Table of contents

Table of contents

```
                                        .TITLE  TSINIT -- TSX startup initialization
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
 000000                                 .CSECT  TSINIT
 000000                         TSINIT:
                                ;
                                ;  There are two external assembly-time switches related to assembling
                                ;  TSINIT for execution on a PRO or a PDP-11.
                                ;
                                ;  The following values for the PROASM flag are defined:
                                ;   0 ==> Assemble for PDP-11 (not Pro) only.
                                ;   1 ==> Assemble for Pro only.
                                ;   2 ==> Assemble for either PDP-11 or Pro execution.
                                ;
                                ;  The following values for the PROCID flag are defined:
                                ;   0 ==> Do not lock system to ID number.
                                ;   1 ==> Lock system to ID number.
                                ;
                                        .IF     NDF,PROASM              ;If PROASM not defined
                                PROASM  =       0                       ;Default value for PROASM if not defined
                                        .ENDC   ;NDF,PROASM
                                ;
                                        .IF     NDF,PROCID              ;If PROCID not defined
                                        .IF     EQ,<PROASM-1>           ;If assembling for PRO only
                                PROCID  =       1                       ;Then check ID by default
                                        .IFF                            ;If not assembling for PRO only
                                PROCID  =       0                       ;Then don't check ID number
                                        .ENDC   ;EQ,<PROASM-1>
                                        .ENDC   ;NDF,PROCID
                                ;
                                        .IF     EQ,PROASM
                                        .GLOBL  TSXPRO
                000000          TSXPRO  =       0                       ;Define dummy base for TSXPRO if not PRO
                                        .ENDC
                                ;
                                ;-----------------------------------------------------------------------------
                                ;  TSINIT is the initialization module of TSX that is executed once
                                ;  during system startup.  Time-sharing character buffers and other
                                ;  run-time data areas are allocated over TSINIT.
                                ;
                                ;  Copyright 1980,1981,1982,1983,1984,1985,1986,1987,1988,1989.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, TN  USA
                                ;
                                ;
                                ;  Macro calls
                                ;
                                        .MCALL  .LOOKUP,.ENTER,.READW,.SAVESTATUS,.GVAL
                                        .MCALL  .TRPSET,.SETTOP,.CLOSE,.TTYOUT,.PRINT,.PURGE
                                        .MCALL  .DELETE,.WRITW,.SERR,.HERR,.EXIT,.UNLOCK
                                        .MCALL  .FETCH,.RELEASE,.LOCK,.GTIM,.DATE,.DSTATUS
                                        .MCALL  .SCCA,.CSTAT
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  TSINIT,INITGO,INITOP,PPTERM,PROITP,PROASM,PISRT
