB    CDOFLG          ;Say output processing being done
                                        CALL    FRKGET          ;Get a fork request block
                                        MOVB    #FP$CDO,FQ$PRI(R4);Set priority for fork request
                                        MOV     CDORTN,FQ$RTN(R4);Set address of routine to call
                                        CALL    FORKQ           ;Queue the fork request
                                ;
                                ;  Processing done on 0.1 second frequency.
                                ;  This is also done by queueing a lower priority fork request.
                                ;
                                7$:     SUB     CLKCNT,TK1CNT   ;Has 0.1 seconds of time passed?
                                        BGT     1$              ;Br if not
                                        ADD     TK1VAL,TK1CNT   ;Reset 0.1 counter
                                        INC     TIKO1S          ;Say another 0.1 seconds has elapsed
                                        BGT     1$              ;Br if haven't finished last 0.1 sec routine
                                        CALL    FRKGET          ;Get a fork request block
                                        MOVB    #FP$CK1,FQ$PRI(R4);Set low priority for fork request
                                        MOV     #CLKO1S,FQ$RTN(R4);Set address of routine to be called
                                        CALL    FORKQ           ;Queue the fork request
                                ;
                                ;  Finished clock processing
                                ;
                                1$:     SUB     CLKCNT,TIKCNT   ;SUBTRACT # CLOCK TICKS ACCOUNTED FOR
                                        BGE     CLKRUN          ;BR IF WE NEED TO CYCLE AGAIN
                                        RETURN                  ;FINISHED
```

```
                                        .SBTTL  CLKDAT -- update time of day and date
                                ;-----------------------------------------------------------------------
                                ;  CLKDAT is the timer subroutine called to keep track of the current
                                ;  time-of-day and date.
                                ;
                                ;  Inputs:
                                ;   CLKCNT = # clock ticks to account for.
                                ;
                                ;  Outputs:
                                ;   SYTIML & SYTIMH = Updated time of day.
                                ;   SYSDAT = Updated date.
                                ;
006344  010146                  CLKDAT: MOV     R1,-(SP)
006346  010246                          MOV     R2,-(SP)
006350  010346                          MOV     R3,-(SP)
                                ;
                                ;  Advance system time counter.
                                ;
006352  063737  000000G 000000G         ADD     CLKCNT,SYTIML   ;ADD TO LOW-ORDER WORD
006360  005537  000000G                 ADC     SYTIMH          ;PROPOGATE CARRY
                                ;
                                ;  See if we need to do a date roll-over.
                                ;
006364  023737  000000G 000000G         CMP     SYTIMH,DATIMH   ;COMPARE HIGH-ORDER TIME VALUE
006372  103465                        BLO     9$              ;BR IF NOT UP TO 24 HOURS YET
006374  023737  000000G 000000G         CMP     SYTIML,DATIML   ;COMPARE LOW-ORDER TIME
006402  103461                        BLO     9$              ;BR IF NOT THERE YET
                                ;
                                ;  Do a date roll-over.
                                ;
006404  163737  000000G 000000G         SUB     DATIML,SYTIML   ;RESET SYSTEM TIMER RELATIVE TO START OF DAY
006412  005037  000000G                 CLR     SYTIMH
006416  013700  000000G                 MOV     SYSDAT,R0       ;GET SYSTEM DATE VALUE
006422  001451                        BEQ     9$              ;BR IF NO DATE WAS ENTERED
006424  010003                        MOV     R0,R3           ;GET YEAR FIELD
006426  042703  177740                BIC     #^C<37>,R3
006432  072027  177773                ASH     #-5,R0          ;RIGHT JUSTIFY DAY #
006436  010001                        MOV     R0,R1
006440  042700  177740                BIC     #^C<37>,R0      ;GET DAY # ONLY
006444  072127  177773                ASH     #-5,R1          ;RIGHT JUSTIFY MONTH VALUE
006450  042701  177740                BIC     #^C<37>,R1      ;GET MONTH VALUE ONLY
006454  005200                        INC     R0              ;INCREMENT CURRENT DAY NUMBER
006456  116102  001321'              MOVB    MONDAY-1(R1),R2 ;GET # DAYS IN CURRENT MONTH
006462  020127  000002                CMP     R1,#2           ;IS THIS FEBRUARY?
006466  001004                        BNE     5$              ;BR IF NOT
006470  032703  000003                BIT     #3,R3           ;IS THIS A LEAP YEAR?
006474  001001                        BNE     5$              ;BR IF NOT
006476  005202                        INC     R2              ;SAY FEB HAS 29 DAYS
006500  020002                5$:     CMP     R0,R2           ;HAVE WE PASSED LAST DAY IN THIS MONTH?
006502  101411                        BLOS    6$              ;BR IF NOT
006504  012700  000001                MOV     #1,R0           ;RESET DAY # TO 1
006510  005201                        INC     R1              ;ADVANCE MONTH NUMBER
006512  020127  000014                CMP     R1,#12.         ;DID WE JUST ADVANCE PAST DECEMBER?
006516  101403                        BLOS    6$              ;BR IF NOT
006520  012701  000001                MOV     #1,R1           ;RESET MONTH TO JANUARY
006524  005203                        INC     R3              ;ADVANCE YEAR NUMBER  (HAPPY NEW YEAR)
006526  072027  000005          6$:   ASH     #5,R0           ;POSITION DAY # VALUE
```

CLKDAT -- update time of day and date

```
     58 006532  050003                          BIS     R0,R3           ;OR INTO YEAR WORD
     59 006534  072127  000012                  ASH     #10.,R1         ;POSITION MONTH #
     60 006540  050103                          BIS     R1,R3           ;COMBINE DATE VALUES
     61 006542  010337  0000000                 MOV     R3,SYSDAT       ;SAVE UPDATED DATE VALUE
     62                             ;
     63                             ;  Finished
     64                             ;
     65 006546  012603              9$:         MOV     (SP)+,R3
     66 006550  012602                          MOV     (SP)+,R2
     67 006552  012601                          MOV     (SP)+,R1
     68 006554  000207                          RETURN
```

```
                                        .SBTTL  CLKJOB -- check time slice job status
                                ;-------------------------------------------------------------
                                ;  CLKJOB is the timer subroutine called every 0.1 seconds to check for
                                ;  time-slice expiration of the currently running job.
                                ;
006556  010146                  CLKJOB: MOV     R1,-(SP)
006560  010246                          MOV     R2,-(SP)
                                ;
                                ;  See if there is a job currently executing
                                ;
006562  113701  000000G                 MOVB    CORUSR,R1       ;GET INDEX # FOR CURRENTLY RUNNING JOB
006566  001543                          BEQ     3$              ;BR IF NO JOB RUNNING NOW
                                ;
                                ;  Increment time quantum for job
                                ;
006570  005261  000000G                 INC     LQUAN(R1)       ;Increment time quantum for job
                                ;
                                ;  Check for time slice expiration for fixed-priority real-time
                                ;  and low priority jobs.
                                ;
006574  016100  000000G                 MOV     LQUAN(R1),R0    ;Get current time quantum for job
006600  116102  000000G                 MOVB    LPRI(R1),R2     ;Get job's execution priority
006604  120237  000000G                 CMPB    R2,VPRIHI       ;Is this a high priority (real time) job?
006610  103412                          BLO     10$             ;Br if not
006612  020037  000000G                 CMP     R0,VQUAN0       ;Have we exceeded QUAN0 time?
006616  101527                          BLOS    3$              ;Br if not
006620  005737  000000G                 TST     VQUAN0          ;Are we doing time slicing for real-time jobs?
006624  001524                          BEQ     3$              ;Br if not
006626  004037  000000G                 JSR     R0,QUNSIG       ;Signal that QUAN0 expired
006632  000000G                         .WORD   $SGQ0
006634  000411                          BR      11$             ;Requeue the job at the tail of the list
006636  120237  000000G         10$:    CMPB    R2,VPRILO       ;Is this a low priority job?
006642  101015                          BHI     12$             ;Br if not
006644  020037  000000G                 CMP     R0,VQUAN3       ;Exceeded low priority quantum?
006650  101512                          BLOS    3$              ;Br if not
006652  004037  000000G                 JSR     R0,QUNSIG       ;Signal that QUAN3 has expired
006656  000000G                         .WORD   $SGQ3
                                ;
                                ;  A real time or low priority job has exceeded its time quantum.
                                ;  Requeue the job at the tail of its execution queue.
                                ;
006660  016100  000000G         11$:    MOV     LSTATE(R1),R0   ;Get job's current execution state
006664  004737  000000G                 CALL    ENQTL           ;Requeue job at tail of execution queue
006670  005061  000000G                 CLR     LQUAN(R1)       ;Reset job time quantum
006674  000500                          BR      3$
                                ;
                                ;  This job is not a low priority or real time job.
                                ;  See if current job is an interactive job, and if so decrement
                                ;  its interactive-CPU time.
                                ;
006676  005761  000000G         12$:    TST     LITIME(R1)      ;Is job interactive?
006702  001407                          BEQ     2$              ;Br if not
006704  005361  000000G                 DEC     LITIME(R1)      ;Reduce time remaining for job
006710  001004                          BNE     2$              ;Br if still interactive
006712  004037  000000G                 JSR     R0,QUNSIG       ;Signal that QUAN1 has expired
006716  000000G                         .WORD   $SGQ1           ;Check QUAN1 signal flag
006720  000464                          BR      6$              ;Now schedule job as compute bound
```

```
                                        ;
                                        ;  Check for job quantum expiration.
                                        ;
006722  016100  0000000       2$:       MOV     LQUAN(R1),R0
                                        ;
                                        ;  See if this job is currently running in a high-priority state.
                                        ;
006726  026127  0000000 0000000          CMP     LSTATE(R1),#S$$HIP;Is job in high-priority state now?
006734  101050                           BHI     4$              ;Br if not
                                        ;
                                        ;  Don't do time-slicing for real-time jobs.
                                        ;
006736  026127  0000000 0000000          CMP     LSTATE(R1),#S$$RT;Is job in high-priority real-time state?
006744  101454                           BLOS    3$              ;Br if yes -- skip time-quantum checking
                                        ;
                                        ;  Job is running in a high-priority state.
                                        ;  See if job is interactive.
                                        ;
006746  005761  0000000                  TST     LITIME(R1)      ;Is this an interactive job?
006752  001007                           BNE     5$              ;Br if yes
006754  020037  0000000                  CMP     R0,VQUN1A       ;Time to requeue as compute bound?
006760  101446                           BLOS    3$              ;Br if not
006762  004037  0000000                  JSR     R0,QUNSIG       ;Signal that QUAN1A expired
006766  0000000                          .WORD   $SGQ1A
006770  000440                           BR      6$              ;Schedule as compute bound job
                                        ;
                                        ;  Job is "interactive"
                                        ;
006772  026127  0000000 0000000 5$:       CMP     LSTATE(R1),#S$HICP;High priority interactive?
007000  103011                           BHIS    9$              ;Br if not
007002  020037  0000000                  CMP     R0,VQUN1C       ;Time to drop to lower level?
007006  101406                           BLOS    9$              ;Br if not
007010  004037  0000000                  JSR     R0,QUNSIG       ;Signal that QUAN1C expired
007014  0000000                          .WORD   $SGQ1C
007016  012700  0000000                  MOV     #S$HICP,R0      ;Drop to interactive computation state
007022  000410                           BR      7$
                                    ;
                                    ;  If QUAN1B has expired, requeue job at tail of current queue
                                    ;
007024  020037  0000000       9$:       CMP     R0,VQUN1B       ;Time to shuffle queue?
007030  101422                           BLOS    3$              ;Br if not
007032  004037  0000000                  JSR     R0,QUNSIG       ;Signal that QUAN1B expired
007036  0000000                          .WORD   $SGQ1B
                                    ;
                                    ;  Requeue job at tail of current execution queue
                                    ;
007040  016100  0000000                  MOV     LSTATE(R1),R0   ;Get job's execution state
007044  004737  0000000       7$:       CALL    ENQTL           ;Requeue job at tail of execution queue
007050  005061  0000000                  CLR     LQUAN(R1)       ;Give job a fresh time quantum
007054  000410                           BR      3$
                                    ;
                                    ;  Job is not in high-priority state.
                                    ;  Schedule jobs on quan2 basis.
                                    ;
007056  020037  0000000       4$:       CMP     R0,VQUAN2       ;HAS JOB USED UP QUAN2 UNITS OF TIME?
007062  101405                           BLOS    3$              ;BR IF NOT -- DON'T RESCHEDULE JOB YET
007064  004037  0000000                  JSR     R0,QUNSIG       ;Signal that QUAN2 expired
```

```
    115 007070  0000000                         .WORD   $SGQ2
    116                                 ;
    117                                 ;  Reschedule job in CPU-bound run state.
    118                                 ;
    119 007072  004737  0000000         6$:     CALL    QCPU            ;RESCHEDULE JOB IN CPU-BOUND STATE
    120                                 ;
    121                                 ;  Finished
    122                                 ;
    123 007076  012602                  3$:     MOV     (SP)+,R2
    124 007100  012601                          MOV     (SP)+,R1
    125 007102  000207                          RETURN
```

```
                                        .SBTTL  CLKO1S -- 0.1 second clock processing
                                ;-------------------------------------------------------------------
                                ;  CLKO1S is the timer called every 0.1 seconds to do processing
                                ;  at this frequency.
                                ;
007104  010246                  CLKO1S: MOV     R2,-(SP)
007106  010346                          MOV     R3,-(SP)
                                ;
                                ;  Get # 0.1 second units that have elapsed since the last time we
                                ;  were called.
                                ;
007110  013703  001342'          16$:    MOV     TIKO1S,R3       ;Get tick counter
007114  005203                          INC     R3              ;Actual time = counter + 1
                                ;
                                ;  See if any jobs need to be restarted because they were waiting for
                                ;  a free message buffer and one was freed.
                                ;
007116  005737  000000G                 TST     MBFFLG          ;Were any message buffers freed?
007122  001406                          BEQ     15$             ;Br if not
007124  005037  000000G                 CLR     MBFFLG          ;Reset message-buffer-freed flag
007130  012700  000000G                 MOV     #S$WSMB,R0      ;Restart any jobs that are
007134  004737  000000G                 CALL    UREGO           ; waiting for message buffers
                                ;
                                ;  Decrement minimum core residency time for jobs in memory.
                                ;
007140  012702  000000G          15$:    MOV     #LSTSL,R2       ;GET # OF LAST JOB
007144  032762  000000G 000000G 13$:    BIT     #$INCOR,LSW(R2) ;IS JOB IN MEMORY NOW?
007152  001415                          BEQ     14$             ;BR IF NOT
007154  005762  000000G                 TST     LMINQ(R2)       ;HAS ITS MIN CORE TIME ALREADY EXPIRED?
007160  001412                          BEQ     14$             ;BR IF YES -- DON'T MAKE IT GO NEGATIVE
007162  005362  000000G                 DEC     LMINQ(R2)       ;DEC MIN CORE TIME REMAINING FOR JOB
007166  001007                          BNE     14$             ;BR IF MIN CORE TIME DID NOT EXPIRE
007170  105737  000000G                 TSTB    SWPCOT          ;DOES SCHEDULER WANT TO BE CALLED?
007174  001404                          BEQ     14$             ;BR IF NOT
007176  105237  000000G                 INCB    DOSCHD          ;REQUEST A JOB SCHEDULER CYCLE
007202  105037  000000G                 CLRB    SWPCOT          ;CLEAR MIN-TIME SCHEDULER REQUEST
007206  162702  000002          14$:    SUB     #2,R2           ;CHECK NEXT LINE
007212  001354                          BNE     13$
                                ;
                                ;  Keep track of number of minutes of uptime for system.
                                ;
007214  160337  000000G                 SUB     R3,MINCTR       ;HAS 1 MINUTE PASSED?
007220  003013                          BGT     1$              ;BR IF NOT
007222  062737  001130  000000G         ADD     #600.,MINCTR    ;RESET COUNTER
007230  005237  000000G                 INC     MINTIM          ;COUNT # MINUTES OF SYSTEM UPTIME
007234  005737  000000G                 TST     DTLX            ;IS THIS A DEMO VERSION OF THE SYSTEM?
007240  001403                          BEQ     1$              ;BR IF NOT
007242  005337  000000G                 DEC     DTLX            ;HAS DEMO TIME LIMIT EXPIRED?
007246  001537                          BEQ     99$             ;BR IF DEMO TIME LIMIT REACHED
                                ;
                                ;  Keep track of user/idle/swap-wait time
                                ;
007250  010302                  1$:     MOV     R3,R2           ;Get timer ticks
007252  006302                          ASL     R2              ;Count 2 time units per interval
007254  060237  000000G                 ADD     R2,TMTOTL       ;COUNT TOTAL TIME
007260  005537  000000G                 ADC     TMTOTH          ;PROPOGATE CARRY
007264  005737  000000G                 TST     UIOCNT          ;IS ANY USER I/O IN PROGRESS NOW?
```

```
                                        BEQ     7$              ;BR IF NOT
                                        ADD     R2,TMIOL        ;COUNT TIME THAT USER I/O IS ACTIVE
                                        ADC     TMIOH
                                7$:     TSTB    OUTBSY          ;IS OUTSWAP IN PROGRESS?
                                        BNE     8$              ;BR IF YES
                                        TSTB    INBSY           ;IS INSWAP IN PROGRESS?
                                        BEQ     9$              ;BR IF NOT
                                8$:     ADD     R2,TMSWPL       ;COUNT TIME SWAP IS IN PROGRESS
                                        ADC     TMSWPH
                                9$:     TSTB    CORUSR          ;IS A USER JOB IN EXECUTION NOW?
                                        BEQ     2$              ;BR IF NOT
                                        ADD     R2,TMUSRL       ;COUNT TIME FOR USER JOB EXECUTION
                                        ADC     TMUSRH
                                        BR      3$
                                ;  No user is running.
                                ;  See if we should count time as swap-wait, i/o-wait or idle.
                                2$:     TSTB    OUTBSY          ;IS AN OUTSWAP IN PROGRESS?
                                        BNE     4$              ;BR IF YES
                                        TSTB    INBSY           ;IS AN INSWAP IN PROGRESS?
                                        BEQ     10$             ;BR IF NOT
                                ;  Swapping is in progress.  See if user I/O is also going on.
                                4$:     TST     UIOCNT          ;IS USER I/O IN PROGRESS?
                                        BEQ     11$             ;BR IF NOT
                                        ASR     R2              ;SPLIT TIME BETWEEN SWAP-WAIT AND I/O-WAIT
                                        ADD     R2,TMIOWL       ;CHARGE TO I/O-WAIT
                                        ADC     TMIOWH
                                11$:    ADD     R2,TMSWTL       ;CHARGE TO SWAP-WAIT
                                        ADC     TMSWTH
                                        BR      3$
                                ;  No swapping going on.  See if user I/O is in progress.
                                10$:    TST     UIOCNT          ;IS USER I/O IN PROGRESS?
                                        BEQ     12$             ;BR IF NOT
                                        ADD     R2,TMIOWL       ;CHARGE TO I/O-WAIT
                                        ADC     TMIOWH
                                        BR      3$
                                ;  System is idle.
                                12$:    ADD     R2,TMIDLL       ;CHARGE TO IDLE TIME
                                        ADC     TMIDLH
                                ;
                                ;  Check for time-slice expiration of current job
                                ;
                                3$:     CALL    CLKJOB          ;CHECK FOR TIME-SLICE EXPIRATION OF CUR JOB
                                ;
                                ;  Check to see if we need to cancel I/O hold flag for any jobs
                                ;
                                        CALL    CLKIOH          ;Check for I/O hold flags
                                ;
                                ;  Check for processing needed for autobaud logic
                                ;
                                        CALL    CLKABD          ;Check for autobaud timer processing
                                ;
                                ;  Processing done with 0.5 second frequency.
                                ;
                                        SUB     R3,TK5CNT       ;Has 0.5 seconds passed?
                                        BGT     6$              ;BR IF NOT
                                        ADD     #5.,TK5CNT      ;RESET TIMER
                                        CALL    TLCHK           ;DO TIMED CHECKS ON TIMESHARING LINES
```

```
    115 007500  004737  010014'                 CALL    WAKEUP          ;SEE IF WE NEED TO WAKE UP SLEEPING JOBS
    116 007504  004737  007660'                 CALL    CHKPRT          ;See if we need to print professional screen
    117 007510  005727  000000G                 TST     #CLTOTL         ;Are there any CL lines?
    118 007514  001404                          BEQ     6$              ;Br if not
    119 007516  005237  000000G                 INC     NEDCDO          ;Say output character processing needed
    120 007522  005237  000000G                 INC     NEDCLO          ;Trigger CL clock-driven processing
    121                                 ;
    122                                 ;  See if any more 0.1 second time units passed while we were working
    123                                 ;
    124 007526  160337  001342'         6$:     SUB     R3,TIK01S       ;Have any more 0.1 second intervals passed?
    125 007532  002402                          BLT     17$             ;Br if not
    126 007534  000137  007110'                 JMP     16$             ;Go back and process again
    127                                 ;
    128                                 ;  Finished
    129                                 ;
    130 007540  012603                  17$:    MOV     (SP)+,R3
    131 007542  012602                          MOV     (SP)+,R2
    132 007544  000207                          RETURN
    133                                 ;
    134                                 ;  Time limit has expired on demo version of TSX-Plus.
    135                                 ;  Kill the system.
    136                                 ;
    137 007546                          99$:    DIE     #EM$DTL         ;SYSTEM CRASH -- DEMO TIME LIMIT REACHED
```

```
                                        .SBTTL  CLKIOH -- See if we need to cancel I/O hold timers
                                ;-------------------------------------------------------------
                                ;  This routine is called every 0.1 second to see if we should cancel
                                ;  the I/O hold timers for any jobs.  The I/O hold timer is set when we
                                ;  want to swap a job out of memory but the job has I/O in progress.
                                ;  To avoid holding the I/O for a job forever, we release the I/O hold
                                ;  after a certain period of time (IOHLTM) if a swap has not taken place.
                                ;
007560  010146                  CLKIOH: MOV     R1,-(SP)
                                ;
                                ;  Begin loop to check I/O hold time for each job
                                ;
007562  012701  000000G                 MOV     #LSTSL,R1       ;Get index to last job
                                ;
                                ;  See if I/O hold flag is set for this job
                                ;
007566  005761  000000G         1$:     TST     LIOHLD(R1)      ;Is I/O hold flag set for job?
007572  001425                          BEQ     2$              ;Br if not
                                ;
                                ;  Decrement the remaining I/O hold time
                                ;
007574  005361  000000G                 DEC     LIOHLD(R1)      ;Less I/O hold time left
007600  003022                          BGT     2$              ;Br if some time left
                                ;
                                ;  We just cancelled the I/O hold time for this job.
                                ;  If the job is in a wait state, restart it.
                                ;
007602  026127  000000G 000000G         CMP     LSTATE(R1),#S$IOWT ;Is job in I/O wait state?
007610  001003                          BNE     3$              ;Br if not
007612  004737  000000G                 CALL    FORCEX          ;Start the job running
007616  000413                          BR      2$
                                ;
                                ;  If the job has any pending completion routines, make sure the job
                                ;  priority is at least as high as that of the 1st completion routine.
                                ;  This is necessary since we held off user completion routines while
                                ;  we were waiting for I/O to stop.
                                ;
007620  016100  000000G         3$:     MOV     LCMPL(R1),R0    ;Does job have any pending compl routines?
007624  001410                          BEQ     2$              ;Br if not
007626  126160  000000G 000000G         CMPB    LSTATE(R1),CQ$RNS(R0);Is job priority high as cpl rtn prio?
007634  101404                          BLOS    2$              ;Br if yes
007636  116000  000000G                 MOVB    CQ$RNS(R0),R0   ;Get job state for compl routine
007642  004737  000000G                 CALL    ENQTL           ;Change job state
                                ;
                                ;  Process next job
                                ;
007646  162701  000002          2$:     SUB     #2,R1           ;Get index of next job
007652  003345                          BGT     1$              ;Loop if more jobs to check
                                ;
                                ;  Finished
                                ;
007654  012601                          MOV     (SP)+,R1
007656  000207                          RETURN
```

```
                                        .SBTTL  CHKPRT -- See if we need to print Professional screen
                                ;-------------------------------------------------------------------
                                ;  CHKPRT is called to see if the PI handler has requested that the
                                ;  contents of the professional screen be printed.  If so, an asynchronous
                                ;  completion routine in the PROPRT program is triggered.
                                ;
007660  010246                  CHKPRT: MOV     R2,-(SP)
007662  010446                          MOV     R4,-(SP)
                                ;
                                ;  Return immediately if we are not running on a professional
                                ;
007664  105737  0000000                 TSTB    PROFLG          ;Are we running on a Professional?
007670  001446                          BEQ     9$              ;Br if not
                                ;
                                ;  See if the PROPRT program is running
                                ;
007672  012702  0000000                 MOV     #LSTSL,R2       ;Get index to last line
007676  026237  0000000 001344' 1$:      CMP     LPRG1(R2),R50PRO;1st 3 chars of name = "PRO"?
007704  001004                          BNE     2$              ;Br if not
007706  026237  0000000 001346'          CMP     LPRG2(R2),R50PRT;2nd 3 chars of name = "PRT"?
007714  001404                          BEQ     3$              ;Br if found program
007716  162702  000002          2$:     SUB     #2,R2           ;More lines to check?
007722  003365                          BGT     1$              ;Loop if yes
007724  000425                          BR      4$              ;PROPRT program is not running
                                ;
                                ;  The PROPRT program is running.  See if it has scheduled a
                                ;  completion routine.
                                ;
007726  005762  0000000         3$:     TST     LBRKCQ(R2)      ;Did it specify a completion routine?
007732  001422                          BEQ     4$              ;Br if not
007734  052737  0000000 0000000         BIS     #SS$RUN,SPSTAT  ;Set flag saying spooler is running
                                ;
                                ;  See if PI handler requested that screen be printed
                                ;
007742  032737  0000000 0000000         BIT     #SS$PRT,SPSTAT  ;Did PI request that screen be printed?
007750  001416                          BEQ     9$              ;Br if not
                                ;
                                ;  Trigger completion routine in PROPRT
                                ;
007752  016204  0000000                 MOV     LBRKCQ(R2),R4   ;Get address of completion queue element
007756  001404                          BEQ     5$
007760  005062  0000000                 CLR     LBRKCQ(R2)      ;Say completion Q element used up
007764  004737  0000000                 CALL    QCOMPL          ;Queue completion routine for the job
007770  042737  000000C 0000000 5$:     BIC     #SS$PRT!SS$RUN,SPSTAT ;Clear print-screen flag
007776  000403                          BR      9$
                                ;
                                ;  The PROPRT program is not running
                                ;
010000  042737  000000C 0000000 4$:     BIC     #SS$RUN!SS$PRT,SPSTAT ;Say program not running
                                ;
                                ;  Finished
                                ;
010006  012604                  9$:     MOV     (SP)+,R4
010010  012602                          MOV     (SP)+,R2
010012  000207                          RETURN
```

```
                                        .SBTTL  WAKEUP -- 0.5 second processing for sleeping users
                                ;-----------------------------------------------------------------------
                                ;  Timer routine called ever 0.5 seconds to see if sleeping users
                                ;  need to be awaken.
                                ;
010014  010146                  WAKEUP: MOV     R1,-(SP)
010016  012701  0000000                 MOV     #LSTSL,R1       ;GET INDEX TO LAST LINE
                                ;
                                ;  Check for jobs that need to have TT reads timed out
                                ;
010022  026127  0000000 0000000 1$:     CMP     LSTATE(R1),#S$INWT;IS JOB WAITING FOR TT INPUT?
010030  001015                          BNE     2$              ;BR IF NOT
010032  005761  0000000                 TST     LRDTIM(R1)      ;DOES JOB HAVE A TT READ TIME VALUE SPECIFIED?
010036  001412                          BEQ     2$              ;BR IF NOT
010040  005361  0000000                 DEC     LRDTIM(R1)      ;HAS TIME EXPIRED?
010044  001007                          BNE     2$              ;BR IF NOT
010046  010546                          MOV     R5,-(SP)
010050  016105  0000000                 MOV     LRTCHR(R1),R5   ;GET TIME-OUT ACTIVATION CHARACTER
010054                                  OCALL   STRACT          ;STORE ACTIVATION CHARACTER
010062  012605                          MOV     (SP)+,R5
                                ;
                                ;  Check next line
                                ;
010064  162701  000002          2$:     SUB     #2,R1           ;GET NEXT LINE INDEX
010070  001354                          BNE     1$              ;BR IF THERE IS ANOTHER LINE TO CHECK
                                ;
                                ;  Finished
                                ;
010072  012601                  MOV     (SP)+,R1
010074  000207                  RETURN
```

```
                                        .SBTTL  CKTWAT -- Check on jobs doing .TWAIT waits
                                ;-----------------------------------------------------------------------
                                ;  CKTWAT is called every clock tick to see if any jobs doing .TWAIT waits
                                ;  need to be restarted.
                                ;
010076  010146                  CKTWAT: MOV     R1,-(SP)
                                ;
                                ;  Check for jobs doing timed waits (.twait)
                                ;
                                        MOV     #LSTSL,R1       ;GET HIGHEST JOB INDEX NUMBER
010104  026127  000000  000000  4$:     CMP     LSTATE(R1),#S$TMWT;IS THIS JOB DOING A TIMED WAIT?
010112  001024                          BNE     5$              ;BR IF NOT
010114  163761  000000  000000          SUB     CLKCNT,LSLEPL(R1);DEC SLEEP TIME
010122  005661  000000                  SBC     LSLEPH(R1)      ;PROPOGATE CARRY
010126  002404                          BLT     6$              ;BR IF COUNT WENT NEGATIVE
010130  001015                          BNE     5$              ;BR IF GREATER THAN ZERO
010132  005761  000000                  TST     LSLEPL(R1)      ;CHECK LOW-ORDER VALUE
010136  001012                          BNE     5$              ;BR IF NOT ZERO
                                ;
                                ;  Timed wait is completed.
                                ;  Put job in high priority execution state.
                                ;
010140  005761  000000          6$:     TST     LITIME(R1)      ;IS THIS AN INTERACTIVE JOB?
010144  001403                          BEQ     11$             ;BR IF NOT
010146  012700  000000                  MOV     #S$HICP,R0      ;PUT JOB IN INTERACTIVE CPU STATE
010152  000402                          BR      10$
010154  012700  000000          11$:    MOV     #S$TWFN,R0      ;PUT JOB IN NORMAL HIGH-PRIO EXECUTION STATE
010160  004737  000000          10$:    CALL    ENQTL
010164  162701  000002          5$:     SUB     #2,R1           ;MORE JOBS TO CHECK?
010170  001345                          BNE     4$              ;BR IF YES
                                ;
                                ;  Finished
                                ;
010172  012601                          MOV     (SP)+,R1
010174  000207                          RETURN
```

```
                                        .SBTTL  CKMRKT -- check mark-time requests
                                ;-----------------------------------------------------------------------
                                ;  CKMRKT is called every clock tick to see if any mark-time requests have
                                ;  reached their specified time to be triggered.
                                ;
010176  010146                 CKMRKT: MOV     R1,-(SP)
010200  010246                         MOV     R2,-(SP)
010202  010446                         MOV     R4,-(SP)
                                ;
                                ;  Check for pending mark-time requests
                                ;
010204  005037  001336'                 CLR     FORKIT          ;Clear fork request flag
010210  012702  000000C                 MOV     #MRKTHD-CQ$LNK,R2;FAKE POINTER TO QUEUE HEAD
010214                                  DISABL                  ;** Disable interrupts **
010222  016204  0000000         1$:     MOV     CQ$LNK(R2),R4   ;;;GET ADDRESS OF NEXT ELEMENT IN LIST
010226  001442                         BEQ     8$              ;;;BR IF END OF LIST REACHED
                                ;
                                ;  Subtract time that has past from specified mark-time interval.
                                ;
010230  163764  0000000 0000000         SUB     CLKCNT,CQ$LOT(R4);;;SUBTRACT FROM LOW-ORDER TIME VALUE
010236  005664  0000000                 SBC     CQ$HOT(R4)      ;;;PROPOGATE BORROW TO HIGH-ORDER VALUE
010242  002406                         BLT     3$              ;;;BR IF TIME WENT NEGATIVE
010244  001003                         BNE     2$              ;;;BR IF TIME STILL POSITIVE
010246  005764  0000000                 TST     CQ$LOT(R4)      ;;;CHECK LOW-ORDER VALUE
010252  001402                         BEQ     3$              ;;;Br if zero (time has elapsed)
010254  010402                 2$:     MOV     R4,R2           ;;;Chain forward to next entry in list
010256  000761                         BR      1$
                                ;
                                ;  This mark-time request has expired.
                                ;  Remove the mark-time request entry from the waiting list.
                                ;
010260  016462  0000000 0000000 3$:     MOV     CQ$LNK(R4),CQ$LNK(R2);;;Remove from pending mark-time chain
                                ;
                                ;  Put request on list of pending requests and schedule a lower-priority
                                ;  fork routine to actually execute the completion routine.
                                ;
010266  005064  0000000                 CLR     CQ$LNK(R4)      ;;;Clear forward link in completed element
010272  013700  0000000                 MOV     SYPNCR,R0       ;;;Get address of 1st pending compl request
010276  001005                         BNE     9$              ;;;Br if there are pending requests
                                ;
                                ;  First entry of completion requires a fork process to be executed.
                                ;
010300  010437  0000000         5$:     MOV     R4,SYPNCR       ;;;Set us as 1st pending compl routine
010304  010437  001336'                 MOV     R4,FORKIT       ;;;Set fork request flag
010310  000744                         BR      1$              ;;;Go check for more finished requests
                                ;
                                ;  Other completion entries exist so add current completion to list tail.
                                ;
010312  005760  0000000         9$:     TST     CQ$LNK(R0)      ;;;Is there another pending request?
010316  001403                         BEQ     6$              ;;;Br if not
010320  016000  0000000                 MOV     CQ$LNK(R0),R0   ;;;Chain forward to next pending request
010324  000772                         BR      9$              ;;;Follow list to end
010326  010460  0000000         6$:     MOV     R4,CQ$LNK(R0)   ;;;Add our entry to end of list
010332  000733                         BR      1$              ;;;Check for more completed requests
                                ;
                                ;  Finished.  Create fork process if needed.
                                ;
```

```
      58 010334                                8$:     ENABL                           ;** Enable interrupts **
      59 010342  005737  001336'                      TST     FORKIT                  ;Check fork request flag
      60 010346  001412                           BEQ     10$                     ;Br if fork is not needed
      61 010350  004737  0000000                      CALL    FRKGET                  ;Get a free fork request block
      62 010354  112764  0000000 0000000             MOVB    #FP$ICF,FQ$PRI(R4);Set fork priority
      63 010362  012764  010404' 0000000             MOV     #CLKSCR,FQ$RTN(R4);Set address of routine to execute
      64 010370  004737  0000000                      CALL    FORKQ                   ;Queue the fork request
      65 010374  012604                        10$:    MOV     (SP)+,R4
      66 010376  012602                           MOV     (SP)+,R2
      67 010400  012601                           MOV     (SP)+,R1
      68 010402  000207                           RETURN
```

```
                                        .SBTTL  CLKSCR -- Execute completed system mark-time requests
                                ;-------------------------------------------------------------------
                                ;  This routine is at a lower-priority clock-driven fork priority
                                ;  to process all completed mark-time completion requests for system
                                ;  routines.
                                ;
010404  010146                  CLKSCR: MOV     R1,-(SP)
010406  010446                          MOV     R4,-(SP)
                                ;
                                ;  Unlink next completed entry from pending list
                                ;
010410                          1$:     DISABL                          ;** Disable interrupts **
010416  013704  0000000                 MOV     SYPNCR,R4               ;;;Get address of next completion block
010422  001433                          BEQ     9$                      ;;;Br if no more pending
010424  016437  0000000 0000000         MOV     CQ$LNK(R4),SYPNCR       ;;;Unlink block from list
010432                                  ENABL                           ;** Enable interrupts **
                                ;
                                ;  See if this mark-time request is for a user job or the system.
                                ;
010440  112764  0000000 0000000         MOVB    #CP$STD,CQ$CP(R4);Set completion routine class priority
010446  116401  0000000                 MOVB    CQ$JOB(R4),R1   ;Get index # of job that did the .MRKT
010452  001414                          BEQ     4$              ;Br if timer request came from the system
                                ;
                                ;  Timer request is for a user job.
                                ;  Call QCOMPL to queue the completion routine for the user job.
                                ;
010454  012700  0000000                 MOV     #S$TWFN,R0      ;Get compl prio for non-interactive jobs
010460  005761  0000000                 TST     LITIME(R1)      ;Is this job interactive?
010464  001402                          BEQ     2$              ;Br if not
010466  012700  0000000                 MOV     #S$HICP,R0      ;Get compl prio for interactive jobs
010472  110064  0000000         2$:     MOVB    R0,CQ$RNS(R4)   ;Set execution state for compl routine
010476  116164  0000000 0000000         MOVB    LPRI(R1),CQ$PRI(R4);Set execution priority value
                                ;
                                ;  Process this completion request
                                ;
010504  004737  0000000         4$:     CALL    QCOMPL          ;Process the completed request
                                ;
                                ;  Go back and see if there are more pending requests
                                ;
010510  000737                          BR      1$
                                ;
                                ;  Finished all pending requests
                                ;
010512                          9$:     ENABL
010520  012604                          MOV     (SP)+,R4
010522  012601                          MOV     (SP)+,R1
010524  000207                          RETURN
```