                                        .GLOBL  DSKBUF,PROBUF,FNDHRB,HANXMR
                                ;
                                ;  Following global only needed for the Pro distribution
                                ;  creation program - MAKPRO and installation program - INSTSX
                                ;
                                .IF     NE,PROASM
                                ;
                                ;** Assemble this code if we are generating for a Pro
                                ;
                                        .GLOBL  PROSIZ,PROINI,PROLIN,PROHAN,PRONOP
                                        .GLOBL  PIHAN,PIDPTR,PIDRIV
                                .IFF    ;NE,PROASM
                                ;
                                ;** Assemble this code if we are not generating for a Pro
                                ;
                                        .GLOBL  TSXPRO
        000000                  TSXPRO  =       0
                                ;
                                ;** End of conditional Pro code
                                ;
                                .ENDC   ;NE,PROASM

                                ;
                                ;  Global references
                                ;
                                        .GLOBL  HANDSK,MAXDEV,NDVRCB,HANRCB,HANRCO
                                        .GLOBL  LXCL,VSYDMP,STKLVL,INTSSZ,INDFIL,NXIVMH,EXCBUF
                                        .GLOBL  VNUIP,NSIP,INSTBL,INSTBN,II$$SZ,DCCSIZ,VNUMDC,NUMCDB
                                        .GLOBL  NSCP,SCPFHD,SP$$SZ,CSHDEV,CSHDVN,VMXCSH,CD$$SZ
                                        .GLOBL  LSTPL,LMXNUM,MXCSR,MXVEC,RSR,INVEC,VHIMEM,CXTPAG
                                        .GLOBL  LSWPBK,LSTSL,SWDBLK,SWPCHN,NUMDEV,CS$NMX,SCHED
                                        .GLOBL  H.DSTS,DVSTAT,HANENT,H.GEN,FORK,INTEN,PNAME
                                        .GLOBL  $SXON,LSW10,LHIRBB,LHIRBE,LHIRBP,LHIRBG,LHIRBA
                                        .GLOBL  LHIRBS,LHIRBC,VNCSLO,VNCXOF,VNCXON,SDDVU,VMSCHR,MAXSLO
                                        .GLOBL  MPARO,MPAR16,PARENL,MPARFL,TSEMT,VDBFLG,DX$EBA
                                        .GLOBL  H.SIZ,HANSIZ,H.DVSZ,DEVSIZ,LOMAP,MMENBL,UPAR7
                                        .GLOBL  PSW,HIMAP,FSTDL,LSTDL,LINBUF,LINSIZ,NUMCCB,TK1SEC
                                        .GLOBL  FRKINI,FRKGEN,NUMFRK,FQ$$SZ,H.CSR,H.INS,VSWPSL,DMYDEV
                                        .GLOBL  LINEND,LOTBUF,LOTSIZ,LOTEND,KMNTOP,KMNHI,NSL,NDL
                                        .GLOBL  DX$MPH,DX$NHM,DX$IBH,HANPAR,HANXIT,MAPPAR,LINSPC
                                        .GLOBL  KMNPGS,KMNSTK,KMNSTR,KMNCHN,SROMMR,KPARO,PROFLG
                                        .GLOBL  EMMAP,IOMAP,SR3MMR,IOPAGE,MAPSIZ,SR3FLG,NSPLDV
                                        .GLOBL  UDDRO,IDSFLG
                                        .GLOBL  UPARO,KPDRO,UPDRO,KPAR7,BASMAP,PTWRD,PTBYT,LOKMEM
                                        .GLOBL  GTBYT,MPPHY,RELOC,BRKPT,TSGEN,TSEXEC,VSWPFL
                                        .GLOBL  CW$GDH,CW$BTH,CW$LGS,CW$FB,CW$FGJ,MSGBAS,RPRVEC
                                        .GLOBL  CW$USR,CW$XM,CONFIG,CW$50H,JMPO,DTLX,USRBAS,WINBAS
                                        .GLOBL  DATIML,DATIMH,RMON,CONFG2,SG$ELG,SG$IOT,CSHBAS
                                        .GLOBL  SG$PAR,SG$MTS,SG$MMU,SG$MTM,LTTPAR,LOKBAS,CSHVEC,LOKVEC
                                        .GLOBL  SYSGEN,AUTHAN,AHEND,CLKRTI,TRP4,CW$PRO,TIOVEC