```
                                        .SBTTL  CLKPM  -- accumulate performance monitoring data
                                ;-------------------------------------------------------------------
                                ;  CLKPM is called to accumulate performance monitoring information.
                                ;
                                ;  Inputs:
                                ;   CLKCNT = Number of clock ticks to charge to job.
                                ;   CLKPC  = PC when clock interrupt occured.
                                ;   CLKPS  = PS when clock interrupt occured.
                                ;   PMUSER = Job number of user who is doing performance analysis.
                                ;   PMBASE = Base address of region being monitored.
                                ;   PMTOP  = Top address of region being monitored.
                                ;   PMFLGS = PF$ control flags
                                ;   LEMTPC(Job) = PC when last EMT was executed for job.
                                ;
                                ;  Outputs:
                                ;   Appropriate cell in performance counter table is incremented.
                                ;
010526  010146                 CLKPM:  MOV     R1,-(SP)
010530  010246                         MOV     R2,-(SP)
010532  010346                         MOV     R3,-(SP)
                                ;
                                ;  See if we are monitoring system execution or user job execution.
                                ;
010534  032737  0000000 0000000         BIT     #PF$SYS,PMFLGS  ;MONITORING SYSTEM OR USER JOB?
010542  001407                         BEQ     4$              ;BR IF MONITORING USER JOB
                                ;  We are monitoring the system.
010544  032737  0000000 0000000         BIT     #UMODE,CLKPS    ;DID INTERRUPT OCCUR IN KERNEL MODE?
010552  001065                         BNE     9$              ;BR IF NOT -- USER WAS RUNNING
010554  013703  0000000               MOV     CLKPC,R3        ;GET ADDRESS WHERE INTERRUPT OCCURED
010560  000432                         BR      3$              ;GO COUNT HIT
                                ;
                                ;  We are monitoring user job execution.
                                ;  Determine if we should count a hit against running job.
                                ;
010562  013701  0000000         4$:     MOV     PMUSER,R1       ;GET # OF JOB BEING MONITORED
010566  032761  0000000 0000000         BIT     #$INKMN,LSW4(R1);IS KMON RUNNING?
010574  001054                         BNE     9$              ;DON'T MONITOR KMON
010576  120137  0000000               CMPB    R1,CORUSR       ;IS JOB BEING MONITORED THE CURRENT JOB?
010602  001007                         BNE     1$              ;BR IF NOT
                                ;  Monitored job is running now.
                                ;  See if interrupt occured in user or kernel mode.
010604  032737  0000000 0000000         BIT     #UMODE,CLKPS    ;DID INTERRUPT OCCUR IN USER OR KERNEL MODE?
010612  001413                         BEQ     2$              ;BR IF IN KERNEL MODE
                                ;  Job was in user mode so use interrupted PC.
010614  013703  0000000               MOV     CLKPC,R3        ;COUNT HIT AGAINST THIS PC
010620  000412                         BR      3$
                                ;  Monitored job is not now running.
                                ;  See if we should charge I/O wait time to job.
010622  032737  0000000 0000000 1$:     BIT     #PF$IOW,PMFLGS  ;SHOULD WE CHARGE FOR I/O WAIT TIME?
010630  001436                         BEQ     9$              ;BR IF NOT
                                ;  See if monitored job is waiting for I/O.
010632  026127  0000000 0000000         CMP     LSTATE(R1),#S$IOWT;IS MONITORED JOB WAITING FOR I/O?
010640  001032                         BNE     9$              ;BR IF  NOT
                                ;  Use last EMT address as cell to charge hit to.
010642  016103  0000000         2$:     MOV     LEMTPC(R1),R3   ;GET ADDRESS OF LAST EMT DONE BY JOB
                                ;
                                ;  At this point we have in R3 the PC that we are to charge this time to.
```

```
                                        ;  See if the PC is in the region being monitored.
                                        ;
010646  020337  0000000G        3$:     CMP     R3,PMBASE       ;IS IT BELOW BASE OF REGION?
010652  103425                          BLO     9$              ;BR IF YES
010654  020337  0000000G              CMP     R3,PMTOP        ;IS IT ABOVE TOP OF REGION?
010660  103022                          BHIS    9$              ;BR IF YES
                                        ;  PC is in region being monitored.
010662  163703  0000000G              SUB     PMBASE,R3       ;SUBTRACT BASE TO GET OFFSET
010666  005002                          CLR     R2              ;SET FOR DIVIDE
010670  071237  0000000G              DIV     PMNBPC,R2       ;DIVIDE BY # BYTES PER CELL
010674  006302                          ASL     R2              ;CONVERT CELL # TO BYTE #
010676  062702  0000000G              ADD     #VPAR6,R2       ;ADD VIRTUAL ADDRESS OF PAR6 REGION
010702  013737  0000000G 0000000G      MOV     PMPAR,@#KPAR6   ;MAP PAR6 TO PM DATA AREA
010710  063712  0000000G              ADD     CLKCNT,(R2)     ;ADD TIME TO COUNTER FOR THIS CELL
010714  103004                          BCC     9$              ;BR IF NO OVERFLOW OF CELL
010716  005312                          DEC     (R2)            ;SET COUNTER VALUE BACK TO -1
010720  052737  0000000G 0000000G      BIS     #PF$OVF,PMFLGS  ;REMEMBER THAN AN OVERFLOW OCCURED
                                        ;
                                        ;  Finished
                                        ;
010726  012603                  9$:     MOV     (SP)+,R3
010730  012602                          MOV     (SP)+,R2
010732  012601                          MOV     (SP)+,R1
010734  000207                          RETURN
```

```
                                        .SBTTL  CKSCHD -- Check jobs and schedule
                                ;-------------------------------------------------------------------
                                ;  CKSCHD will check all the jobs and schedule those that have been flagged
                                ;  as needing a priority boost because of output buffer empty or low.
                                ;
                                ;  Inputs:
                                ;       LSW7 - job scheduling flag
                                ;
010736  010046                  CKSCHD: MOV     R0,-(SP)                ;Save registers
010740  010146                          MOV     R1,-(SP)
010742  012701  0000000                 MOV     #LSTSL,R1               ;Obtain index to the last line
                                ;
                                ;  Check all jobs to see if any need a priority boost for terminal buffer
                                ;  empty or low.
                                ;
010746  032761  0000000 0000000 1$:     BIT     #$OTFN,LSW7(R1)         ;Check for scheduling flag enable
010754  001417                          BEQ     10$                     ;Br if no scheduling required
010756  042761  0000000 0000000         BIC     #$OTFN,LSW7(R1)         ;Reset job scheduling flag
010764  012700  0000000                 MOV     #S$OTFN,R0              ;Get output-buffer empty state
010770  005761  0000000                 TST     LITIME(R1)              ;Is this an interactive job?
010774  001002                          BNE     2$                      ;Br if yes
010776  012700  0000000                 MOV     #S$OTLO,R0              ;If not interactive then use lower pri
011002  026100  0000000         2$:     CMP     LSTATE(R1),R0           ;Is job already in this prio or better
011006  101402                          BLOS    10$                     ;Br if yes
011010  004737  0000000                 CALL    ENQTL                   ;Queue job at tail of execution list
                                ;
                                ;  Check the next line.
                                ;
011014  162701  000002          10$:    SUB     #2,R1                   ;Check the next job
011020  003352                          BGT     1$                      ;Continue until all job tested
011022  012601                          MOV     (SP)+,R1                ;Restore registers
011024  012600                          MOV     (SP)+,R0
011026  000207                          RETURN
```

```
                                        .SBTTL  CLKABD -- Clock processing for autobaud logic
                                ;-----------------------------------------------------------------------
                                ;  CLKABD is called on a 1/10 second basis to do clock driven processing
                                ;  related to autobaud logic.
                                ;
011030  010146                 CLKABD: MOV     R1,-(SP)
                                ;
                                ;  Begin loop to check each line
                                ;
011032  012701  0000000                 MOV     #LSTPL,R1       ;Get index # of last line
                                ;
                                ;  See if this line has autobaud control
                                ;
011036  032761  0000000 0000000 1$:     BIT     #$AUTO,ILSW2(R1);Does this line have autobaud control?
011044  001421                          BEQ     2$              ;Br if not
                                ;
                                ;  Decrement autobaud timer for this line
                                ;
011046  005761  0000000                 TST     LABTIM(R1)      ;Is the autobaud timer active?
011052  001416                          BEQ     2$              ;Br if not
011054  005361  0000000                 DEC     LABTIM(R1)      ;Decrement timer
011060  001013                          BNE     2$              ;Br if timer did not time out
                                ;
                                ;  Timer timed out for this line.
                                ;  See if we need to reset the line speed.
                                ;
011062  032761  0000000 0000000         BIT     #$NABRS,LSW9(R1);Do we need to reset the line speed?
011070  001407                          BEQ     2$              ;Br if not
011072  042761  0000000 0000000         BIC     #$NABRS,LSW9(R1);Clear the flag
011100  012700  0000000                 MOV     #S9600,R0       ;Reset line speed to 9600 baud
011104  004737  0000000                 CALL    SETSPD          ;Reset speed
                                ;
                                ;  Check next line
                                ;
011110  162701  000002          2$:     SUB     #2,R1           ;More lines to do?
011114  003350                          BGT     1$              ;Loop if yes
                                ;
                                ;  Finished
                                ;
011116  012601                          MOV     (SP)+,R1
011120  000207                          RETURN
```

```
                                        .SBTTL  TLCHK  -- Check Dial-up line status
                                ;-----------------------------------------------------------------
                                ;  TLCHK is called from the clock interrupt routine every 0.5 seconds
                                ;  to see if dial-up lines need to be answered or hung up.
                                ;
011122  010146                  TLCHK:  MOV     R1,-(SP)
011124  010246                          MOV     R2,-(SP)
                                ;
                                ;  Begin loop to check each physical line
                                ;
011126  012701  000000G                 MOV     #LSTHL,R1       ;Index to last real line
                                ;
                                ;  See if this line is installed
                                ;
011132  032761  000000G 000000G 1$:     BIT     #$DEAD,LSW3(R1) ;Is this line installed?
011140  001024                          BNE     2$              ;Br if not
011142  032761  000000G 000000G         BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
011150  001420                          BEQ     2$              ;Br if not
                                ;
                                ;  Call processing routine based on type of communications device
                                ;
011152  016102  000000G                 MOV     LCDTYP(R1),R2   ;Get comm device index number
011156  004772  000000G                 CALL    @CDCLOK(R2)     ;Call processing routine for this line
                                ;
                                ;  If this is a dial-up line, check on line ringing, lost carrier, etc.
                                ;
011162  005761  000000G                 TST     LCLUNT(R1)      ;Is this line in use as a CL unit?
011166  002011                          BGE     2$              ;Br if this is a CL line
011170  032761  000000G 000000G         BIT     #$PHONE,ILSW2(R1);Is this a dial-up line?
011176  001405                          BEQ     2$              ;Br if not
011200  020127  000000G                 CMP     R1,#LSTPL       ;Is this a time-sharing or CL line?
011204  101002                          BHI     2$              ;Don't do phone checks for CL lines
011206  004737  011226'                 CALL    CLKPHN          ;Check phone line
                                ;
                                ;  See if there are more lines to be checked
                                ;
011212  162701  000002          2$:     SUB     #2,R1           ;Get index number for next line
011216  001345                          BNE     1$              ;Loop if more lines to check
                                ;
                                ;  Finished
                                ;
011220  012602                          MOV     (SP)+,R2
011222  012601                          MOV     (SP)+,R1
011224  000207                          RETURN
```

```
                                        .SBTTL  CLKPHN -- Do timer driven checks of dial-up lines
                                ;-----------------------------------------------------------------------
                                ;  CLKPHN is called periodically to perform checks on dial-up lines.
                                ;  Checks are made to see if the phone is ringing or if carrier
                                ;  has been detected or lost.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;
011226  010246                  CLKPHN: MOV     R2,-(SP)
                                ;
                                ;  Call device-dependent routine to get the data set status for this line
                                ;
011230  016102  0000000                 MOV     LCDTYP(R1),R2   ;Get comm device type code index
011234  004772  0000000                 CALL    @CDGDSS(R2)     ;Get data set status
                                ;
                                ;  At this point, the generic modem status (MS$xxx flags) is in R0.
                                ;  See if the phone is ringing
                                ;
011240  105737  0000000                 TSTB    STPFLG          ;Is system being stopped?
011244  001020                          BNE     5$              ;Br if yes
011246  032700  0000000                 BIT     #MS$RNG,R0      ;Is line ringing?
011252  001415                          BEQ     5$              ;Br if not
011254  032700  0000000                 BIT     #MS$DTR,R0      ;Have we enabled answer?
011260  001012                          BNE     5$              ;Br if yes
011262  052700  0000000                 BIS     #MS$DTR,R0      ;Enable answer
011266  004772  0000000                 CALL    @CDSDSS(R2)     ;Set data set status
011272  013761  0000000 0000000         MOV     VTMOUT,LCDTIM(R1);Start carrier-down timer
011300  013761  0000000 0000000         MOV     VOFFTM,LOFFTM(R1);Drop DTR if not logged on by this time
                                ;
                                ;  Check status of carrier on dial-up lines.
                                ;
011306  032700  0000000         5$:     BIT     #MS$CAR,R0      ;Is carrier up or down?
011312  001054                          BNE     16$             ;Br if up
                                ;
                                ;  Carrier is down
                                ;
011314  005361  0000000         11$:    DEC     LCDTIM(R1)      ;Has it been down very long?
011320  003057                          BGT     8$              ;Br if not
011322  042761  0000000 0000000         BIC     #$CARUP,LSW3(R1);Remember that we have lost carrier
011330  032761  0000000 0000000         BIT     #$DILUP,LSW(R1) ;Is line active?
011336  001407                          BEQ     1$              ;Br if not
011340  105737  0000000                 TSTB    VUSPHN          ;Should we always mon. carrier for line?
011344  001004                          BNE     1$              ;Br if so
011346  032761  0000000 0000000         BIT     #$CARMN,LSW5(R1);Are we monitoring carrier for this line?
011354  001436                          BEQ     17$             ;Br if not
011356  052761  0000000 0000000 1$:     BIS     #$NOIN,LSW3(R1) ;Ignore tt input from the line
011364  032761  0000000 0000000         BIT     #$DILUP,LSW(R1) ;Is line active?
011372  001415                          BEQ     21$             ;Br if not
011374  032761  0000000 0000000         BIT     #<$DISCN+$DOOFF>,LSW(R1);Are we logging off line now?
011402  001026                          BNE     8$              ;Br if yes -- that takes care of it
011404  032761  0000000 0000000         BIT     #$LOFCF,LSW9(R1);Are we doing logoff command file now?
011412  001022                          BNE     8$              ;Br if yes
011414  010100                          MOV     R1,R0           ;Get index of job being aborted
011416                                  @CALL   KILJOB          ;Kill the job
011424  000415                          BR      8$
011426  042700  0000000         21$:    BIC     #MS$DTR,R0      ;Drop data terminal ready (hang up)
```

```
     58 011432  004772  0000000                    CALL    @CDSDSS(R2)     ;Set data set status
     59 011436  005061  0000000                    CLR     LOFFTM(R1)      ;Clear logoff timer
     60 011442  000403                             BR      17$
     61                                 ;
     62                                 ;  Carrier is up
     63                                 ;
     64 011444  052761  0000000 0000000 16$:      BIS     #$CARUP,LSW3(R1);Remember carrier is up
     65                                 ;
     66                                 ;  Reset lost-carrier timer
     67                                 ;
     68 011452  013761  0000000 0000000 17$:      MOV     VTMOUT,LCDTIM(R1);Reset carrier-lost timer
     69                                 ;
     70                                 ;  If jobs on dial-up lines remain in a logged off state for more than
     71                                 ;  a specified interval, drop DTR to hang up on them.
     72                                 ;
     73 011460  005761  0000000         8$:      TST     LOFFTM(R1)      ;Do we need to check time for this job?
     74 011464  001407                           BEQ     6$              ;Br if not
     75 011466  005361  0000000                  DEC     LOFFTM(R1)      ;Is it time to drop DTR for this line?
     76 011472  003004                           BGT     6$              ;Br if not
     77 011474  042700  0000000                  BIC     #MS$DTR,R0      ;Drop data terminal ready (hang up)
     78 011500  004772  0000000                  CALL    @CDSDSS(R2)     ;Set data set status
     79                                 ;
     80                                 ;  Finished
     81                                 ;
     82 011504  012602                  6$:      MOV     (SP)+,R2
     83 011506  000207                           RETURN
```

```
      1                                         .SBTTL  DLGDSS -- Get data set status for DL11 line
      2                                 ;-----------------------------------------------------------------------
      3                                 ;  DLGDSS is called to get the data set status for a DL11 line.
      4                                 ;
      5                                 ;  Inputs:
      6                                 ;   R1 = Physical line index number.
      7                                 ;
      8                                 ;  Outputs:
      9                                 ;   R0 = Generic data set status flags (MS$xxx)
     10                                 ;
     11 011510  010346                  DLGDSS: MOV     R3,-(SP)
     12 011512  005000                          CLR     R0              ;Form result in R0
     13                                 ;
     14                                 ;  Get contents of DL11 receiver status register
     15                                 ;
     16 011514  017103  000000G                 MOV     @RSR(R1),R3     ;Get receiver status register contents
     17                                 ;
     18                                 ;  See if line is ringing
     19                                 ;
     20 011520  032703  000000G                 BIT     #RING,R3        ;Is phone ringing?
     21 011524  001402                          BEQ     1$              ;Br if not
     22 011526  052700  000000G                 BIS     #MS$RNG,R0      ;Set ring flag
     23                                 ;
     24                                 ;  See if carrier is up or down
     25                                 ;
     26 011532  032703  000000G         1$:     BIT     #CARDET,R3      ;Is carrier up or down?
     27 011536  001402                          BEQ     2$              ;Br if down
     28 011540  052700  000000G                 BIS     #MS$CAR,R0      ;Set carrier-up flag
     29                                 ;
     30                                 ;  See if Data Terminal Ready is asserted
     31                                 ;
     32 011544  032703  000000G         2$:     BIT     #TRMRDY,R3      ;Is DTR asserted?
     33 011550  001402                          BEQ     3$              ;Br if not
     34 011552  052700  000000G                 BIS     #MS$DTR,R0      ;Set DTR flag
     35                                 ;
     36                                 ;  Finished
     37                                 ;
     38 011556  012603                  3$:     MOV     (SP)+,R3
     39 011560  000207                          RETURN
```

```
                                        .SBTTL  DLSDSS -- Set data set status for DL11 line
                                ;-----------------------------------------------------------------------
                                ;  DLSDSS is called to set data set control status for a DL11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;   R0 = Control flags (MS$DTR)
                                ;
011562                          DLSDSS:
                                ;
                                ;  See if we should set or drop DTR
                                ;
011562  032700  0000000                 BIT     #MS$DTR,R0      ;Set or drop DTR?
011566  001404                          BEQ     1$              ;Br to drop DTR
                                ;
                                ;  Set DTR
                                ;
011570  052771  0000000 0000000         BIS     #TRMRDY,@RSR(R1);Set Data Terminal Ready
011576  000403                          BR      9$
                                ;
                                ;  Drop DTR
                                ;
011600  042771  0000000 0000000 1$:     BIC     #TRMRDY,@RSR(R1);Drop DTR
                                ;
                                ;  Finished
                                ;
011606  000207                  9$:     RETURN
```

```
                                        .SBTTL  DLSBRK -- Control break transmission for a DL11 line
                                ;-----------------------------------------------------------------------
                                ;  DLSBRK is called to start or stop sending a break character to a DL11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line number.
                                ;   R0 = Break control flag (MS$BRK)
                                ;
011610                          DLSBRK:
                                ;
                                ;  See if we are to start or stop transmitting a break
                                ;
011610  032700  0000000                 BIT     #MS$BRK,R0      ;Start or stop break?
011614  001404                          BEQ     1$              ;Br if stop
                                ;
                                ;  Start transmitting a break
                                ;
011616  052771  0000000 0000000         BIS     #TRBRK,@TSR(R1) ;Start transmitting a break
011624  000403                          BR      9$
                                ;
                                ;  Stop transmitting a break
                                ;
011626  042771  0000000 0000000 1$:     BIC     #TRBRK,@TSR(R1) ;Stop transmitting a break
                                ;
                                ;  Finished
                                ;
011634  000207                  9$:     RETURN
```

```
                                        .SBTTL  DLSSPD -- Set transmission speed for DL11 line
                                ;-----------------------------------------------------------------------
                                ;  DLSSPD is called to set the transmission speed for a DL11 line.
                                ;
                                ;  Inputs:
                                ;   R0 = Speed code.
                                ;   R1 = Physical line index number.
                                ;
011636  010246                  DLSSPD: MOV     R2,-(SP)
                                ;
                                ;  Set speed in DL11 control register
                                ;
011640  110061  0000010                 MOVB    R0,LMXPRM+1(R1) ;Store new code flags for line
011644  010002                          MOV     R0,R2           ;Get speed code
011646  072227  000014                  ASH     #12.,R2         ;Position the speed code
011652  052702  004000                  BIS     #004000,R2      ;Set programmable-baud-rate-enable bit
011656                                  DISABL                  ;;;** Disable interrupts **
011664  017146  0000000                 MOV     @TSR(R1),-(SP)  ;;;Get current transmitter status
011670  042716  170000                  BIC     #170000,(SP)    ;;;Clear the baud rate field
011674  050216                          BIS     R2,(SP)         ;;;Set new baud rate value
011676  012671  0000000                 MOV     (SP)+,@TSR(R1)  ;;;Store new value for transmitter
011702                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Finished
                                ;
011710  012602                          MOV     (SP)+,R2
011712  000207                          RETURN
```

```
                                        .SBTTL  DZGDSS -- Get data set status for DZ11 line
                                ;-----------------------------------------------------------------------
                                ;  DZGDSS is called to get the data set status for a DZ11 line
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;
                                ;  Outputs:
                                ;   R0 = Generic data set status flags (MS$xxx)
                                ;
011714  010246                  DZGDSS: MOV     R2,-(SP)
011716  010346                          MOV     R3,-(SP)
011720  005000                          CLR     R0              ;Build result in R0
                                ;
                                ;  Get DZ11 index number
                                ;
011722  016102  0000000                 MOV     LMXNUM(R1),R2   ;Get DZ11 number
011726  016103  0000000                 MOV     LMXLN(R1),R3    ;Get # of line within DZ11 group (0-7)
011732  116303  001312'                 MOVB    MXLBIT(R3),R3   ;Get line select bit for DZ11 registers
                                ;
                                ;  See if line is ringing
                                ;
011736  130372  0000000                 BITB    R3,@MXRING(R2)  ;Is this line ringing?
011742  001402                          BEQ     1$              ;Br if not
011744  052700  0000000                 BIS     #MS$RNG,R0      ;Set ring flag
                                ;
                                ;  See if carrier is up
                                ;
011750  130372  0000000        1$:      BITB    R3,@MXCAR(R2)   ;Is carrier up or down?
011754  001402                          BEQ     2$              ;Br if down
011756  052700  0000000                 BIS     #MS$CAR,R0      ;Set carrier-up flag
                                ;
                                ;  See if Data Terminal Ready is asserted
                                ;
011762  130372  0000000        2$:      BITB    R3,@MXDTR(R2)   ;Is DTR asserted?
011766  001402                          BEQ     3$              ;Br if not
011770  052700  0000000                 BIS     #MS$DTR,R0      ;Set DTR flag
                                ;
                                ;  Finished
                                ;
011774  012603                  3$:     MOV     (SP)+,R3
011776  012602                          MOV     (SP)+,R2
012000  000207                          RETURN
```

```
                                        .SBTTL  DZSDSS -- Set data set status for a DZ11 line
                                ;-----------------------------------------------------------------------
                                ;  DZSDSS is called to set data set status for a DZ11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line number.
                                ;   R0 = Data set status flags (MS$DTR).
                                ;
012002  010246                  DZSDSS: MOV     R2,-(SP)
012004  010346                          MOV     R3,-(SP)
                                ;
                                ;  Get DZ11 index number
                                ;
012006  016102  0000000                 MOV     LMXNUM(R1),R2   ;Get DZ11 number
012012  016103  0000000                 MOV     LMXLN(R1),R3    ;Get line # within DZ11 (0-7)
012016  116303  001312'                 MOVB    MXLBIT(R3),R3   ;Get line select bit
                                ;
                                ;  See if we should set or drop Data Terminal Ready
                                ;
012022  032700  0000000                 BIT     #MS$DTR,R0      ;Set or drop DTR?
012026  001003                          BNE     1$              ;Br if set DTR
                                ;
                                ;  Drop DTR
                                ;
012030  140372  0000000                 BICB    R3,@MXDTR(R2)   ;Clear DTR flag for our line
012034  000402                          BR      9$
                                ;
                                ;  Set DTR
                                ;
012036  150372  0000000         1$:     BISB    R3,@MXDTR(R2)   ;Set DTR flag for our line
                                ;
                                ;  Finished
                                ;
012042  012603                  9$:     MOV     (SP)+,R3
012044  012602                          MOV     (SP)+,R2
012046  000207                          RETURN
```

```
                                        .SBTTL  DZSBRK -- Control break transmission for a DZ11 line
                                ;-------------------------------------------------------------------
                                ;  DZSBRK is called to start or stop transmitting a break character
                                ;  to a DZ11 line.
                                ;
                                ;  Inputs:
                                ;   R0 = Break control flag (MS$BRK)
                                ;   R1 = Physical line index number.
                                ;
012050  010246                  DZSBRK: MOV     R2,-(SP)
012052  010346                          MOV     R3,-(SP)
012054  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get DZ11 index number
                                ;
012056  016102  000000G                 MOV     LMXNUM(R1),R2   ;Get DZ11 number
012062  016103  000000G                 MOV     LMXLN(R1),R3    ;Get line # within DZ11 (0-7)
012066  116303  001312'                  MOVB    MXLBIT(R3),R3   ;Get line select bit
                                ;
                                ;  We keep a "shadow" copy of the break register in memory since we
                                ;  cannot read the status of the hardware break register.
                                ;
012072  116204  000000G                 MOVB    MXSBRK(R2),R4   ;Get contents of shadow register
012076  140304                          BICB    R3,R4           ;Assume we want to stop sending break
012100  032700  000000G                 BIT     #MS$BRK,R0      ;Do we want to start sending break?
012104  001401                          BEQ     1$              ;Br if not
012106  150304                          BISB    R3,R4           ;Set break flag for the line
                                ;
                                ;  Set new break control flags in hardware register and shadow register
                                ;
012110  110472  000000G         1$:     MOVB    R4,@MXBRK(R2)   ;Set status in hardware register
012114  110462  000000G                 MOVB    R4,MXSBRK(R2)   ;Update shadow register
                                ;
                                ;  Finished
                                ;
012120  012604                  9$:     MOV     (SP)+,R4
012122  012603                          MOV     (SP)+,R3
012124  012602                          MOV     (SP)+,R2
012126  000207                          RETURN
```

```
                                        .SBTTL  DZSSPD -- Set transmission speed for a DZ11 line
                                ;-----------------------------------------------------------------
                                ;  DZSSPD is called to set the transmit/receive speed for a DZ11 line.
                                ;
                                ;  Inputs:
                                ;   R0 = Speed code.
                                ;   R1 = Physical line index number.
                                ;
012130  010346                  DZSSPD: MOV     R3,-(SP)
                                ;
                                ;  Build line parameter register value
                                ;
012132  110061  0000010                 MOVB    R0,LMXPRM+1(R1) ;Save new parameter flags
012136  010003                          MOV     R0,R3           ;Get speed code
012140  042703  000000C                 BIC     #^C<LP$SPD>,R3  ;Clear all but speed code
012144  000303                          SWAB    R3              ;Position speed code to match LPR field
012146  032700  0000000                 BIT     #LP$7BT,R0      ;Are 7 bit characters wanted?
012152  001003                          BNE     1$              ;Br if yes
012154  052703  0000000                 BIS     #DZ$8BT,R3      ;Select 8 bit characters
012160  000402                          BR      2$
012162  052703  0000000          1$:    BIS     #DZ$7BT,R3      ;Select 7 bit characters
012166  032700  0000000          2$:    BIT     #LP$PAR,R0      ;Is parity wanted?
012172  001407                          BEQ     3$              ;Br if not
012174  052703  0000000                 BIS     #DZ$PAR,R3      ;Enable parity
012200  032700  0000000                 BIT     #LP$ODD,R0      ;Is odd parity wanted?
012204  001402                          BEQ     3$              ;Br if not
012206  052703  0000000                 BIS     #DZ$ODD,R3      ;Select odd parity
                                ;
                                ;  Store LPR value for line
                                ;
012212  052703  010000           3$:    BIS     #10000,R3       ;Set receiver-on flag
012216  056103  0000000                 BIS     LMXLN(R1),R3    ;Get line within mux (0-7)
012222  016100  0000000                 MOV     LMXNUM(R1),R0   ;Get DZ11 number
012226  010370  0000000                 MOV     R3,@MXLPR(R0)   ;Set speed for the line
                                ;
                                ;  Finished
                                ;
012232  012603                          MOV     (SP)+,R3
012234  000207                          RETURN
```

```
                                        .SBTTL  DHGDSS -- Get data set status for a DH11 line
                                ;-----------------------------------------------------------------------
                                ;  DHGDSS is called to get the data set status for a DH11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line number
                                ;
                                ;  Outputs:
                                ;   R0 = Generic modem status flags (MS$xxx)
                                ;
012236  010246                  DHGDSS: MOV     R2,-(SP)
012240  010346                          MOV     R3,-(SP)
012242  005000                          CLR     R0              ;Build result in R0
                                ;
                                ;  Get DH11 index number
                                ;
012244  016102  0000000                 MOV     LMXNUM(R1),R2   ;Get DH11 index number
012250  016103  0000000                 MOV     LMXLN(R1),R3    ;Get line within DH11 (0-15)
                                ;
                                ;  Get modem status
                                ;
012254                                  DISABL                  ;;;** Disable interrupts **
012262  042772  0000000 0000000        BIC     #MF$LIN,@DM$CSR(R2) ;;;Clear DM11 line select field
012270  050372  0000000                 BIS     R3,@DM$CSR(R2)  ;;;Select line
012274  017203  0000000                 MOV     @DM$LSR(R2),R3  ;;;Get line status value
012300                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  See if phone is ringing
                                ;
012306  032703  0000000                 BIT     #MF$RNG,R3      ;Is the phone ringing?
012312  001402                          BEQ     1$              ;Br if not
012314  052700  0000000                 BIS     #MS$RNG,R0      ;Set ring flag
                                ;
                                ;  See if carrier is detected
                                ;
012320  032703  0000000         1$:     BIT     #MF$CAR,R3      ;Is carrier detected?
012324  001402                          BEQ     2$              ;Br if not
012326  052700  0000000                 BIS     #MS$CAR,R0      ;Set carrier flag
                                ;
                                ;  See if Data Terminal Ready is asserted
                                ;
012332  032703  0000000         2$:     BIT     #MF$DTR,R3      ;Is DTR asserted?
012336  001402                          BEQ     3$              ;Br if not
012340  052700  0000000                 BIS     #MS$DTR,R0      ;Set DTR flag
                                ;
                                ;  Finished
                                ;
012344  012603                  3$:     MOV     (SP)+,R3
012346  012602                          MOV     (SP)+,R2
012350  000207                          RETURN
```

```
                                        .SBTTL  DHSDSS -- Set data set status for a DH11 line
                                ;-----------------------------------------------------------------------
                                ;  DHSDSS is called to set the data set status for a DH11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;   R0 = Data set status flags (MS$DTR)
                                ;
012352  010246                  DHSDSS: MOV     R2,-(SP)
012354  010346                          MOV     R3,-(SP)
                                ;
                                ;  Get modem index number and select our line
                                ;
012356  016102  0000000                 MOV     LMXNUM(R1),R2   ;Get DH11 index number
012362  016103  0000000                 MOV     LMXLN(R1),R3    ;Get line # within DH11 (0-15)
012366                                  DISABL                  ;;;** Disable interrupts **
012374  042772  0000000 0000000         BIC     #MF$LIN,@DM$CSR(R2) ;;;Clear DM11 line # field
012402  050372  0000000                 BIS     R3,@DM$CSR(R2)  ;;;Select our line
                                ;
                                ;  See if we should set or drop Data Terminal Ready
                                ;
012406  032700  0000000                 BIT     #MS$DTR,R0      ;;;Set or drop DTR?
012412  001004                          BNE     1$              ;;;Br to set DTR
012414  042772  0000000 0000000         BIC     #MF$DTR,@DM$LSR(R2) ;;;Drop DTR
012422  000403                          BR      9$
012424  052772  0000000 0000000 1$:     BIS     #MF$DTR,@DM$LSR(R2);;;Set DTR
                                ;
                                ;  Finished
                                ;
012432                          9$:     ENABL                   ;** Enable interrupts **
012440  012603                          MOV     (SP)+,R3
012442  012602                          MOV     (SP)+,R2
012444  000207                          RETURN
```

```
                                        .SBTTL  DHSSPD -- Set transmit/receive speed for DH11 line
                                ;+-----------------------------------------------------------------
                                ;  DHSSPD is called to set the transmit/receive speed for a DH11 line.
                                ;  The parity and character length parameters are also set.
                                ;
                                ;  Inputs:
                                ;   R0 = Speed, length, and parity codes.
                                ;   R1 = Physical line index number.
                                ;
012446  010246                  DHSSPD: MOV     R2,-(SP)
012450  010346                          MOV     R3,-(SP)
012452  010446                          MOV     R4,-(SP)
                                ;
                                ;  Update the LMXPRM table for this line
                                ;
012454  110061  000010                  MOVB    R0,LMXPRM+1(R1) ;Store new codes for line
                                ;
                                ;  Convert TSX-Plus speed code into DH11 speed code
                                ;
012460  010003                          MOV     R0,R3           ;Get speed code
012462  042703  00000C                  BIC     #^C<LP$SPD>,R3  ;Clear all but speed code
012466  116303  001352'                 MOVB    DHSPCT(R3),R3   ;Convert to DH11 speed code
                                ;
                                ;  Get DH11 index number
                                ;
012472  016102  000000                  MOV     LMXNUM(R1),R2   ;Get DH11 index number
012476  116104  000000                  MOVB    LMXLN(R1),R4    ;Get # of line within mux group
012502  052704  000000                  BIS     #HF$RIE,R4      ;Set receiver interrupt enable flag
                                ;
                                ;  Build value to use for line parameter register
                                ;
012506  072327  000006                  ASH     #6,R3           ;Position speed code for receive speed
012512  010346                          MOV     R3,-(SP)        ;Save positioned receive speed
012514  072327  000004                  ASH     #4,R3           ;Position code for transmit speed
012520  052603                          BIS     (SP)+,R3        ;Combine transmit and receive speed codes
012522  032700  000000                  BIT     #LP$7BT,R0      ;7 bit characters wanted?
012526  001003                          BNE     1$              ;Br if yes
012530  052703  000000                  BIS     #HF$8BT,R3      ;Select 8 bit characters
012534  000402                          BR      2$
012536  052703  000000          1$:     BIS     #HF$7BT,R3      ;Select 7 bit characters
012542  032700  000000          2$:     BIT     #LP$PAR,R0      ;Parity wanted?
012546  001407                          BEQ     3$              ;Br if not
012550  052703  000000                  BIS     #HF$PAR,R3      ;Enable parity
012554  032700  000000                  BIT     #LP$ODD,R0      ;Odd parity wanted?
012560  001402                          BEQ     3$              ;Br if not
012562  052703  000000                  BIS     #HF$ODD,R3      ;Select odd parity
                                ;
                                ;  Select LPR register for line being set and store the LPR value
                                ;
012566                          3$:     DISABL                  ;;;** Disable interrupts **
012574  110472  000000                  MOVB    R4,@MH$SCR(R2)  ;;;Select our mux line
012600  010372  000000                  MOV     R3,@MH$LPR(R2)  ;;;Store the LPR value for this line
012604                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Finished
                                ;
012612  012604                          MOV     (SP)+,R4
```

```
     58 012614  012603                          MOV     (SP)+,R3
     59 012616  012602                          MOV     (SP)+,R2
     60 012620  000207                          RETURN
```

```
                                        .SBTTL  DHSBRK -- Control break transmission for a DH11 line
                                ;-----------------------------------------------------------------------
                                ;  DHSBRK is called to start or stop transmitting a break to a DH11 line.
                                ;
                                ;  Inputs:
                                ;   R0 = Break control flag (MS$BRK)
                                ;   R1 = Line index number
                                ;
012622  010246                  DHSBRK: MOV     R2,-(SP)
012624  010346                          MOV     R3,-(SP)
                                ;
                                ;  Get DH11 index number and line select flag
                                ;
012626  016102  0000000                 MOV     LMXNUM(R1),R2   ;Get DH11 index number
012632  016103  0000000                 MOV     LMXLN(R1),R3    ;Get line within DH11 (0-15)
012636  006303                          ASL     R3              ;Convert line # to word table index
012640  016303  001252'                 MOV     DHLBIT(R3),R3   ;Get flag bit corresponding to line #
                                ;
                                ;  See if we should start or stop sending a break
                                ;
012644  032700  0000000                 BIT     #MS$BRK,R0      ;Start or stop sending break?
012650  001403                          BEQ     1$              ;Br if stop
                                ;
                                ;  Start sending a break to this line
                                ;
012652  050372  0000000                 BIS     R3,@MH$BRK(R2)  ;Set break flag for our line
012656  000402                          BR      9$
                                ;
                                ;  Stop sending a break to this line
                                ;
012660  040372  0000000         1$:     BIC     R3,@MH$BRK(R2)  ;Stop sending a break to this line
                                ;
                                ;  Finished
                                ;
012664  012603                  9$:     MOV     (SP)+,R3
012666  012602                          MOV     (SP)+,R2
012670  000207                          RETURN
```

```
                                        .SBTTL  VHGDSS -- Get data set status for a DHV11 line
                                ;-----------------------------------------------------------------------
                                ;  VHGDSS is called to get the data set status for a DHV11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;
                                ;  Outputs:
                                ;   R0 = Generic modem status flags (MS$xxx)
                                ;
012672  010246                  VHGDSS: MOV     R2,-(SP)
012674  010346                          MOV     R3,-(SP)
012676  010446                          MOV     R4,-(SP)
012700  005000                          CLR     R0              ;Form result in R0
                                ;
                                ;  Get DHV11 index number and line number
                                ;
012702  016102  0000000                 MOV     LMXNUM(R1),R2   ;Get mux index number
012706  016103  0000000                 MOV     LMXLN(R1),R3    ;Get line # within mux group
012712  052703  0000000                 BIS     #VF$RIE,R3      ;Set receiver interrupt enable flag
                                ;
                                ;  Get modem status
                                ;
012716                                  DISABL                  ;;;** Disable interrupts **
012724  110372  0000000                 MOVB    R3,@VH$CSR(R2)  ;;;Select our line in mux
012730  017204  0000000                 MOV     @VH$LCR(R2),R4  ;;;Get line control register
012734  017203  0000000                 MOV     @VH$LSR(R2),R3  ;;;Get current line status
012740                                  ENABL                   ;** Enable interrupts
                                ;
                                ;  See if line is ringing
                                ;
012746  032703  0000000                 BIT     #VF$RNG,R3      ;Is the line ringing?
012752  001402                          BEQ     1$              ;Br if not
012754  052700  0000000                 BIS     #MS$RNG,R0      ;Set ringing flag
                                ;
                                ;  See if carrier is up
                                ;
012760  032703  0000000         1$:     BIT     #VF$DCD,R3      ;Is carrier detected?
012764  001402                          BEQ     2$              ;Br if not
012766  052700  0000000                 BIS     #MS$CAR,R0      ;Set carrier flag
                                ;
                                ;  See if Data Terminal Ready is asserted
                                ;
012772  032704  0000000         2$:     BIT     #VF$DTR,R4      ;Is Data Terminal Ready asserted?
012776  001402                          BEQ     3$              ;Br if not
013000  052700  0000000                 BIS     #MS$DTR,R0      ;Set DTR flag
                                ;
                                ;  Finished
                                ;
013004  012604                  3$:     MOV     (SP)+,R4
013006  012603                          MOV     (SP)+,R3
013010  012602                          MOV     (SP)+,R2
013012  000207                          RETURN
```