                                        .GLOBL  TRP10,TRP20,TRP24,EMTENT,TRP34,INIJMP,MHNSIZ
                                        .GLOBL  TK1VAL,INRECV,OTRECV,INMXV,OTMXV,DHBFSZ,MXTYPE
                                        .GLOBL  ZCLR,MXRBUF,MXDTR,INTMX1,$PHONE,LCDTYP,TIOBAS
                                        .GLOBL  LDHB1B,LDHB1P,LDHB2B,CLVERS,CXTSIZ,CXTWDS,CXTPDR
                                        .GLOBL  CLORSZ,TSXSIT,JM$$SZ,VMXMON,MONFQH,CXTRMN,CXTBAS
                                        .GLOBL  ILSW2,$NOIN,LSW3,MXLPR,CW$ESP,CLTOTL,RMNPDR,MA$SYS
                                        .GLOBL  SFCB,SFCBND,SFCBFH,SFCBSZ,NSPLFL,NSPLBL,INTSTK,INTSND
        .GLOBL  NFRESB,PVSPBL,VMXWIN,DW$$SZ,LDVERS,CW$QBS
        .GLOBL  FC$LBN,VMLBLK,VMXSF,VMXSFC,FF$$SZ,FW$$SZ,SWPJOB,SWPPOS
        .GLOBL  TSR,RBR,RDINT,LSTMX,SS,CHAIN,JSWLOC,MU$TXT,SLTSIZ
        .GLOBL  NUMIOQ,FREIOQ,UMODE,FPTRAP,MXLNT,DI$LD,DI$CL,CLSTS
        .GLOBL  FREPGS,IOQSIZ,SYUNIT,UMSYTP,DI$TT,CXTBUF,SSEND
        .GLOBL  SYINDX,MONVEC,KMNBAS,SDANAM,VBUSTP,MINCTR
        .GLOBL  NUMRDB,RDB,RDBEND,RT$SKP,RT$TOP,NLINES,SHRRCB,SHRRCN
        .GLOBL  RT$BAS,UPMODE,SPLNB,CSHALC,NIOL,CHNSIZ,RC$$SZ,VNGR
        .GLOBL  UPAR6,UPDR6,RT$$SZ,VINABT,$DEAD,LSW6
        .GLOBL  SYTIMH,SYSDAT,TRP250,ODTTRP,TRP14,SYTIML
        .GLOBL  DS$ABT,CL$ORB,CL$ORE,CL$ORG,CL$ORP,CL$ORA
        .GLOBL  $TAB,$FORM,CO$TAB,CO$FF,CO$DEF,CL$EPS,CLEOFS
        .GLOBL  CL$OPT,CL$STA,CL$ORS,LSTLIN,VCSHNB,CL$EPP,CL$EPN
        .GLOBL  CCLSAV,SPLND,SDCB,SPLDEV,SPLANM,MIODBG
        .GLOBL  SDNAME,SDCHAN,SDCBSZ,SPLDVN,DTYPE
        .GLOBL  DS$NRD,DX$NMT,$8BIT,CO$8BT,UEXRTN,VUXIFL
        .GLOBL  SPLBLK,SPLCHN,MVSIZ,MEMPAR,UEXINT,DX$NRD
        .GLOBL  NMSNMB,SNMSHD,SB$$SZ,VPMSIZ,PMPAR,PMCELS
        .GLOBL  NUMDCD,MEM256,LOKCSH,DC$$SZ
        .GLOBL  JCXPGS,MXJMEM,VDFMEM,DFJMEM,TK5VAL,TK3SVL
        .GLOBL  VPAR6,IOTIMR,ERRLOG,VNFCSH,FC$$SZ
        .GLOBL  O.ADR,O.BLK,O.PAR,O.SIZ,VPAR5,KPAR5,DZOINT,DHOINT
        .GLOBL  OVRADD,$OVRH,SYSMAP,MAPSYS,VSLEDT,LCLUNT
        .GLOBL  UBUSMP,UMRADR,IOMAP,QBUS,UNIBUS,DX$NST
        .GLOBL  DVFLAG,DX$DMA,RT$NAM,DS$DIR,LDDEVX,DS$VSZ
        .GLOBL  INDSAV,INDDBL,INDTSV,INDDBS,DS$SFN
        .GLOBL  SYNAME,UCLNAM,RSFBLK,VPLAS,SEGCHN
        .GLOBL  MXJADR,$MEMSZ,PHYMEM,SG$TSX,CDX$DH
        .GLOBL  CLHEAD,CLSIZE,CLDEVX,C.CSW,C.SBLK,C.DEVQ,C1DEVX
        .GLOBL  VU$CL,VUCLMC,UK$$SZ,US$$SZ,UC$$SZ,UCLBLK,UCLDAT
        .GLOBL  VLDSYS,VMXMSG,VMAXMC,MB$$SZ,MR$$SZ,CS$OPN,CS$ENT
        .GLOBL  DX$MAP,MIOFLG,MI$SBP,MI$LNK,MIOBHD,VMIOSZ
        .GLOBL  VMIOBF,MI$$SZ,MW$$SZ,MIONWB,MIOWHD,MW$LNK
        .GLOBL  CSHSIZ,CSHBFP,CA$BLK,CA$DVU,CA$WCT,VMXMRB
        .GLOBL  CA$UFL,CA$UBL,CA$HFL,CA$HBL,CA$HSH,NUMRDB
        .GLOBL  SRTSIZ,SMRSIZ,CCBHD,CC$$SZ,CDX$DZ,MF$LIN
        .GLOBL  CDX$DL,HF$TSB,MH$SCR,LMXLN,HF$LIN,HF$RIE,HF$TIE
        .GLOBL  MH$LPR,DM$CSR,MF$LE,DM$LSR,HF$MC,MF$CS,MF$CM
        .GLOBL  CDX$VH,VH$CSR,VH$LPR,MH$PBR,VF$TIE,VF$RIE,VF$MR
        .GLOBL  VF$LIN,VF$SC,VF$RE,VH$LCR,VHOINT,TTINCP
        .GLOBL  $HARD,LOUTIR,LINIR,NEDCHR,CLOTIR,CLINCP,FSTIOL,LSTHL
        .GLOBL  SYSVER,SYSUPD,DI$DU,DI$XL,DI$MU,CL$LIX
        .GLOBL  CL$LEN,DI$PI,GENTOP
        .GLOBL  LSW5,DX$NCA,KPAR6,CLKVEC
;-----------------------------------------------------------------------
;  Macros to enable and disable interrupts
;
        .MACRO  DISABL          ;Disable interrupts
        BIS     #340,@#PSW
        .ENDM   DISABL

        .MACRO  ENABL
        BIC     #340,@#PSW
        .ENDM   ENABL

;-----------------------------------------------------------------------
;  Offsets in block 0 of ODT REL file.