```
                                        .SBTTL  VHSDSS -- Set data set status for a DHV11 line
                                ;-----------------------------------------------------------------------
                                ;  VHSDSS is called to set the data set status for a DHV11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;   R0 = Data set status flags (MS$DTR).
                                ;
013014  010246                  VHSDSS: MOV     R2,-(SP)
013016  010346                          MOV     R3,-(SP)
                                ;
                                ;  Get DHV11 mux index number and line number
                                ;
013020  016102  0000000                 MOV     LMXNUM(R1),R2   ;Get mux index number
013024  016103  0000000                 MOV     LMXLN(R1),R3    ;Get # of line within mux group
013030  052703  0000000                 BIS     #VF$RIE,R3      ;Set receiver interrupt enable flag
                                ;
                                ;  Set or drop the Data Terminal Ready flag
                                ;
013034                                  DISABL                  ;;;** Disable interrupts **
013042  110372  0000000                 MOVB    R3,@VH$CSR(R2)  ;;;Select our line in mux
013046  032700  0000000                 BIT     #MS$DTR,R0      ;;;Set or drop DTR?
013052  001004                          BNE     1$              ;;;Br if want to set DTR
013054  042772  0000000 0000000         BIC     #VF$DTR,@VH$LCR(R2);;;Clear DTR bit
013062  000403                          BR      2$
013064  052772  0000000 0000000 1$:     BIS     #VF$DTR,@VH$LCR(R2);;;Set DTR bit
                                ;
                                ;  Finished
                                ;
013072                          2$:     ENABL                   ;** Enable interrupts **
013100  012603                          MOV     (SP)+,R3
013102  012602                          MOV     (SP)+,R2
013104  000207                          RETURN
```

```
                        .SBTTL  VHSSPD -- Set transmit/receive speed for a DHV11 line
                ;-----------------------------------------------------------------------
                ;  Set the transmit/receive speed for a DHV11 line.
                ;
                ;  Inputs:
                ;   R0 = Speed code.
                ;   R1 = Line index number
                ;
013106  010246  VHSSPD: MOV     R2,-(SP)
013110  010346          MOV     R3,-(SP)
013112  010446          MOV     R4,-(SP)
                ;
                ;  Update the LMXPRM table for this line
                ;
013114  110061  000001  MOVB    R0,LMXPRM+1(R1) ;Store flags for line
                ;
                ;  Convert TSX-Plus speed code into DHV11 speed code
                ;
013120  010003          MOV     R0,R3           ;Get flags
013122  042703  177770  BIC     #^C<LP$SPD>,R3  ;Clear all but speed flags
013126  116303  001372' MOVB    VHSPCT(R3),R3   ;Convert to DHV11 speed code
                ;
                ;  Get DHV11 index number
                ;
013132  016102  000000  MOV     LMXNUM(R1),R2   ;Get DHV11 index number
013136  116104  000000  MOVB    LMXLN(R1),R4    ;Get line # within mux group
013142  052704  000000  BIS     #VF$RIE,R4      ;Set receiver interrupt enable flag
                ;
                ;  Construct line parameter value for this line
                ;
013146  000303          SWAB    R3              ;Position speed for receive speed
013150  010346          MOV     R3,-(SP)
013152  072327  000004  ASH     #4,R3           ;Position speed for transmit speed
013156  052603          BIS     (SP)+,R3        ;Combine receive and transmit speeds
013160  032700  000000  BIT     #LP$7BT,R0      ;7 bit characters wanted
013164  001003          BNE     2$              ;Br if yes
013166  052703  000000  BIS     #VF$8BT,R3      ;Set 8 bit characters
013172  000402          BR      3$
013174  052703  000000  2$:     BIS     #VF$7BT,R3      ;Set 7 bit characters
013200  032700  000000  3$:     BIT     #LP$PAR,R0      ;Parity wanted?
013204  001407          BEQ     1$              ;Br if not
013206  052703  000000          BIS     #VF$PAR,R3      ;Enable parity
013212  032700  000000          BIT     #LP$ODD,R0      ;Odd parity wanted?
013216  001002          BNE     1$              ;Br if yes
013220  052703  000000          BIS     #VF$EVN,R3      ;Select even parity
                ;
                ;  Select our line and store the parameter value
                ;
013224          1$:     DISABL                  ;;;** Disable interrupts **
013232  110472  000000          MOVB    R4,@VH$CSR(R2)  ;;;Select our mux line
013236  010372  000000          MOV     R3,@VH$LPR(R2)  ;;;Set LPR value for this line
013242                          ENABL                   ;** Enable interrupts **
                ;
                ;  Finished
                ;
013250  012604          MOV     (SP)+,R4
013252  012603          MOV     (SP)+,R3
```

```
     58 013254  012602                          MOV     (SP)+,R2
     59 013256  000207                          RETURN
```

```
                                        .SBTTL  VHSBRK -- Control break transmission for a DHV11 line
                                ;-----------------------------------------------------------------------
                                ;  Start or stop transmitting a break to a DHV11 line.
                                ;
                                ;  Inputs:
                                ;   R0 = Break control flag (MS$BRK)
                                ;   R1 = Line index number
                                ;
013260  010246                  VHSBRK: MOV     R2,-(SP)
013262  010346                          MOV     R3,-(SP)
                                ;
                                ;  Get mux index number and line number
                                ;
013264  016102  0000000                 MOV     LMXNUM(R1),R2   ;Get mux index number
013270  016103  0000000                 MOV     LMXLN(R1),R3    ;Get # of line within mux
013274  052703  0000000                 BIS     #VF$RIE,R3      ;Set receiver interrupt enable flag
                                ;
                                ;  Set or drop the break control flag within the line control register
                                ;
013300                                  DISABL                  ;;;** Disable interrupts **
013306  110372  0000000                 MOVB    R3,@VH$CSR(R2)  ;;;Select our mux line
013312  032700  0000000                 BIT     #MS$BRK,R0      ;;;Start or stop break?
013316  001404                          BEQ     1$              ;;;Br if stop
013320  052772  0000000 0000000         BIS     #VF$BC,@VH$LCR(R2);;;Set the break flag for the line
013326  000403                          BR      2$
013330  042772  0000000 0000000 1$:     BIC     #VF$BC,@VH$LCR(R2);;;Clear the break flag for the line
                                ;
                                ;  Finished
                                ;
013336                          2$:     ENABL                   ;** Enable interrupts **
013344  012603                          MOV     (SP)+,R3
013346  012602                          MOV     (SP)+,R2
013350  000207                          RETURN
```

```
                                        .SBTTL  DLCLOK -- Timer driven routine for DL11 lines
                                ;-----------------------------------------------------------------
                                ;  DLCLOK is a timer-driven routine called periodically to check on the
                                ;  status of a DL11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number
                                ;
013352                          DLCLOK:
                                ;
                                ;  See if this line has been stolen by some special device handler
                                ;
013352  027127  0000000 0000000         CMP     @INVEC(R1),#INRECV;HAS LX HANDLER STOLEN INTERRUPT VECTOR
013360  101050                          BHI     20$             ;BR IF YES
                                ;
                                ;  Check for lost input interrupts
                                ;
013362  032771  0000000 0000000         BIT     #RCVDON,@RSR(R1);IS AN INPUT CHAR PENDING NOW?
013370  001424                          BEQ     1$              ;BR IF NOT
013372  032761  0000000 0000000         BIT     #$IITIM,LSW5(R1);HAVE WE STARTED TIMER YET?
013400  001415                          BEQ     15$             ;BR IF NOT
013402                                  DISABL                  ;** DISABLE ** (NEEDED FOR FLAKEY DL11'S)
013410  042771  0000000 0000000         BIC     #RDINT,@RSR(R1) ;WE SEEM TO HAVE LOST AN INTERRUPT
013416  052771  0000000 0000000         BIS     #RDINT,@RSR(R1) ;TRY TO FORCE AN INTERRUPT
013424                                  ENABL                   ;** ENABLE **
013432  000403                          BR      1$
013434  052761  0000000 0000000 15$:    BIS     #$IITIM,LSW5(R1);START INPUT INTERRUPT TIMER
                                ;
                                ;  Check for lost output interrupts
                                ;
013442  032761  0000000 0000000 1$:     BIT     #$OITIM,LSW5(R1);Have we started timer interval?
013450  001411                          BEQ     13$             ;Br if not
013452  032761  0000000 0000000         BIT     #$XCHAR,LSW3(R1);Are we still waiting for interrupt?
013460  001410                          BEQ     20$             ;Br if not
013462  042761  0000000 0000000         BIC     #$XCHAR,LSW3(R1);Say wait is over
013470  004737  0000000                 CALL    DLSTRT          ;Try to start transmitter
013474  052761  0000000 0000000 13$:    BIS     #$OITIM,LSW5(R1);Start timed interval
                                ;
                                ;  Finished
                                ;
013502  000207                  20$:    RETURN
```

```
                        .SBTTL  DZCLOK -- Timer driven routine for DZ11 lines
                ;-----------------------------------------------------------------------
                ;  DZCLOK is called periodically from the clock routine to check on the
                ;  status of DZ11 lines.
                ;
                ;  Inputs:
                ;   R1 = Physical line index number
                ;
013504  010246          DZCLOK: MOV     R2,-(SP)
013506  010346                  MOV     R3,-(SP)
                ;
                ;  Get DZ11 mux index number and number of line within mux
                ;
013510  016102  0000000         MOV     LMXNUM(R1),R2   ;Get DZ11 mux index number
013514  016103  0000000         MOV     LMXLN(R1),R3    ;Get # of line within mux (0-7)
013520  116303  001312'         MOVB    MXLBIT(R3),R3   ;Get line select bit for mux register
                ;
                ;  Check for lost input interrupts
                ;
013524  032772  0000000 0000000 BIT     #RDONE,@MXCSR(R2);Does DZ11 have a pending input character?
013532  001415                  BEQ     3$              ;Br if not
013534  032761  0000000 0000000 BIT     #$IITIM,LSW5(R1);Have we started input interrupt timer?
013542  001406                  BEQ     1$              ;Br if not
013544  042772  0000000 0000000 BIC     #RIE,@MXCSR(R2) ;Drop receiver interrupt enable
013552  052772  0000000 0000000 BIS     #RIE,@MXCSR(R2) ;and raise it again to try to force interrupt
013560  052761  0000000 0000000 1$:     BIS     #$IITIM,LSW5(R1);Start input interrupt timer
                ;
                ;  Check for lost output interrupts
                ;
013566  032761  0000000 0000000 3$:     BIT     #$OITIM,LSW5(R1);Have we started timer interval?
013574  001415                  BEQ     13$             ;Br if not
013576  032761  0000000 0000000 BIT     #$XCHAR,LSW3(R1);Are we still waiting for interrupt?
013604  001414                  BEQ     9$              ;Br if not
013606  032761  0000000 0000000 BIT     #$CTRLS,LSW3(R1);Is output suspended due to ctrl-S?
013614  001010                  BNE     9$              ;Br if yes
013616  042761  0000000 0000000 BIC     #$XCHAR,LSW3(R1);Say wait is over
013624  004737  0000000         CALL    DZSTRT          ;Try to start transmitter
013630  052761  0000000 0000000 13$:    BIS     #$OITIM,LSW5(R1);Start timed interval
                ;
                ;  Finished
                ;
013636  012603          9$:     MOV     (SP)+,R3
013640  012602                  MOV     (SP)+,R2
013642  000207                  RETURN
```

```
                                        .SBTTL  DHCLOK -- Timer driven routine for DH11 lines
                                ;-----------------------------------------------------------------------
                                ;  DHCLOK is called every 0.5 seconds from the clock driven routine to
                                ;  do checking for DH11 and DHV11 lines
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;
013644                          VHCLOK:
013644                          DHCLOK:
                                ;
                                ;  Set flag which requests clock driven output processing for the line
                                ;
013644  052761  0000000 0000000         BIS     #$DHCDO,LSWIO(R1)       ;Request clock-driven output
013652  005237  0000000                 INC     NEDCDO                  ;Say clock processing needed
                                ;
                                ;  Finished
                                ;
013656  000207                          RETURN
```

```
                                        .SBTTL  SYSDIE -- Fatal system halt
                                ;-----------------------------------------------------------------------
                                ;  SYSDIE is entered from the system SYSHLT routine when a system
                                ;  crash is occurring because a DIE macro was executed.
                                ;
                                ;  Inputs:
                                ;   DIEMSG = Address of error message to print.
                                ;   DIEARG = Argument value to print with error message.
                                ;   DIEPC  = Address of call to SYSHLT.
                                ;   DIESP  = Stack pointer at time of crash.
                                ;   TRPAR5 = Kernel PAR5 contents at time of crash.
                                ;
                                ;  Save all registers on the stack.
                                ;
013660  010046                  SYSDIE: MOV     R0,-(SP)
013662  010146                          MOV     R1,-(SP)
013664  010246                          MOV     R2,-(SP)
013666  010346                          MOV     R3,-(SP)
013670  010446                          MOV     R4,-(SP)
013672  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize some cells that are used to pass information to TSDUMP
                                ;
013674  005037  000000G                 CLR     DMPOVL          ;Overlay name
013700  005037  000000G                 CLR     DMPHND          ;Handler name
                                ;
                                ;  Print message heading.
                                ;
013704  012701  001056'                 MOV     #TXFSE,R1       ;"FATAL SYSTEM ERROR..."
013710  004737  014242'                 CALL    HLTPRT          ;PRINT IT
                                ;
                                ;  Print abort location.
                                ;
013714  013701  000000G                 MOV     DIEPC,R1        ;GET ADDRESS OF CALL TO SYSHLT
013720  004737  014310'                 CALL    HLTOCT          ;PRINT OCTAL VALUE
                                ;
                                ;  Print error message.
                                ;
013724  013701  000000G                 MOV     DIEMSG,R1       ;GET ADDRESS OF ERROR MESSAGE
013730  062701  000002'                 ADD     #DIEBAS,R1
013734  004737  014242'                 CALL    HLTPRT          ;PRINT IT
                                ;
                                ;  Print argument value.
                                ;
013740  012701  001120'                 MOV     #TXARG,R1       ;"ARGUMENT VALUE = "
013744  004737  014242'                 CALL    HLTPRT          ;PRINT HEADING
013750  013701  000000G                 MOV     DIEARG,R1       ;GET ARGUMENT VALUE
013754  004737  014310'                 CALL    HLTOCT          ;PRINT OCTAL VALUE
                                ;
                                ;  If the argument value is in the par 5 range, it is probably in
                                ;  a system overlay.
                                ;
013760  013701  000000G                 MOV     TRPAR5,R1       ;Get the KPAR5 value
013764  001422                          BEQ     21$             ;Br if zero - print value
                                ;
                                ;  Check for base address of system overlay region.
                                ;
```

```
     58 013766  013700  0000000                MOV     OVRADD,R0       ;Find address of the overlay table
     59 013772  026001  0000000        1$:     CMP     0.PAR(R0),R1    ;Check PAR5 with mapped overlay address
     60 013776  001426                         BEQ     2$              ;Br if values match
     61 014000  062700  000006                 ADD     #6,R0           ;Find the next overlay region
     62 014004  021027  004537                 CMP     (R0),#4537      ;Check for end of table, a <JSR R5,$OVRH>
     63 014010  001370                         BNE     1$              ;Br to check next table entry
     64                                ;
     65                                ;  Check for base of loaded handler region.
     66                                ;
     67 014012  013700  0000000                MOV     NUMDEV,R0       ;Get highest byte index for loaded devices
     68 014016  026001  0000000        11$:    CMP     HANPAR(R0),R1   ;Check PAR5 with mapped handler address
     69 014022  001446                         BEQ     12$             ;Br if values match
     70 014024  162700  000002                 SUB     #2,R0           ;Offset to next device handler
     71 014030  002372                         BGE     11$             ;Br to check next handler entry
     72                                ;
     73                                ;  Print kernel PAR5 contents.
     74                                ;
     75 014032  012701  001137'       21$:    MOV     #TXPAR5,R1      ;"PAR5 VALUE = "
     76 014036  004737  014242'               CALL    HLTPRT          ;PRINT HEADING
     77 014042  013701  0000000               MOV     TRPAR5,R1       ;GET ARGUMENT VALUE
     78 014046  004737  014310'               CALL    HLTOCT          ;PRINT OCTAL VALUE
     79 014052  000445                        BR      3$              ;Go halt the system
     80                                ;
     81                                ;  PAR5 address located in mapped system region - report segment ID.
     82                                ;
     83 014054  016004  0000000        2$:     MOV     0.ADR(R0),R4    ;Get the rad50 overlay identifier
     84 014060  163700  0000000                SUB     OVRADD,R0       ;Sub the base address of the overlay table
     85 014064  010003                         MOV     R0,R3           ;Move to low-order address
     86 014066  005002                         CLR     R2              ;Clear high-order address
     87 014070  071227  000006                 DIV     #6,R2           ;Divide by 6 (# bytes/ overlay table entry)
     88 014074  005202                         INC     R2              ;Normalize base to one
     89 014076  012701  001155'               MOV     #TXSEG,R1       ;"Seg. value ="
     90 014102  004737  014242'               CALL    HLTPRT          ;Print heading
     91 014106  010201                        MOV     R2,R1           ;Get the segment number
     92 014110  004737  014310'               CALL    HLTOCT          ;Print the octal segment number
     93 014114  012701  001173'               MOV     #TXOID,R1       ;"Overlay: "
     94 014120  004737  014242'               CALL    HLTPRT          ;Print heading
     95 014124  010401                        MOV     R4,R1           ;Restore the overlay identifier
     96 014126  010137  0000000               MOV     R1,DMPOVL       ;Pass overlay name to TSDUMP
     97 014132  004737  014366'               CALL    HLTRAD          ;Print the rad50 overlay identifier
     98 014136  000413                        BR      3$              ;Go halt the system
     99                                ;
    100                                ;  PAR5 address located in loaded device handler - report device name.
    101                                ;
    102 014140  016002  0000000       12$:    MOV     PNAME(R0),R2    ;Get the RAD50 device name
    103 014144  012701  001205'               MOV     #TXDEV,R1       ;"Device name :"
    104 014150  004737  014242'               CALL    HLTPRT          ;Print heading
    105 014154  010201                        MOV     R2,R1           ;Get the device name
    106 014156  010137  0000000               MOV     R1,DMPHND       ;Pass device name to TSDUMP
    107 014162  004737  014366'               CALL    HLTRAD          ;Print the rad50 device name
    108                               ;
    109                               ;  Print stack pointer at time of crash
    110                               ;
    111 014166  012701  001223'       3$:     MOV     #SPTXT,R1       ;Point to message heading
    112 014172  004737  014242'               CALL    HLTPRT          ;Print the heading
    113 014176  013701  0000000               MOV     DIESP,R1        ;Get SP at time of crash
    114 014202  004737  014310'               CALL    HLTOCT          ;Print it
```

```
                                        ;
                                        ;  See if we should call the system crash dump module or halt the system
                                        ;
014206  105737  0000000                       TSTB    VSYDMP          ;Should we do a system dump?
014212  001001                                BNE     4$              ;Br if yes
014214  000000                                HALT                    ;Halt the system
                                        ;
                                        ;  Enter system overlay to produce a crash dump
                                        ;
014216  013701  0000000         4$:           MOV     DIEMSG,R1       ;Get address of error message text
014222  062701  000002'                       ADD     #DIEBAS,R1
014226  012702  0000000                       MOV     #DMPTXT,R2      ;Point to area where we pass message
014232  112122                  5$:           MOVB    (R1)+,(R2)+     ;Store message text
014234  001376                                BNE     5$              ;Loop till all of message moved
014236  000137  0000000                       JMP     DODUMP          ;Enter dump routine
```

```
                                ;-------------------------------------------------------------------------
                                ;  HLTPRT is called to print an ASCIZ string on the console terminal.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of ASCIZ string to print.
                                ;
014242  010146          HLTPRT: MOV     R1,-(SP)
014244  112100          1$:     MOVB    (R1)+,R0        ;GET NEXT CHAR FROM TEXT STRING
014246  001406                  BEQ     2$              ;BR IF HIT END OF STRING
014250  120027  000200          CMPB    R0,#200         ;END WITHOUT CR-LF?
014254  001413                  BEQ     3$              ;BR IF YES
014256  004737  014524'         CALL    HLTCHR          ;SEND CHAR TO TERMINAL
014262  000770                  BR      1$              ;GO GET NEXT CHAR
                                ;  Print CR-Lf.
014264  012700  000015  2$:     MOV     #CR,R0          ;PRINT CR
014270  004737  014524'         CALL    HLTCHR
014274  012700  000012          MOV     #LF,R0          ;PRINT LF
014300  004737  014524'         CALL    HLTCHR
                                ;  Finished
014304  012601  3$:     MOV     (SP)+,R1
014306  000207                  RETURN

                                ;-------------------------------------------------------------------------
                                ;  HLTOCT is called to convert a binary value to an octal character string
                                ;  and print that string on the console terminal.
                                ;
                                ;  Inputs:
                                ;   R1 = Binary value to be converted and printed.
                                ;
014310  010146          HLTOCT: MOV     R1,-(SP)
014312  010246                  MOV     R2,-(SP)
014314  012702  000006          MOV     #6.,R2          ;GET # OF OCTAL DIGITS IN RESULT STRING
014320  005000                  CLR     R0              ;SET FOR SHIFT
014322  073027  000001          ASHC    #1,R0           ;SHIFT 1ST BIT INTO R0
014326  000403                  BR      2$              ;ENTER CONVERSION LOOP
014330  005000          1$:     CLR     R0              ;SET FOR SHIFT
014332  073027  000003          ASHC    #3,R0           ;SHIFT AN OCTAL DIGIT INTO R0
014336  062700  000060  2$:     ADD     #'0,R0          ;CONVERT BINARY VALUE TO ASCII CHARACTER
014342  004737  014524'         CALL    HLTCHR          ;PRINT A CHARACTER
014346  077210                  SOB     R2,1$           ;LOOP TO PRINT REST OF VALUE
014350  012701  001136'         MOV     #TXNUL,R1       ;NOW PRINT CR-LF
014354  004737  014242'         CALL    HLTPRT
014360  012602                  MOV     (SP)+,R2
014362  012601                  MOV     (SP)+,R1
014364  000207                  RETURN

                                ;-------------------------------------------------------------------------
                                ;  HLTRAD is called to convert a RAD50 value to an ascii character string
                                ;  and print that string on the console terminal.
                                ;
                                ;  Inputs:
                                ;   R1 = RAD50 value to be converted and printed.
                                ;
014366  010146          HLTRAD: MOV     R1,-(SP)
014370  010246                  MOV     R2,-(SP)
014372  005000                  CLR     R0              ;Clear high order
014374  071027  003100          DIV     #50*50,R0       ;Divide for 1st byte
```

```
     58 014400  116000  014454'                MOVB    R50CHR(R0),R0   ;Get output character
     59 014404  004737  014524'                CALL    HLTCHR          ;Print a character
     60 014410  005000                          CLR     R0              ;Clear high order
     61 014412  071027  000050                  DIV     #50,R0          ;Divide for 2nd byte
     62 014416  116000  014454'                MOVB    R50CHR(R0),R0   ;Get output character
     63 014422  004737  014524'                CALL    HLTCHR          ;Print a character
     64 014426  116100  014454'                MOVB    R50CHR(R1),R0   ;Get output character
     65 014432  004737  014524'                CALL    HLTCHR          ;Print a character
     66 014436  012701  001136'                MOV     #TXNUL,R1       ;Now print cr-lf
     67 014442  004737  014242'                CALL    HLTPRT
     68 014446  012602                          MOV     (SP)+,R2
     69 014450  012601                          MOV     (SP)+,R1
     70 014452  000207                          RETURN
     71
     72                                         .EVEN
     73
     74
     75 014454      040      101      102  R50CHR: .ASCII  / ABCDEFGHIJKLMNOPQRSTUVWXYZ$. 0123456789/
     76
     77
     78                                         .EVEN
     79
     80                                ;-------------------------------------------------------------------
     81                                ;  HLTCHR is called to print a single character on the console terminal
     82                                ;  during a system crash.
     83                                ;
     84                                ;  Inputs:
     85                                ;   R0 = Character to print.
     86                                ;
     87 014524  032737  0000000 0000000 HLTCHR: BIT     #TRRDY,@#CTTSR  ;IS TERMINAL READY FOR ANOTHER CHARACTER?
     88 014532  001774                          BEQ     HLTCHR          ;BR IF NOT
     89 014534  110037  0000000                 MOVB    R0,@#CTTBR      ;SEND CHARACTER TO CONSOLE TERMINAL
     90 014540  000207                          RETURN
```

```
                                        .SBTTL  EXCINI -- Final system initialization
                                ;-------------------------------------------------------------------------
                                ;  EXCINI is the last part of the system start-up initialization routine.
                                ;  It is placed in TSEXEC rather than TSINIT so that tables that are allocated
                                ;  over TSINIT can be clobbered during this part of the initialization.
                                ;
014542                          EXCINI:
                                ;
                                ;  Set up I/O queue free chain
                                ;
014542  013701  000000G                 MOV     FREIOQ,R1       ;BASE OF I/O QUEUE AREA
014546  012702  177770                  MOV     #NUMIOQ-1,R2    ;# I/Q QUEUE ELEMENTS - 1
014552  010103                  10$:    MOV     R1,R3           ;GET ADDRESS OF CURRENT QUEUE ELEMENT
014554  062703  000000G                 ADD     #IOQSIZ,R3      ;POINT TO NEXT QUEUE ELEMENT
014560  010361  000000G                 MOV     R3,Q.LINK(R1)   ;SET FORWARD LINK IN OUR ELEMENT
014564  010301                          MOV     R3,R1           ;POINT TO NEXT ONE
014566  077207                          SOB     R2,10$          ;GO DO IT
014570  005063  000000G                 CLR     Q.LINK(R3)      ;ZERO LAST FORWARD POINTER
                                ;
                                ;  Set up fork blocks that are allocated in init area
                                ;
014574  012702  000000C                 MOV     #<<NUMFRK-FRKGEN>-1>,R2 ;Get # fork blocks to allocate - 1
014600  003417                          BLE     31$             ;Br if none to allocate
014602  013701  000000G                 MOV     FRKINI,R1       ;Point to start of fork area
014606  010103                  30$:    MOV     R1,R3           ;Get address of current block
014610  062703  000000G                 ADD     #FQ$$SZ,R3      ;Point to next block
014614  010361  000000G                 MOV     R3,FQ$LNK(R1)   ;Set forward link in our element
014620  010301                          MOV     R3,R1           ;Get address of next block
014622  077207                          SOB     R2,30$          ;Allocate all but last block
014624  013763  000000G 000000G         MOV     FREFRK,FQ$LNK(R3) ;Add static fork blocks to end of list
014632  013737  000000G 000000G         MOV     FRKINI,FREFRK   ;Put new fork blocks at head of list
                                ;
                                ;  Set up cache control block free chain
                                ;
014640  005737  000000G         31$:    TST     CSHALC          ;Is data caching wanted?
014644  001415                          BEQ     20$             ;Br if not
014646  012702  177777G                 MOV     #NUMCCB-1,R2    ;Get # cache control blocks - 1
014652  013701  000000G                 MOV     CCBHD,R1        ;Base of control block area
014656  010103                  21$:    MOV     R1,R3           ;Get address of current control block
014660  062703  000000G                 ADD     #CC$$SZ,R3      ;Get address of next control block
014664  010361  000000G                 MOV     R3,CC$LNK(R1)   ;Make current block point to next one
014670  010301                          MOV     R3,R1           ;Get address of next block
014672  077207                          SOB     R2,21$          ;Loop if more to link together
014674  005061  000000G                 CLR     CC$LNK(R1)      ;Zero last link
                                ;
                                ;  Initialize device mount table
                                ;
014700  013701  000000G         20$:    MOV     CSHDEV,R1       ;Point to start of area
014704  005021                  40$:    CLR     (R1)+           ;Zero the entire table
014706  020137  000000G                 CMP     R1,CSHDVN       ;Reached end?
014712  103774                          BLO     40$             ;Loop if not
                                ;
                                ;  Initialize shared PLAS region control blocks
                                ;
014714  013701  000000G                 MOV     SHRRCB,R1       ;Point to 1st region control block
014720  020137  000000G         36$:    CMP     R1,SHRRCN       ;Have we initialized entire area?
014724  103002                          BHIS    35$             ;Br if yes
```

```
     58 014726  005021                        CLR     (R1)+           ;Zero the area
     59 014730  000773                        BR      36$
     60                                ;
     61                                ;  Initialize free list of swap command packets
     62                                ;
     63 014732  013701  000000G       35$:    MOV     SCPFHD,R1       ;Point to area where packets are
     64 014736  001413                        BEQ     37$             ;Br if nothing to initialize
     65 014740  012702  177777G               MOV     #NSCP-1,R2      ;Get # packets -1
     66 014744  010103                38$:    MOV     R1,R3           ;Get pointer to current packet
     67 014746  062703  000000G               ADD     #SP$$SZ,R3      ;Get pointer to next packet
     68 014752  010361  000000G               MOV     R3,SP$LNK(R1)   ;Make our packet point to next
     69 014756  010301                        MOV     R3,R1           ;Get pointer to next packet
     70 014760  077207                        SOB     R2,38$          ;Loop till all packets but last linked in
     71 014762  005061  000000G               CLR     SP$LNK(R1)      ;Say last packet is end of list
     72                                ;
     73                                ;  Initialize the free chain of monitor control blocks
     74                                ;
     75 014766  013701  000000G       37$:    MOV     MONFQH,R1       ;Get base of area for control blocks
     76 014772  013702  000000G               MOV     VMXMON,R2       ;Get # monitor blocks
     77 014776  001413                        BEQ     28$             ;Br if none wanted
     78 015000  005302                        DEC     R2              ;Get one less than # wanted
     79 015002  001407                        BEQ     42$             ;Br if only one wanted
     80 015004  010103                29$:    MOV     R1,R3           ;Get address of current block
     81 015006  062703  000000G               ADD     #JM$$SZ,R3      ;Get address of next control block
     82 015012  010361  000000G               MOV     R3,JM$LNK(R1)   ;Make current block point to next
     83 015016  010301                        MOV     R3,R1           ;Get address of next block
     84 015020  077207                        SOB     R2,29$          ;Br if more to allocate
     85 015022  005061  000000G       42$:    CLR     JM$LNK(R1)      ;Zero last link
     86                                ;
     87                                ;  Initialize the tables that keep track of free space in job swap file
     88                                ;
     89 015026  013700  000000G       28$:    MOV     VSWPSL,R0       ;Get # slots in swap file
     90 015032  001412                        BEQ     33$             ;Br if no swap file
     91 015034  013702  000000G               MOV     SWPPOS,R2       ;Point to table that has starting blk #'s
     92 015040  013703  000000G               MOV     SWPJOB,R3       ;Point to table that has job #'s
     93 015044  005004                        CLR     R4              ;1st slot is at block 0
     94 015046  010422                32$:    MOV     R4,(R2)+        ;Set block # for this slot
     95 015050  005023                        CLR     (R3)+           ;Say no job using this slot now
     96 015052  063704  000000G               ADD     SLTSIZ,R4       ;Add # blocks used by a slot
     97 015056  077005                        SOB     R0,32$          ;Loop till all slots initialized
     98                                ;
     99                                ;  Initialize vector for each multiplexor that is used to map from
    100                                ;  the Mux line number to the TSX-Plus logical line number
    101                                ;
    102 015060  012701  000000G       33$:    MOV     #LSTMX,R1       ;Get index # of last mux
    103 015064  001470                        BEQ     27$             ;Branch if no mux's to initialize
    104 015066  016103  000000G       15$:    MOV     MXLNT(R1),R3    ;GET ADDRESS OF MUX MAPPING TABLE
    105 015072  012700  000020                MOV     #16.,R0         ;ZERO 16 BYTES IN TABLE
    106 015076  105023                17$:    CLRB    (R3)+
    107 015100  077002                        SOB     R0,17$
    108 015102  162701  000002                SUB     #2,R1           ;More mux tables to init?
    109 015106  003367                        BGT     15$             ;Loop if yes
    110 015110  012701  000000G       16$:    MOV     #LSTHL,R1       ;GET INDEX # OF LAST PHYSICAL LINE
    111 015114  032761  000000G 000000G 19$:  BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
    112 015122  001410                        BEQ     18$             ;Br if not
    113 015124  016102  000000G               MOV     LMXNUM(R1),R2   ;IS THIS LINE CONNECTED TO A MUX?
    114 015130  001405                        BEQ     18$             ;BR IF NOT
```

```
     115 015132  016202  0000000                      MOV     MXLNT(R2),R2    ;GET ADDRESS OF MAP VECTOR FOR THIS MUX
     116 015136  066102  0000000                      ADD     LMXLN(R1),R2    ;ADD OFFSET WITHIN VECTOR
     117 015142  110112                               MOVB    R1,(R2)         ;SET TSX LINE # WITHIN MAP VECTOR
     118 015144  162701  000002           18$:        SUB     #2,R1           ;GET INDEX # OF NEXT LINE
     119 015150  003361                               BGT     19$             ;LOOP IF MORE LINES
     120                                  ;
     121                                  ;  Enable interrupts for multiplexors
     122                                  ;
     123 015152  105737  0000000                      TSTB    PROFLG          ;Are we running on a Pro?
     124 015156  001033                               BNE     27$             ;Br if yes -- Don't have any mux's on pro
     125 015160  012701  0000000                      MOV     #LSTMX,R1       ;Get index # of last mux
     126 015164  005761  0000000          23$:        TST     MXCSR(R1)       ;Is this mux installed?
     127 015170  001423                               BEQ     25$             ;Br if not
     128 015172  016100  0000000                      MOV     MXTYPE(R1),R0   ;Get mux type code
     129 015176  020027  0000000                      CMP     R0,#CDX$DZ      ;Is this a DZ11?
     130 015202  001004                               BNE     24$             ;Br if not
     131 015204  012771  0000000 0000000              MOV     #INTMX1,@MXCSR(R1);Enable DZ11 interrupts
     132 015212  000412                               BR      25$
     133 015214  020027  0000000          24$:        CMP     R0,#CDX$VH      ;Is this a DHV11?
     134 015220  001004                               BNE     26$             ;Br if not
     135 015222  052771  000000C 0000000              BIS     #<VF$TIE!VF$RIE>,@VH$CSR(R1);Enable DHV11 interrupts
     136 015230  000403                               BR      25$
     137 015232  052771  000000C 0000000  26$:        BIS     #<HF$TIE!HF$RIE>,@MH$SCR(R1);Enable DH11 interrupts
     138 015240  162701  000002           25$:        SUB     #2,R1           ;More mux's to initialize?
     139 015244  003347                               BGT     23$             ;Loop if yes
     140                                  ;
     141                                  ;  Initialize system message buffer free list.
     142                                  ;
     143 015246  013703  0000000          27$:        MOV     SNMSHD,R3       ;GET ADDRESS OF 1ST MESSAGE BUFFER
     144 015252  012704  1777770                      MOV     #<NMSNMB-1>,R4  ;GET # MESSAGE BUFFERS -1
     145 015256  010302                   7$:         MOV     R3,R2           ;GET ADDRESS OF MESSAGE BUFFER
     146 015260  062702  0000000                      ADD     #SB$$SZ,R2      ;POINT TO FOLLOWING BUFFER
     147 015264  010263  0000000                      MOV     R2,SB$LNK(R3)   ;CHAIN TOGETHER THE ENTRIES
     148 015270  010203                               MOV     R2,R3           ;MOVE ON TO NEXT ENTRY
     149 015272  077407                               SOB     R4,7$           ;DO ALL BUT LAST
     150 015274  005063  0000000                      CLR     SB$LNK(R3)      ;SET FORWARD LINK FOR LAST ENTRY TO ZERO
     151                                  ;
     152                                  ;  Initialize INSTALLed program table
     153                                  ;
     154 015300  004737  015746'                      CALL    INSINI          ;Initialize installed program table
     155                                  ;
     156                                  ;  Initialize CL table information
     157                                  ;
     158 015304  012704  000000C                      MOV     #2*<CLTOTL-1>,R4;Get index to last CL unit
     159 015310  002407                               BLT     39$             ;Br if there are no CL units
     160 015312  016400  0000000          41$:        MOV     CL$EPS(R4),R0   ;Get pointer to EOF string buffer
     161 015316  001401                               BEQ     45$             ;Br if no string buffer
     162 015320  105010                               CLRB    (R0)            ;Say no EOF string
     163 015322  162704  000002           45$:        SUB     #2,R4           ;Get next index
     164 015326  002371                               BGE     41$             ;Loop if more CL units
     165                                  ;
     166                                  ;  Initialize the file management tables
     167                                  ;
     168 015330                           39$:        OCALL   USRINI          ;Call TSUSR initialization routine
     169                                  ;
     170                                  ;  Initialize spool buffer list
     171                                  ;
```

```
                                ;
                                ;  Initialize the record locking system
                                ;
015352  005737  0000000  11$:    TST     VMXSF           ;Is record locking support wanted?
015356  001402                  BEQ     12$             ;Br if not
015360  004777  0000000          CALL    @LOKINI         ;Initialize the shared file system
                                ;
                                ;  Initialize the message communication system
                                ;
015364  005737  0000000  12$:    TST     VMAXMC          ;Is message communication support wanted?
015370  001403                  BEQ     13$             ;Br if not
015372                          OCALL   MSGINI          ;Initialize the message system
                                ;
                                ;  Initialize the data caching facility
                                ;
015400  005737  0000000  13$:    TST     CSHALC          ;Is data caching wanted?
015404  001402                  BEQ     43$             ;Br if not
015406  004777  0000000          CALL    @CSHINI         ;Initialize data caching facility
                                ;
                                ;  Initialize the PLAS system
                                ;
015412  005737  0000000  43$:    TST     VPLAS           ;Is PLAS support included in system?
015416  001403                  BEQ     44$             ;Br if not
015420                          OCALL   PLSINI          ;Do PLAS initialization
                                ;
                                ;  Initialize the dislay window management system
                                ;
015426  005737  0000000  44$:    TST     VMXWIN          ;Is window support wanted?
015432  001403                  BEQ     34$             ;Br if not
015434                          OCALL   WININI          ;Initialize window system
                                ;
                                ;  Connect clock interrupt to clock interrupt routine
                                ;
015442  012737  0000000 000100 34$:    MOV     #CLKINT,@#100   ;Set up clock interrupt vector
015450  105737  0000000          TSTB    PROFLG          ;Is this a PRO?
015454  001410                  BEQ     22$             ;Br if not
015456  012737  0000000 000230  MOV     #CLKINT,@#230   ;380 clock interrupt vector
015464  012737  000340  000232  MOV     #340,@#232
015472  005737  0000000          TST     @#PCCCR2        ;Access CSR2 to start clock interrupts
                                ;
                                ;  Initialize time-sharing line parameters and speeds
                                ;
015476  004737  015626' 22$:    CALL    INISPD          ;Initialize time-sharing line speeds
                                ;
                                ;  Start lines that specified $START when genned.
                                ;
015502  012701  000002          MOV     #2,R1           ;INDEX # OF 1ST LINE
015506  032761  0000000 0000000 2$:     BIT     #$START,ILSW2(R1);DOES THIS LINE WANT AUTO STARTUP?
015514  001413                  BEQ     3$              ;BR IF NOT
015516  032761  0000000 0000000  BIT     #$DEAD,LSW3(R1) ;IS THIS LINE INSTALLED?
015524  001007                  BNE     3$              ;BR IF NOT
015526  032761  0000000 0000000  BIT     #$PHONE,ILSW2(R1);IS THIS A DIAL-UP LINE?
015534  001003                  BNE     3$              ;BR IF IT IS (NO AUTO STARTUP THEN)
015536  005000                  CLR     R0              ;No secondary start-up command file
```