```

```
    172                                             ;
    173         000040                              STA     =       40              ;PROGRAM START ADDRESS
    174         000042                              STK     =       42              ;INITIAL STACK POINTER
    175         000052                              RSZ     =       52              ;ROOT SIZE
    176         000056                              OSZ     =       56              ;OVERLAY SIZE
    177         000060                              RID     =       60              ;REL FILE ID
    178         000062                              RBD     =       62              ;DISPLACEMENT TO 1ST REL BLOCK
    179         001000                              ODTBAS  =       1000            ;BASE ADDRESS ODT WAS LINKED FOR
    180                                             ;
    181                                             ;   Data areas
    182                                             ;
    183 000000                                      AREA:   .BLKW   8.
    184 000020  000000  000000  000000              NFSBLK: .WORD   0,0,0,0,0,0     ;EXTENDED TO 6 WORDS FOR .CSTAT
        000026  000000  000000  000000
    185 000034  000000                              ODTFLG: .WORD   0
    186 000036  000000                              ODTTOP: .WORD   0
    187 000040  000000                              CCAFLG: .WORD   0
    188 000042  000000                              CLK100: .WORD   0
    189 000044  000000                              RTTRP4: .WORD   0
    190 000046  000000                              RTMNVC: .WORD   0
    191 000050                                      SAVBLK: .BLKW   5
    192 000062  075250  100020  000000              TSXSAV: .RAD50  /SY TSX   SAV/
        000070  073376
    193 000072  075250  100003  051646              KMNNAM: .RAD50  /SY TSKMONSAV/
        000100  073376
    194 000102  075250  011504  000000              CCLNAM: .RAD50  /SY CCL   SAV/
        000110  073376
    195 000112  000000  000000  000000              DSTBLK: .WORD   0,0,0,0
        000120  000000
    196 000122  000000                              XMVBAS: .WORD   0
    197 000124  000000                              NMXHAN: .WORD   0
    198 000126  000000                              HMAP:   .WORD   0
    199 000130  000000                              FETDEV: .WORD   0
    200 000132  000000                              TOPMEM: .WORD   0
    201 000134  000000                              FMEMHI: .WORD   0               ;64-byte block # below high alloc memory
    202 000136  000000                              FMEMLO: .WORD   0               ;64-byte block # above top of low alloc memory
    203 000140  000000                              OVLBAS: .WORD   0               ;Start loading overlays over TSINIT from here
    204 000142  000000                              FILBLK: .WORD   0
    205 000144  000000                              CURDEV: .WORD   0
    206 000146  000000                              CURNAM: .WORD   0
    207 000150  000000                              PROBUF: .WORD   0
    208 000152  030074'                             WRKBUF: .WORD   INITOP
    209 000154  004000                              WRKSIZ: .WORD   2048.
    210 000156  000322'                             RTVPTR: .WORD   RTVEND          ;Initially no limit on system version number
    211                                             ;   The following RAD50 definitions were converted from word cells to
    212                                             ;   definitions to conserve space at V6.40.  All subsequent references to
    213                                             ;   one of these cells are converted from:
    214                                             ;   relative (cmp R50xxx,r1) to immediate (cmp #R50xxx,r1).