```
    229 015540  004737  001606'                 CALL    INITLN          ;INITIATE THE LINE
    230 015544  062701  000002          3$:     ADD     #2,R1           ;ADVANCE JOB #
    231 015550  020127  000000G                 CMP     R1,#LSTPL       ;MORE TO CHECK?
    232 015554  101754                          BLOS    2$              ;BR IF YES
    233                                 ;
    234                                 ;  Start any detached jobs
    235                                 ;
    236 015556  012701  000000G                 MOV     #FSTDL,R1       ;# OF FIRST DETACHED JOB
    237 015562  020127  000000G         6$:     CMP     R1,#LSTDL       ;DONE ALL DETACHED JOBS?
    238 015566  101013                          BHI     4$              ;BR IF YES
    239 015570  016102  000000G                 MOV     LSUCF(R1),R2    ;DOES THIS JOB HAVE A START-UP COMMAND FILE?
    240 015574  001405                          BEQ     5$              ;BR IF NOT
    241 015576  105712                          TSTB    (R2)            ;IS COMMAND FILE NAME NULL?
    242 015600  001403                          BEQ     5$              ;BR IF YES
    243 015602  005000                          CLR     R0              ;No secondary start-up command file
    244 015604  004737  001606'                 CALL    INITLN          ;INITIATE THE LINE
    245 015610  062701  000002          5$:     ADD     #2,R1           ;CHECK NEXT LINE
    246 015614  000762                          BR      6$
    247 015616                          4$:
    248                                 ;
    249                                 ;  Finished system initialization
    250 015616  105037  000000G                 CLRB    INITFL          ;SAY SYSTEM INITIALIZATION IS FINISHED
    251                                 ;
    252                                 ;  Enter job scheduler to wait for first job to run
    253                                 ;
    254 015622  000137  000000G                 JMP     EXEC            ;ENTER JOB SCHEDULER
```

```
                                        .SBTTL  INISPD -- Initialize time-sharing line speeds
                                ;-------------------------------------------------------------------------
                                ;  INISPD is called to initialize the transmit/receive speeds for
                                ;  time-sharing lines.
                                ;
015626  010146                  INISPD: MOV     R1,-(SP)
015630  010246                          MOV     R2,-(SP)
                                ;
                                ;  Begin loop to set each line
                                ;
015632  012701  000000G                 MOV     #LSTHL,R1       ;Get index to last hardware line
                                ;
                                ;  Skip this line if it is dead or not connected to hardware
                                ;
015636  032761  000000G 000000G 1$:     BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
015644  001432                          BEQ     2$              ;Br if not
015646  032761  000000G 000000G         BIT     #$DEAD,LSW3(R1) ;Is this line installed?
015654  001026                          BNE     2$              ;Br if not
                                ;
                                ;  Set the speed of this line
                                ;
015656  116100  000001G                 MOVB    LMXPRM+1(R1),R0 ;Get speed parameters
                                ;
                                ;  Initialize speed to 9600 baud if autobaud was specified for line
                                ;
015662  032761  000000G 000000G         BIT     #$AUTO,ILSW2(R1);Is autobaud wanted for this line?
015670  001402                          BEQ     3$              ;Br if not
015672  012700  000000G                 MOV     #S9600,R0       ;Set speed to 9600
015676  016102  000000G         3$:     MOV     LCDTYP(R1),R2   ;Get device type code for this line
015702  004772  000000G                 CALL    @CDSSPD(R2)     ;Call hardware-dependent routine to set speed
                                ;
                                ;  Convert $TDEAD lines (deaded with TSXMOD) to $DEAD lines
                                ;
015706  020127  000000G                 CMP     R1,#LSTPL       ;Is this a time-sharing line?
015712  101007                          BHI     2$              ;Skip if not (skip sub, det & io lines)
015714  032761  000000G 000000G         BIT     #$TDEAD,LSW11(R1) ;Do we want this line to be dead?
015722  001403                          BEQ     2$              ;Br if not
015724  052761  000000G 000000G         BIS     #$DEAD,LSW3(R1) ;Flag line as dead
                                ;
                                ;  See if there are more lines
                                ;
015732  162701  000002          2$:     SUB     #2,R1           ;Are there more lines to do?
015736  003337                          BGT     1$              ;Br if yes
                                ;
                                ;  Finished
                                ;
015740  012602                          MOV     (SP)+,R2
015742  012601                          MOV     (SP)+,R1
015744  000207                          RETURN
```

```
                                        .SBTTL  INSINI -- Initialize installed program table
                                ;-------------------------------------------------------------------------------
                                ;  Initialize the installed program table.
                                ;
015746  010246                  INSINI: MOV     R2,-(SP)
015750  010346                          MOV     R3,-(SP)
                                ;
                                ;  Initially, zero the entire table
                                ;
015752  013702  0000000                 MOV     INSTBL,R2       ;Point to start of table
015756  005022                  1$:     CLR     (R2)+           ;Zero the table
015760  020237  0000000                 CMP     R2,INSTBN       ;Reached end of table?
015764  103774                          BLO     1$              ;Loop if not
                                ;
                                ;  Now install certain system programs
                                ;
015766  013702  0000000                 MOV     INSTBL,R2       ;Point to 1st table entry
015772  012703  001412'                 MOV     #SRFPRG,R3      ;Point to table with info about sys programs
                                ;
                                ;  Set file spec for program
                                ;
015776  013762  0000000 0000000 2$:     MOV     SYNAME,II$NAM(R2);Set SY as device name
016004  012362  0000020                 MOV     (R3)+,II$NAM+2(R2) ;Set 1st 3 chars of program name
016010  012362  0000040                 MOV     (R3)+,II$NAM+4(R2) ;Set 2nd 3 chars of program name
016014  013762  001350' 0000060         MOV     R50SAV,II$NAM+6(R2);Set SAV as file extension
                                ;
                                ;  Set run attribute flags
                                ;
016022  012362  0000000                 MOV     (R3)+,II$FLG(R2);Set run attribute flags
                                ;
                                ;  Set privileges for program
                                ;
016026  052762  0000000 0000000         BIS     #PO$DBG,II$NPV(R2) ;Set NODEBUG privilege flag
016034  012704  0000000         3$:     MOV     #II$PRV,R4      ;Assume we will set some privileges
016040  012300                          MOV     (R3)+,R0        ;Are there any privilege flags?
016042  001412                          BEQ     4$              ;Br if not
016044  002003                          BGE     5$              ;Br if we are to set privileges
016046  005400                          NEG     R0              ;Get positive offset
016050  012704  0000000                 MOV     #II$NPV,R4      ;Point to reset-privilege vector
016054  005300                  5$:     DEC     R0              ;Convert to offset
016056  006300                          ASL     R0              ;Convert to word offset
016060  060004                          ADD     R0,R4           ;Point to word in vector to change
016062  060204                          ADD     R2,R4           ;Add address of install table entry
016064  012314                          MOV     (R3)+,(R4)      ;Set bits in install table entry
016066  000762                          BR      3$              ;Go see if more privileges for program
                                ;
                                ;  See if there are more programs to install
                                ;
016070  062702  0000000         4$:     ADD     #II$$SZ,R2      ;Point to next install table entry
016074  020327  001606'                 CMP     R3,#SRFEND      ;Installed all system programs?
016100  103736                          BLO     2$              ;Loop if not
                                ;
                                ;  Finished
                                ;
016102  012603                          MOV     (SP)+,R3
016104  012602                          MOV     (SP)+,R2
016106  000207                          RETURN
```

INSINI -- Initialize installed program table

```
     58
     59          000000                              .END
Errors detected:   0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  9768 Words  ( 39 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:01:02.04
DK:TSEXC2,LP:TSEXC2=DK:TSEXC2.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

```
$1STLG    1-55      10-169
$AUTO     1-51      10-105    28-14     53-26
$CARMN    1-61       5-90      5-95     30-45
$CARUP    1-55       5-93     10-171    30-40     30-64
$CTRLC    1-74       8-29      8-38
$CTRLO    1-51       6-111
$CTRLS    1-70       6-111    10-30     48-34
$DBGBK    1-107      8-18     12-28
$DBGMD    1-58       8-17
$DEAD     1-62      10-171    29-15     52-224    53-17     53-38
$DEBUG    1-92      13-57
$DEFER    1-87       6-125
$DETCH    1-40       6-104    10-10
$DHBF1    1-90      10-34
$DHBF2    1-90      10-34
$DHCDO    1-40      49-14
$DILUP    1-40       5-74     10-28     10-65     30-41     30-48
$DISCN    1-46       7-29      7-33     10-50     10-126    30-50
$DODFR    1-87       6-127
$DOOFF    1-72       7-31     30-50
$FPUEX    1-85      14-11
$GCECO    1-87       6-127
$GEMAR    1-34       8-13
$HARD     1-78      10-171    29-17     52-111    53-15
$IITIM    1-82      47-20     47-27     48-22     48-26
$INCOR    1-102      5-64     18-27
$INIT     1-65       6-9
$INKMN    1-64       6-15      7-14      7-41     12-14     13-23     26-36
$IOMAP    1-75       8-17
$LOFCF    1-66      10-124    30-52
$MAPOK    1-83       7-42
$MLOCK    1-75       8-17
$NABRS    1-51      10-108    28-27     28-29
$NOABT    1-84       8-18
$NOIN     1-72       5-75      7-32     30-47
$NOLF     1-58       8-17
$NOUCR    1-53       7-11      7-25
$OITIM    1-97      47-31     47-37     48-30     48-38
$PHONE    1-62       5-91     29-29     52-226
$PWKEY    1-35       5-77
$RDSAV    1-34       8-13
$RNMLK    1-58       8-18
$SQQ0     1-93      17-30
$SQQ1     1-104     17-56
$SQQ1A    1-104     17-81
$SQQ1B    1-104     17-100
$SQQ1C    1-104     17-91
$SQQ2     1-104     17-115
$SQQ3     1-93      17-37
$SOTFN    1-87      27-16     27-18
$START    1-62      52-222
$SUCF     1-59       5-76
$TDEAD    1-62      53-36
$VIRJB    1-80       7-43
$VNOTT    1-70      10-16
$XCHAR    1-83      10-36     10-171    47-33     47-35     48-32     48-36
```

Cross reference table (CREF V05.04)

```
...V1     7-58      7-72      7-72
...V2     7-58      7-58#     7-72      7-72      7-72#     7-72#
ABORT     1-26     11-52     13-71#    14-20     14-28
ABRTAD    1-64     13-71*
ABRTCD    1-64     13-72*
AF$BYA    1-45      4-49      4-75
AF$DUP    1-36      4-29
AF$HIE    1-45      4-37      4-41      4-45
AF$IND    1-36      4-33
AF$IOP    1-45      4-58
AF$MEM    1-45      4-63
AF$NOI    1-45      4-71
AF$NOW    1-45      4-33      4-37      4-41      4-45      4-67      4-71      4-83
AF$NPW    1-35      4-71
AF$PLK    1-44      4-49
AF$SCA    1-45      4-37      4-41      4-45      4-54      4-67      4-71      4-83
AF$SET    1-36      4-58
AF$UCL    1-36      4-79
BELL      1-120#    3-27
C.NUMQ    1-34      6-48*
CANCPL    8-104     9-10#
CANIOT    1-56      8-44
CANMKT    1-76      8-70
CARDET    1-81     31-26
CC$$SZ    1-105    52-40
CC$LNK    1-105    52-41*    52-44*
CCBHD     1-66     52-38
CDCLOK    1-60     29-23
CDGDSS    1-106    30-15
CDIFLG    1-111    15-58     15-60*
CDIRTN    1-111    15-63
CDOFLG    1-111    15-67     15-69*
CDORTN    1-111    15-72
CDSDSS    1-106    30-27     30-58     30-78
CDSSPD    1-40     53-30
CDX$DZ    1-52     52-129
CDX$VH    1-107    52-133
CHKABT    1-57     11-66     13-29
CHKPRT   18-116    20-7#
CHKUSP    1-63     11-50     13-20     14-25
CINFLG    1-37      8-31*     8-42
CKMRKT   15-23     24-6#
CKSCHD   15-34     27-9#
CKTWAT   15-27     22-6#
CL$EPS    1-42     52-160
CLENUP    7-24      8-9#
CLKO1S   15-85     18-6#
CLKABD   18-107    28-6#
CLKCNT    1-88     15-8*     15-18     15-78     15-90     16-19     22-13     24-20     26-71
CLKDAT   15-12     16-13#
CLKINT    1-61     52-208    52-211
CLKIOH   18-103    19-9#
CLKJOB   17-6#     18-99
CLKPC     1-101    26-29     26-45
CLKPHN   29-33     30-10#
CLKPM    15-40     26-18#
```

Cross reference table (CREF V05.04)

```
CLKPS     1-101     26-27     26-42
CLKRUN    1-27       1-88     15-6#     15-91
CLKSCR   24-63      25-7#
CLSCDB    1-77       8-111
CLTOTL    1-46      18-117    52-158
CONFIG    1-58      14-39
CORUSR    1-56       7-6      10-186*   11-19     13-22     13-56     15-16     17-11     18-67     26-38
CP$STD    1-102     25-20
CQ$CP     1-77      25-20*
CQ$HOT    1-100     24-21*
CQ$JOB    1-88      25-21
CQ$LNK    1-76       1-86      1-100     24-13     24-15     24-32     24-32*    24-37*    24-49     24-51     24-53*    25-15
CQ$LOT    1-100     24-20*    24-24
CQ$PRI    1-97      25-32*
CQ$RNS    1-100     19-40     19-42     25-31*
CR        1-118#     3-27     51-15
CSHALC    1-66      52-35     52-190
CSHDEV    1-39      52-48
CSHDVN    1-39      52-50
CSHFIN    1-51       9-50
CSHINI    1-100     52-192
CSIARE    1-74       7-72
CTTBR     1-58      51-89*
CTTSR     1-58      51-87
CURCP     1-101      9-14*
CURRDB    1-68       8-128*
CURVC     1-85      11-21     11-44
CW$FPU    1-57      14-39
CXBOWN    1-47       8-60
CXTBAS    1-105      6-39
CXTPAG    1-79       5-50
CXTRMN    1-68       6-57      6-66      6-73      7-59
CXTWDS    1-105      6-40      6-45
D$RUN     1-108     13-17
D.FLAG    1-108      8-25*     13-17
DATIMH    1-89      16-24
DATIML    1-89      16-26     16-31
DBGTRP    1-94      13-60
DEQ       1-55      10-165
DFJMEM    1-71       5-48      6-132
DHCLOK    1-30      49-10#
DHGDSS    1-28      39-11#
DHLBIT    3-39#     42-17
DHSBRK    1-30      42-9#
DHSDSS    1-28      40-9#
DHSPCT    3-59#     41-22
DHSSPD    1-30      41-10#
DIEARG    1-37       7-8*     11-34*    11-46*    50-47
DIEBAS    3-5#       3-6       3-7       3-8       3-9       3-10      3-11      3-12      3-13      3-14      3-15      3-16
          3-17       3-18      3-19      3-20      3-21      3-22      3-23      50-40     50-125
DIEMSG    1-37       7-8*      7-74*     11-35*    11-46*    18-137*   50-39     50-124
DIEPC     1-37      50-34
DIESP     1-37      50-113
DLCLOK    1-29      47-9#
DLGDSS    1-28      31-11#
DLSBRK    1-27      33-9#
```

Cross reference table (CREF V05.04)

```
DLSDSS    1-28      32-9#
DLSSPD    1-29      34-9#
DLSTRT    1-77      47-36
DM$CSR    1-109     39-23*    39-24*    40-17*    40-18*
DM$LSR    1-109     39-25     40-24*    40-26*
DMPHND    1-38      50-25*    50-106*
DMPOVL    1-38      50-24*    50-96*
DMPTXT    1-38      50-126
DODUMP    1-37      50-129
DOSCHD    1-85      18-35*
DOTRMP    1-34       8-129*
DTLX      1-92      18-46     18-48*
DZ$7BT    1-48      38-21
DZ$8BT    1-48      38-19
DZ$LEN    1-48
DZ$ODD    1-48      38-27
DZ$PAR    1-48      38-24
DZCLOK    1-29      48-9#
DZGDSS    1-28      35-11#
DZSBRK    1-27      37-10#
DZSDSS    1-28      36-9#
DZSSPD    1-27      38-9#
DZSTRT    1-77      48-37
EM$DTL    1-102      3-6       3-6#      18-137
EM$FRK    3-7        3-7#
EM$JMO    3-8        3-8#
EM$KRE    3-9        3-9#      7-74
EM$KTP    3-10       3-10#     11-35
EM$LMF    3-11       3-11#
EM$MIO    3-12       3-12#
EM$MPR    3-13       3-13#
EM$NQE    3-14       3-14#
EM$NSP    3-15       3-15#
EM$PFT    3-16       3-16#
EM$RIT    3-17       3-17#     11-46
EM$SFO    3-18       3-18#
EM$SIE    3-19       3-19#
EM$SJN    3-21       3-21#      7-8
EM$SOF    3-23       3-23#
EM$SSE    3-20       3-20#
EM$UEI    3-22       3-22#
EMTBLK    1-36      10-94
EMTCAD    1-71       6-52*     8-24*
EMTCAS    1-96       6-52      8-24
EMTLEV    1-72       6-51*     7-12*
EMTRAD    1-71       8-22*
ENQTL     1-99      17-43     17-105    19-43     22-28     27-25
ERRLOC    1-72       7-40      7-51*
EXCINI    1-25      52-7#
EXEC      1-56      10-109    52-254
FORCEX    1-46      10-127    19-30
FORKIT    3-50#     24-12*    24-44*    24-59
FORKQ     1-110     15-64     15-73     15-86     24-64
FP$CDI    1-111     15-62
FP$CDO    1-111     15-71
FP$CK1    1-110     15-84
```

Cross reference table (CREF V05.04)

```
FP$IOF    1-52      24-67
FPTRPX    1-25      14-108*
FPUUSE    1-83       8-124*
FQ$$SZ    1-54      52-26
FQ$LNK    1-54      52-27*    52-30*
FQ$PRI    1-110     15-62*    15-71*    15-84*    24-62*
FQ$RTN    1-110     15-63*    15-72*    15-85*    24-63*
FRECXT    1-112      8-62
FREFRK    1-54      52-30     52-31*
FREIOQ    1-59      52-11
FREMEM    1-78      10-158
FRESPD    1-76       8-66
FREUSR    1-76       8-80
FRKGEN    1-54      52-22
FRKGET    1-110     15-61     15-70     15-83     24-61
FRKINI    1-54      52-24     52-31
FRKPRI    1-100     11-23
FSTDL     1-62      52-236
GETMEM    1-96       5-62
HANPAR    1-113     50-68
HF$7BT    1-49      41-40
HF$8BT    1-49      41-38
HF$LEN    1-49
HF$ODD    1-49      41-46
HF$PAR    1-49      41-43
HF$RIE    1-77       1-90     41-28     52-137
HF$TIE    1-77      52-137
HF$TSB    1-53
HLTCHR    51-12     51-16     51-18     51-39     51-59     51-63     51-65     51-87#    51-88
HLTOCT    50-35     50-40     50-78     50-92     50-114    51-30#
HLTPRT    50-30     50-41     50-46     50-76     50-90     50-94     50-104    50-112    51-7#     51-42     51-67
HLTRAD    50-97     50-107    51-54#
IB$IJ     1-35       6-83
IB$SF2    1-35       6-85
II$$SZ    1-44      54-49
II$FLG    1-44      54-29*
II$NAM    1-44      54-22*    54-23*    54-24*    54-25*
II$NPV    1-44      54-33*    54-39
II$PRV    1-43      54-34
ILSW2     1-62       5-91      6-109     10-98     10-105    28-14     29-29     52-222    52-226    53-26
INBSY     1-94      18-63     18-76
INISPD    52-217    53-6#
INITFL    1-63      52-250*
INITLN    1-27       5-73#    52-229    52-244
INRECV    1-80      47-13
INSINI    52-154    54-5#
INSTBL    1-44      54-10     54-17
INSTBN    1-44      54-12
INTLVL    1-57      11-25
INTMX1    1-52      52-131
INTPRI    1-63       7-10      9-27      10-187    24-58     25-16     25-44     34-22     39-26     40-30     41-53     43-28
          44-30     45-52     46-30     47-25
INVEC     1-80      47-13
IOHALT    1-75       8-45
IOQSIZ    1-59      52-14
IOSTOP    1-75
```

Cross reference table (CREF V05.04)

```
ITRMTP    1-69       6-110
JCDB      1-56       8-108      8-110
JM$$SZ    1-99      52-81
JM$LNK    1-89      52-82*     52-85*
JOBCCB    1-53       9-46       9-48
JSTK      1-72       6-138      7-9
JSTKND    1-72       6-53*
JSWLOC    1-73       7-52*
KILJOB    1-69      30-55
KMNBAS    1-73       7-47       7-69
KMNCHN    1-74       7-62
KMNPGS    1-79       5-51       5-53
KMNSTK    1-74       7-78
KMNSTR    1-74       7-88
KMNTOP    1-73       7-46       7-50       7-68
KPAR6     1-68       6-35*     26-70*
LABTIM    1-51      10-107*    28-19      28-21*
LACTIV    1-66       6-20*
LAFSIZ    1-67       6-22*
LBASE     1-46      10-156     10-161*
LBRKCH    1-67       6-26*      8-21*
LBRKCQ    1-67       6-25*      8-84       8-86*     20-29      20-40      20-42*
LBSPRI    1-79       5-78*
LCDTIM    1-81      30-28*     30-38*     30-68*
LCDTYP    1-60      29-22      30-14      53-29
LCLUNT    1-93       5-29      29-27
LCMPL     1-76       5-44*      9-28      19-38
LCOL      1-67       6-21*
LCONTM    1-68       6-30*
LCPUHI    1-67       1-88       6-28*     15-19*
LCPULO    1-68       1-88       6-29*     15-18*
LCXPAR    1-58       6-35
LCXTBL    1-36      10-112
LDKMON    6-142      7-30       7-37#
LEMTPC    1-102     26-55
LF        1-119#     3-27       3-27      51-17
LF$IN     1-83       8-32
LHIPCT    1-80       5-80*
LINBUF    1-65       5-68
LINCNT    1-65       5-73*
LINCUR    1-68       6-31*
LINNXT    1-65       5-69*
LINPNT    1-65       5-71*
LINSIZ    1-60       5-72
LINSPC    1-61       5-72*
LINSWT    1-41      10-70
LIOCNT    1-79       5-43*      9-20       9-30
LIOHLD    1-56      19-17      19-22*
LITIME    1-70       1-103      5-81*     17-51      17-53*     17-76      22-23      25-28      27-20
LJSW      1-65       6-11*      7-38*      7-39       7-53*
LMEMIN    1-55       5-58*     10-160*
LMINQ     1-92      18-29      18-31*
LMXLN     1-64      35-18      36-15      37-17      38-32      39-18      40-15      41-27      42-15      43-19      44-15      45-26
         46-15      48-15      52-116
LMXNUM    1-60      35-17      36-14      37-16      38-33      39-17      40-14      41-26      42-14      43-18      44-14      45-25
         46-14      48-14      52-113
```

Cross reference table (CREF V05.04)

```
LMXPRM   1-112    34-13*    38-13*    41-16*    45-15*    53-22
LNBLKS   1-46      5-42*     5-61*    10-157    10-159*
LNMAP    1-46     10-68     10-100*
LNPRIM   1-41      6-117    10-27     10-52     10-64     10-67     10-85*
LNSBLK   1-78      6-14*
LNSPAC   1-35     10-175*
LOFFTM   1-90      5-89*    10-97*    30-29*    30-59*    30-73     30-75*
LOGCHR   1-83      8-35      8-36
LOGCR    1-83      8-37
LOGFLG   1-83      8-32
LOGOFF   1-25     10-10#
LOKINI   1-70     52-180
LOTBUF   1-66      6-16
LOTNXT   1-66      6-17*
LOTPNT   1-66      6-18*
LOTSIZ   1-40      6-19     10-32
LOTSPC   1-40      6-19*    10-32
LP$7BT   1-47     38-17     41-36     45-35
LP$ODD   1-47     38-25     41-44     45-43
LP$PAR   1-47     38-22     41-41     45-40
LP$SPD   1-47     38-15     41-21     45-20
LPARNT   1-42     10-148    10-150*   10-174*
LPRG1    1-84     20-18
LPRG2    1-84     20-20
LPRI     1-79      8-79*    17-22     25-32
LPROG    1-67      6-23*     6-121     6-121*
LPROJ    1-67      5-32*     6-81      6-93*     6-120     6-120*
LQUAN    1-65      1-74      6-10*     7-83*    17-16*    17-21     17-44*    17-61    17-106*
LRDTIM   1-75      8-19*    21-13     21-15*
LRTCHR   1-86     21-18
LSCCA    1-67      6-24*     8-20*
LSECPT   1-41     10-55     10-77     10-119
LSLEPH   1-74      1-99      8-14*    22-14*
LSLEPL   1-74      1-99      8-15*    22-13*    22-17
LSPND    1-75      8-16*
LSTACT   1-65      5-70*
LSTATE   1-85     17-42     17-65     17-70     17-86     17-104    19-28     19-40    21-11    22-11    26-52    27-23
LSTDL    1-62      5-108     6-97     52-237
LSTHL    1-79     29-11     52-110    53-11
LSTMX    1-60     52-102    52-125
LSTPL    1-41      1-60      5-87      5-104     6-99      10-41     28-10     29-31    52-231   53-34
LSTSL    1-85     10-147    18-26     19-13     20-17     21-7      22-10     27-11
LSUCF    1-62     52-239
LSW      1-40      5-41*     5-64*     5-74*     6-9*      6-104*    7-29      7-31*    7-33*    8-29     8-38*    10-10
        10-16     10-28     10-50     10-65     10-126*   10-166*   14-11*    18-27    30-41    30-48    30-50
LSW10    1-90     10-34     49-14*
LSW11    1-35      5-77*     8-13*    53-36
LSW2     1-55      6-109*    6-118     6-118*    6-125     10-98*
LSW3     1-63      5-75*     5-93      6-111*    6-127*    7-32*     10-30*    10-36    10-173*  29-15    29-17    30-40*
        30-47*    30-64*    47-33     47-35*    48-32     48-34     48-36*    52-111   52-224   53-15    53-17    53-38*
LSW4     1-55      6-15*     7-14      7-41*    10-167*   12-14     13-23     26-36
LSW5     1-55      5-90*     5-95*    10-168*   30-45     47-20     47-27*    47-31    47-37*   48-22    48-26*   48-30
        48-38*
LSW6     1-55      8-17*    10-169*
LSW7     1-56      6-12*     7-42*    10-170*   27-16     27-18*
LSW8     1-104     6-13*
```

Cross reference table (CREF V05.04)

```
LSW9      1-59    5-76*   7-11*   7-25*   7-43*   8-18*   10-108*  10-124  12-28*  13-57   28-27   28-29*
         30-52
LTRMTP    1-69    6-110*  6-119   6-119*
LTTCR     1-36    6-27*   8-94    8-96*
MAPUSR    1-64   10-186*
MAXJOB    1-5     5-34
MAXMEM    1-71    6-134*
MAXSEC    1-41   10-58    10-81   10-120
MBFFLG    1-98   18-16    18-20*
MF$CAR    1-109  39-36
MF$DTR    1-109  39-42    40-24   40-26
MF$LIN    1-109  39-23    40-17
MF$RNG    1-109  39-30
MH$BRK    1-53   42-26*   42-31*
MH$LPR    1-53   41-32*
MH$SCR    1-52   41-51*   52-137*
MINCTR    1-92   18-42*   18-44*
MINTIM    1-92   18-45*
MONABT    1-64    8-74
MONDAY    3-48#  16-43
MONFQH    1-99   52-75
MRKTHD    1-86   24-13
MS$BRK    1-103  33-13    37-25   42-21   46-22
MS$CAR    1-106  30-33    31-28   35-31   39-38   43-40
MS$DTR    1-106  30-24    30-26   30-57   30-77   31-34   32-13   35-37   36-20   39-44   40-22   43-46
         44-22
MS$RNG    1-106  30-22    31-22   35-25   39-32   43-34
MSGABT    1-63    8-80
MSGINI    1-69   52-186
MVWDS     1-69    6-59
MXBRK     1-112  37-31*
MXCAR     1-82   35-29
MXCSR     1-82   48-20    48-24*  48-25*  52-126  52-131*
MXDTR     1-81   35-35    36-25*  36-30*
MXLBIT    3-44#  35-19    36-16   37-18   48-16
MXLNT     1-60    1-60    52-104  52-115
MXLPR     1-112  38-34*
MXRING    1-81   35-23
MXSBRK    1-112  37-23    37-32*
MXTYPE    1-52   52-128
NEDCDI    1-82   15-56
NEDCDO    1-82   15-65    18-119*  49-15*
NEDCLO    1-41   18-120*
NEDSOT    1-87   15-31    15-33*
NEWUSR    1-26    6-5#
NMSNMB    1-61   52-144
NPCCB     1-54    9-20     9-30
NSCP      1-56   52-65
NSIP      1-25    4-26#   4-31    4-31#   4-35    4-35#   4-39    4-39#   4-43    4-43#   4-47    4-47#
          4-52    4-52#   4-56    4-56#   4-60    4-60#   4-65    4-65#   4-69    4-69#   4-73    4-73#
          4-77    4-77#   4-81    4-81#   4-86    4-86#
NSPLDV    1-70   52-172
NUMCCB    1-66   52-37
NUMDEV    1-113  50-67
NUMFRK    1-54   52-22
NUMIOQ    1-59   52-12
```

Cross reference table (CREF V05.04)

```
NUMON     1-41      5-36      5-106*    10-92*
O.ADR     1-113     50-63
O.PAR     1-113     50-59
OUTBSY    1-94      18-61     18-74
OVRADD    1-113     50-58     50-84
OVRHC     1-64      8-35      8-36      8-37      8-49      8-55      8-62      8-66      8-70      8-74      8-80      8-118
          10-70     10-137    10-143    21-19     30-55     52-168    52-174    52-186    52-198    52-204
PO$BYP    1-42      4-50
PO$DBG    1-43      54-33
PO$LOK    1-43      4-84
PO$MEM    1-43      4-63
PO$NAM    1-42      4-50
PO$NFR    1-43
PO$NFW    1-43
PO$SPV    1-42      4-50
PO$SYS    1-42      4-50
PCCCR2    1-105     52-213
PF$IOW    1-101     26-49
PF$OVF    1-101     26-74
PF$SYS    1-101     26-24
PIDPTR    1-51      15-45
PLSINI    1-38      52-198
PLSOFF    1-106     10-143
PLSXIT    1-71      8-118
PMBASE    1-86      26-60     26-65
PMFLGS    1-102     26-24     26-49     26-74*
PMNBPC    1-86      26-67
PMPAR     1-87      26-70
PMRUN     1-56      10-180*   15-38
PMTOP     1-86      26-62
PMUSER    1-55      10-177    10-179*   26-35
PNAME     1-113     50-102
PROFLG    1-105     20-12     52-123    52-209
PROODC    1-106     3-51      15-50
PROTIM    3-51#     15-47*    15-50*
PSW       1-63      7-10*     7-13*     7-37*     9-19*     9-27*     10-187*   14-10*    24-14*    24-58*    25-12*    25-16*
          25-44*    34-17*    34-22*    39-22*    39-26*    40-16*    40-30*    41-50*    41-53*    43-24*    43-28*    44-20*
          44-30*    45-49*    45-52*    46-20*    46-30*    47-22*    47-25*
PVON      1-41      5-110*    10-46*    10-93*
Q.LINK    1-59      52-15*    52-18*
QCOMPL    1-99      20-43     25-36
QCPU      1-98      17-119
QFREE     1-76      5-123     6-91      8-89      8-99
QHIPRI    1-80      5-99
QNSPND    1-70      9-23
QUNSIG    1-103     17-29     17-36     17-55     17-80     17-90     17-99     17-114
R$CH17    1-73      7-60
R$MFMV    1-83      6-74
R$SWPC    1-34      6-67
R50CHR    51-58     51-62     51-64     51-75#
R50PRO    3-53#     20-18
R50PRT    3-54#     20-20
R50SAV    3-55#     54-25
RCBBAS    1-71      1-78
RCBEND    1-71      1-78
RCVDON    1-82      47-18
```

Cross reference table (CREF V05.04)

RDINT	1-82	47-23	47-24				
RDONE	1-82	48-20					
RIE	1-82	48-24	48-25				
RING	1-81	31-20					
RPAR	1-68	8-131*					
RPDR	1-68	8-132*					
RSR	1-80	31-16	32-18*	32-23*	47-18	47-23*	47-24*
RTSTOP	1-75	8-49					
S$$HIP	1-97	17-65					
S$$RT	1-97	17-70					
S$HICP	1-69	17-86	17-92	22-25	25-30		
S$INWT	1-85	21-11					
S$IOWT	1-86	9-22	19-28	26-52			
S$OTFN	1-87	27-19					
S$OTLO	1-87	27-22					
S$TMWT	1-97	22-11					
S$TWFN	1-97	22-27	25-27				
S$WSMB	1-98	18-21					
S150	1-56						
S9600	1-51	28-30	53-28				
SB$$SZ	1-61	52-146					
SB$LNK	1-61	52-147*	52-150*				
SCHED	1-54	9-37					
SCPFHD	1-57	52-63					
SETMAP	1-71	6-131	7-49				
SETSPD	1-51	28-31					
SHRRCB	1-101	52-55					
SHRRCN	1-101	52-56					
SLTSIZ	1-39	52-96					
SNMSHD	1-61	52-143					
SP$$SZ	1-61	52-67					
SP$LNK	1-48	52-68*	52-71*				
SPCPS	1-72	8-23*					
SPIJ	1-35	6-83*	10-48				
SPLINI	1-70	52-174					
SPSTAT	1-84	20-31*	20-35	20-44*	20-49*		
SPTXT	3-34#	50-111					
SRFEND	4-87#	54-50					
SRFPRG	4-27#	54-18					
SS	1-56	10-185					
SS$PRT	1-84	20-35	20-44	20-49			
SS$RUN	1-84	20-31	20-44	20-49			
STOP	1-26	1-64	7-6#	13-73			
STPFLG	1-80	30-20					
STRACT	1-92	21-19					
SUCF2	1-35	6-86					
SUTOP	1-73	7-48					
SWPCOT	1-85	18-33	18-36*				
SWPJOB	1-39	52-92					
SWPPOS	1-39	52-91					
SYNAME	1-43	54-22					
SYPNCR	1-59	24-38	24-43*	25-13	25-15*		
SYSDAT	1-85	16-33	16-61*				
SYSDIE	1-26	50-15#					
SYSHL1	1-37	11-39					
SYSHLT	1-63	1-96	7-8	7-74	11-46	18-137	

Cross reference table (CREF V05.04)

```
SYSXIT    1-47      12-30
SYTIMH    1-89      10-18     10-22     16-20*    16-24     16-32*
SYTIML    1-89      10-19     10-25     16-19*    16-26     16-31*
TIKO1S    3-52#     15-81*    18-12     18-124*
TJKCNT    1-88      15-6      15-90*
TK1CNT    1-89      15-78*    15-80*
TK1VAL    1-88      15-80
TK5CNT    1-89      18-111*   18-113*
TLCHK    18-114     29-6#
TMIDLH    1-96      18-95*
TMIDLL    1-96      18-94*
TMIOH     1-94      18-60*
TMIOL     1-94      18-59*
TMIOWH    1-95      18-83*    18-91*
TMIOWL    1-95      18-82*    18-90*
TMSWPH    1-95      18-66*
TMSWPL    1-95      18-65*
TMSWTH    1-96      18-85*
TMSWTL    1-96      18-84*
TMTOTH    1-94      18-56*
TMTOTL    1-94      18-55*
TMUSRH    1-95      18-70*
TMUSRL    1-95      18-69*
TOTON     1-55       5-103*   10-176*
TRBRK     1-103     33-18     33-23
TRMRDY    1-81      31-32     32-18     32-23
TRNSTR    1-78      10-31
TRPAR5    1-38      50-53     50-77
TRPBPT    1-26      12-10#
TRPCOM    1-25      11-29     11-60     11-62     12-37     13-11#
TRRDY     1-58      51-87
TSEXC2    1-10#
TSR       1-112     33-18*    33-23*    34-18     34-21*
TSXTX     1-25      11-15#
TXARG     3-28#     50-45
TXDEV     3-33#     50-103
TXFSE     3-27#     50-29
TXNUL     3-29#     51-41     51-66
TXOID     3-32#     50-93
TXPAR5    3-30#     50-75
TXSEQ     3-31#     50-89
UERSEV    1-72       7-18*
UFPTRP    1-57       8-123*   14-13     14-57     14-59*
UHIMEM    1-73       7-50*
UIOCNT    1-92      18-57     18-79     18-88
UMODE     1-57       1-102     7-87     11-16     11-79     13-15     13-43     14-58     26-27     26-42
UPMODE    1-57       7-13      7-37     11-79     13-43     14-10     14-58
UREGO     1-98      18-22
USRINI    1-59      52-168
USRJOB    1-76       8-53
UTRPAD    1-57       8-122*   11-56     11-84     11-85*
VDBFLG    1-47      12-26
VDMKTP    1-38      11-27
VECBAS    1-69       6-58
VF$7BT    1-50      45-39
VF$8BT    1-50      45-37
```

Cross reference table (CREF V05.04)

```
VF$BC     1-53      46-24     46-26
VF$DCD    1-108     43-38
VF$DTR    1-108     43-44     44-24     44-26
VF$EVN    1-50      45-45
VF$LEN    1-50
VF$PAR    1-50      45-42
VF$RIE    1-52       1-90     43-20     44-16     45-27     46-16     52-135
VF$RNG    1-108     43-32
VF$SC     1-53
VF$TIE    1-52      52-135
VH$CSR    1-52       1-107    43-25*    44-21*    45-50*    46-21*    52-135*
VH$LCR    1-46       1-107    43-26     44-24*    44-26*    46-24*    46-26*
VH$LPR    1-53      45-51*
VH$LSR    1-107     43-27
VHCLOK    1-30      49-9#
VHGDSS    1-29      43-11#
VHSBRK    1-30      46-9#
VHSDSS    1-29      44-9#
VHSPCT    3-63#     45-21
VHSSPD    1-30      45-9#
VINTIO    1-103      5-80
VMAXMC    1-69       8-78     52-184
VMXMON    1-99      52-76
VMXSF     1-70      52-178
VMXWIN    1-91      10-135    52-202
VOFFTM    1-90      10-96     30-29
VPAR5     1-38
VPAR6     1-86      26-69
VPLAS     1-78       8-116    10-141    52-196
VPRIDF    1-79       5-78      5-79
VPRIHI    1-93      17-23
VPRILO    1-93      17-32
VQUAN0    1-93      17-25     17-27
VQUAN1    1-98       5-81
VQUAN2    1-98      17-112
VQUAN3    1-93      17-34
VQUN1A    1-98      17-78
VQUN1B    1-99      17-97
VQUN1C    1-103     17-88
VSWPFL    1-73       5-59      7-44
VSWPSL    1-39      52-89
VSYDMP    1-37      50-118
VTMOUT    1-81      30-28     30-68
VUSPHN    1-34      30-43
WAKEUP   18-115     21-6#
WININI    1-91      52-204
WINREL    1-49      10-137
```

Cross reference table (CREF V05.04)