    215         052077                              R50MSG  = ^RMSG
    216         110466                              R50WIN  = ^RWIN
    217         046543                              R50LOK  = ^RLOK
    218         103112                              R50USR  = ^RUSR
    219         012700                              R50CSH  = ^RCSH
    220         077167                              R50TIO  = ^RTIO
    221         100040                              R50TT   = ^RTT          ;"TT "
    222         075250                              R50SY   = ^RSY          ;"SY "
    223         045640                              R50LD   = ^RLD          ;"LD "
```

```
        062550                    R50PI   = ^RPI          ;"PI "
        012240                    R50CL   = ^RCL          ;"CL "
        012276                    R50CL0  = ^RCL0
        012305                    R50CL7  = ^RCL7
        013630                    R50C1   = ^RC1          ;"C1 "
        013666                    R50C10  = ^RC10
        013675                    R50C17  = ^RC17
        105610                    R50VM   = ^RVM          ;"VM "
        046770                    R50LS   = ^RLS          ;"LS "
        057164                    R50ODT  = ^RODT
000160  100040  015270  075250    SKPDEV: .RAD50  /TT DK SY CL C1 PI    /
000166  012240  013630  062550
000174  000000
000176     000     110            GTLIN:  .BYTE   0,110
000200  075250  114730  000000    HANNAM: .RAD50  /SY XXX   TSX/
000206  100020
000210  075250  075273  057164    ODTBLK: .RAD50  /SY SYSODTREL/
000216  070524
000220  075250  035164  000000    INDNAM: .RAD50  /SY IND   SAV/
000226  073376
000230  000000                    RLBF:   .WORD   0
000232  000000                    RLBFND: .WORD   0
000234  000000                    ODTSTA: .WORD   0
000236  000000                    MEMLIM: .WORD   0
000240  000000                    HGENFL: .WORD   0
                                  ;
                                  ;   Initialization configuration word
                                  ;
000242  000000                    ICONFG: .WORD   0               ;Initialization configuration word
                                  ;
                                  ;   Flag bits in ICONFIG
                                  ;
        000001                    EXTLSI  =       1               ;Q-bus system with more than 256Kb
                                  ;
                                  ;   Simulated shared run-time control block for PI handler
                                  ;
000244  075250  062550  000000    PISRT:  .RAD50  /SY PI    TSX/
000252  100020
000254  000000  000000  000000            .WORD   0,0,0
                                  ;
                                  ;   Byte data cells
                                  ;
000262     000                    PPTERM: .BYTE   0               ;1 if printer port is T/S terminal
                                          .EVEN
```

```
                                        ;-----------------------------------------------------------------------
                                        ;  The following table is used to identify the supporting RT-11 monitor
                                        ;  version number for those features which depend on it.
                                        ;
                                        ;  The DL, XL and MU handlers require a minimum supporting RT-11 version.
                                        ;  The CL version number emulates the supporting RT-11 XL's $$$VER.
                                        ;  The LD translation table format changed at RT-11 V5.4.
                                        ;
                                        ;  The format of the table is:
                                        ;
          000000                                RT$VER  = 0     ;Major system version number
          000001                                RT$UPD  = 1     ;Minor system version number (update number)
          000002                                CL$VER  = 2     ;Emulated XL version number
          000003                                RTV$SZ  = 3     ;Size of a version table entry
                                        ;
 000264                                 RTVER:                  ;V4.0 is the earliest supported version of RT-11
 000264       004      000      377     RT40:   .BYTE   4,0,-1                  ;4.0 did not support XL
 000267       005      000      377     RT50:   .BYTE   5,0,-1                  ;5.0 did not support XL
 000272       005      001      020     RT51:   .BYTE   5,1,16.
 000275       005      035      020     RT51X:  .BYTE   5,35,16.                ;Another flavor of 5.1
 000300       005      006      020     RT51B:  .BYTE   5,6,16.
 000303       005      044      020     RT51C:  .BYTE   5,44,16.
 000306       005      002      021     RT52:   .BYTE   5,2,17.
 000311       005      003      021     RT53:   .BYTE   5,3,17.
 000314                                 RTVDEF:                                 ;Default emulation version
 000314       005      004      022     RT54:   .BYTE   5,4,18.
 000317       005      005      022     RT55:   .BYTE   5,5,18.
 000322                                 RTVEND:
                                                .EVEN
```

```
                                ;----------------------------------------------------------------------------
                                ;  The following tables are used to determine the minimum RT-11 monitor
                                ;  and update versions required for particular devices.
                                ;  There are three arguments for each handler definition:
                                ;   Arg 1 = Handler id code.
                                ;   Arg 2 = Ptr to minimum acceptable RT-11 version and update entry.
                                ;
                                        .MACRO  HANVER  DEVID,MNVPTR
                                        .BYTE   DEVID           ;ID code for device type
                                        .BYTE   0               ;Unused filler entry
                                        .WORD   MNVPTR          ;Minimum RT-11 version and update label
                                        .ENDM   HANVER
                                ;
                                ;  Define offsets into handler version table
                                ;
          000000                HV$ID   =       0               ;Handler identification code
          000001                HV$DMY  =       1               ;Unused entry
          000002                HV$VER  =       2               ;Minimum RT-11 version
          000004                HV$$SZ  =       4               ;Size of handler version table entry
                                ;
                                ;  Define minimum versions for various handlers
                                ;
000322                          HVTBL:
000322                                  HANVER  DI$DU,RT50      ;DU - (5.0)
000326                                  HANVER  DI$XL,RT51B     ;XL - (5.1B)
000332                                  HANVER  DI$MU,RT54      ;MU - (5.4)
000336                          HVEND:
```

```
                        ;------------------------------------------------------------------
                        ;   The following table defines default control flags for certain devices.
                        ;
        000000          DV$NAM  =       0               ;Rad50 name of device
        000002          DV$FLG  =       2               ;Flags for device
        000004          DV$$SZ  =       4               ;Size of a table entry
                        ;
                                .MACRO  DEFFLG  DEV,FLAGS
                                .RAD50  /'DEV/          ;DV$NAM
                                .WORD   FLAGS           ;DV$FLG
                                .ENDM   DEFFLG
                        ;
000336                  DVFLBS:
000336                          DEFFLG  <CR>,<DX$MPH>
000342                          DEFFLG  <CT>,<DX$MPH>
000346                          DEFFLG  <DB>,<DX$DMA!DX$MPH>
000352                          DEFFLG  <DD>,<DX$NHM>
000356                          DEFFLG  <DL>,<DX$DMA!DX$MPH!DX$IBH>
000362                          DEFFLG  <DM>,<DX$DMA!DX$NHM>
000366                          DEFFLG  <DP>,<DX$DMA>
000372                          DEFFLG  <DS>,<DX$DMA>
000376                          DEFFLG  <DT>,<DX$DMA>
000402                          DEFFLG  <DU>,<DX$DMA!DX$NHM!DX$NST>
000406                          DEFFLG  <DW>,<DX$MPH>
000412                          DEFFLG  <DX>,<DX$MPH>
000416                          DEFFLG  <DY>,<DX$DMA!DX$NHM>
000422                          DEFFLG  <DZ>,<DX$MPH>
000426                          DEFFLG  <FW>,<DX$DMA>
000432                          DEFFLG  <LP>,<DX$MPH>
000436                          DEFFLG  <LS>,<DX$MPH>
000442                          DEFFLG  <MM>,<DX$DMA!DX$MPH!DX$IBH>
000446                          DEFFLG  <MS>,<DX$DMA!DX$MPH!DX$IBH>
000452                          DEFFLG  <MT>,<DX$DMA!DX$MPH!DX$IBH>
000456                          DEFFLG  <MU>,<DX$DMA!DX$NHM!DX$IBH!DX$NST>
000462                          DEFFLG  <NL>,<DX$MPH>
000466                          DEFFLG  <PC>,<DX$MPH>
000472                          DEFFLG  <RF>,<DX$DMA>
000476                          DEFFLG  <RK>,<DX$DMA!DX$MPH>
000502                          DEFFLG  <VM>,<DX$EBA!DX$NCA!DX$NHM>
000506                          DEFFLG  <XC>,<DX$MPH>
000512                          DEFFLG  <XL>,<DX$MPH>
000516                  DVFLND:
```

```
                        ;---------------------------------------------------------------------
                        ;  The following data structures are used to hold information about
                        ;  TSX-Plus overlays as they are being initialized.
                        ;
                        ;  Offsets in structure for each overlay
                        ;
          000000        OS$SIZ  =       0               ;Total space needed for overlay
          000002        OS$FLG  =       2               ;0==>Load into XM space, 1==>over TSINIT
          000004        OS$OVL  =       4               ;Pointer to overlay table entry
          000006        OS$$SZ  =       6               ;Size of each overlay entry
                        ;
          000031        MAXOVL  =       25.             ;Maximum number of system overlays
                        ;
000516                  OSTABL: .BLKB   OS$$SZ*MAXOVL   ;Reserve room for table
000744                  OSEND:                          ;-Define end of table
000744    000516'       OSLAST: .WORD   OSTABL          ;Pointer past last used entry in table
                        ;
                        ;  Table of system overlays that must be loaded over TSINIT
                        ;
000746    012700        LOWOVL: .RAD50  /CSH/           ;TSCASH
000750    077167                .RAD50  /TIO/           ;TSTIO
000752    046543                .RAD50  /LOK/           ;TSLOCK
000754                  LOWEND:                         ;End of table
```

```
                                        ;------------------------------------------------------------
                                        ;   Text messages
                                        ;
                                                .NLIST  BEX
  000754    077     124     123     TSXHD:  .ASCII  /?TSX-F-/<200>
  000764    111     156     166     BADLIN: .ASCII  'Invalid CSR for T/S line: '<200>
  001017    111     156     166     BDVMSG: .ASCII  'Invalid vector for T/S line: '<200>
  001055    114     151     156     BDLMSG: .ASCII  /Line # = /<200>
  001067    000                     CRLF:   .BYTE   0
  001070    124     123     130     REQMIS: .ASCII  /TSX generation did not include device /<200>
  001137    103     141     156     BADOPN: .ASCIZ  /Cannot open program swap file/
  001175    103     141     156     RSFERR: .ASCIZ  /Cannot open PLAS region swap file/
  001237    116     165     155     CONSPC: .ASCII  /Number of contiguous blocks needed = /<200>
  001305    103     141     156     BDSPOP: .ASCII  /Cannot open spooled device: /<200>
  001342    111     156     163     BOSF:   .ASCIZ  /Insufficient disk space for spool file/
  001411    103     141     156     NOKMON: .ASCIZ  /Cannot find "SY:TSKMON.SAV" file/
  001452    103     141     156     NOCCL:  .ASCIZ  /Cannot find "SY:CCL.SAV" file/
  001510    103     141     156     CFHMSG: .ASCII  /Cannot find device handler file: /<200>
  001552    105     162     162     ERHMSG: .ASCII  /Error reading device handler file: /<200>
  001616    111     156     166     ERHNDV: .ASCII  /Invalid RT-11 version for device handler:/<200>
  001670    111     156     166     NOCSRR: .ASCII  /Invalid CSR for device: /<200>
  001721    105     162     162     ERHINS: .ASCII  /Error executing installation code for device: /<200>
  002000    124     123     130     TSXRUN: .ASCIZ  /TSX is already running/
  002027    110     141     156     HSGER.  .ASCII  /Handler not generated with XM support: /<200>
  002077    103     157     155     NOCLOK: .ASCIZ  /Computer line time clock (50 or 60 Hz) is not working/
  002165    116     157     040     NXMMSG: .ASCIZ  /No memory management hardware/
  002223    116     157     040     NEXMSG: .ASCIZ  /No extended memory management hardware/
  002272    103     141     156     NOODT:  .ASCIZ  /Cannot locate "SY:SYSODT.REL" file/
  002335    115     141     160     HN2BIG: .ASCII  /Mapped handler is larger than 8KB: /<200>
  002401    105     162     162     ODTRDM: .ASCIZ  /Error on read of SYSODT rel file/
  002442    110     141     156     NOSYDV: .ASCIZ  /Handler for SY device was not loaded/
  002507    107     145     156     TOOBIG: .ASCIZ  /Generated TSX system is too large/
  002551    122     145     144     REDUCE: .ASCII  /Reduce size of TSGEN by /<200>
  002602    056     040     142     BYTES:  .ASCIZ  /. bytes/
  002612    111     156     163     PHSOVF: .ASCIZ  /Insufficient total physical memory for generated system/
  002702    103     141     156     COSRT:  .ASCII  /Cannot open shared run-time file: /<200>
  002745    103     141     156     SVERR:  .ASCIZ  /Cannot locate "SY:TSX.SAV"/
  003000    111     156     163     TSXSIZ: .ASCIZ  /Insufficient memory to load all mapped system regions/
  003066    105     162     162     RDERR:  .ASCIZ  /Error reading "SY:TSX.SAV"/
  003121    111     156     163     SRTOVF: .ASCIZ  /Insufficient memory to load all shared run-time systems/
  003211    111     156     163     CSHOVF: .ASCIZ  /Insufficient memory space for data cache/
  003262    103     141     156     INDOPN: .ASCIZ  /Cannot open TSXIND file/
  003312    103     141     156     UCLOPN: .ASCIZ  /Cannot open TSXUCL data file/
  003347    040     101     102     R50CHR: .ASCII  / ABCDEFGHIJKLMNOPQRSTUVWXYZ$.*0123456789/
                                                .EVEN
                                                .LIST   BEX
                                        .SBTTL  * * *  TSX Initialization  * * *
                                        .SBTTL  * * * Initialization taking over control from RT-11 * * *
                                ;-----------------------------------------------------------------------
                                ;  The initialization code from this point onward takes over
                                ;  control from RT-11.
                                ;  This code is placed at the front of TSINIT so that non-initialized
                                ;  data structures can be allocated over it.
                                ;
003420                          TAKOVR:
                                ;
                                ;  Read in system overlays that go over TSINIT
                                ;
003420  004737  004260'                 CALL    INIOVL          ;Read overlays over TSINIT
                                ;
                                ;  Set pointer to monitor offset vector
                                ;
003424  012737  000000G 000000G         MOV     #MONVEC,@#RMON  ;SET POINTER TO MONVEC TABLE
                                ;
                                ;  Initialize last word in interrupt stack area so we won't report a
                                ;  stack overflow if an interrupt occurs.
                                ;  Set STKLVL to 0 to cause INTEN not to switch to interrupt
                                ;  stack during initialization.
                                ;
003432  012777  123456  000000G         MOV     #123456,@INTSND ;Say stack has not overflowed
003440  105037  000000G                 CLRB    STKLVL          ;Say we are already on interrupt stack
                                ;
                                ;  If we are running on a Professional, disable its interrupts
                                ;
                                .IF     NE,PROASM
                                        TSTB    PROFLG          ;Are we running on a PRO?
                                        BEQ     7$              ;Br if not on a PRO
                                        CALL    PRONOP          ;Disable its interrupts
                                        BR      5$              ;Ignore unexpected interrupts on PRO
                                .ENDC   ;NE,PROASM
                                ;
                                ;  Set up vectors to catch unexpected interrupts
                                ;  Note: We encode the interrupt vector address in the PS --
                                ;  the low-order two bits of the address are dropped (they are
                                ;  always zero) and the remainder of the address is encoded in the
                                ;  PS fields priority (high-order 3 bits) and n-z-v-c (low-order 4 bits).
                                ;
003444  012702  000000G         7$:     MOV     #UEXINT,R2      ;SEND UNEXPECTED INTERRUPTS TO THIS ROUTINE
003450  012700  000044                  MOV     #44,R0          ;120 ENCODED IN PS FIELDS
003454  105737  000000G                 TSTB    VUXIFL          ;ARE WE TO IGNORE UNEXPECTED INTERRUPTS?
003460  001004                          BNE     10$             ;BR IF NOT
003462  012702  000000G         5$:     MOV     #UEXRTN,R2      ;ROUTINE TO GO TO TO IGNORE INTERRUPT
003466  012700  000340                  MOV     #340,R0         ;SET PRIO=7 IN PS
003472  012701  000120          10$:    MOV     #120,R1         ;INIT ALL VECTORS STARTING AT 120
003476  010221                  1$:     MOV     R2,(R1)+        ;SET PC FOR INTERRUPT
003500  010021                          MOV     R0,(R1)+        ;SET PS FOR INTERRUPT (ENCODED ADDRESS VALUE)
003502  105737  000000G         6$:     TSTB    VUXIFL          ;ARE WE TO IGNORE UNEXPECTED INTS?
003506  001411                          BEQ     2$              ;BR IF YES
003510  105737  000000G                 TSTB    PROFLG          ;IS THIS A PRO?
003514  001006                          BNE     2$              ;BR IF YES
003516  005200                          INC     R0              ;ADVANCE ENCODED ADDRESS
003520  032700  000020                  BIT     #20,R0          ;DID WE CARRY INTO "T"-FIELD?
003524  001402                          BEQ     2$              ;BR IF NOT
```

```
003526  062700  000020                  ADD     #20,R0          ;FORCE CARRY OUT OF T-FIELD AND INTO PRIO FIELD
003532  020127  000420          2$:     CMP     R1,#420         ;DONE ALL INTERRUPT VECTORS OF INTEREST?
003536  103757                          BLO     1$              ;BR IF NOT
003540  010237  000060                  MOV     R2,@#60         ;CATCH CONSOLE TERMINAL VECTOR TOO
003544  012737  000014  000062          MOV     #14,@#62        ;ENCODED 60
003552  010237  000064                  MOV     R2,@#64
003556  012737  000015  000066          MOV     #15,@#66        ;ENCODED 64
                                ;
                                ;  Direct clock interrupt to a dummy RTI instruction to avoid it causing
                                ;  trouble during the initialization process when things aren't set up
                                ;  and ready to go.
                                ;
003564  012700  000340          11$:    MOV     #340,R0         ;PRIORITY 7 PS
003570  012737  000000G 000000G         MOV     #CLKRTI,@#CLKVEC;Send clock interrupt to RTI instruct for now
003576  010037  000002G                 MOV     R0,@#CLKVEC+2
                                ;
                                ;  Take over traps, EMT, BPT, etc.
                                ;
003602  005001                          CLR     R1              ;Start at location 0
003604  012721  000137                  MOV     #137,(R1)+      ;[JMP @#JMP0] ==> 0
003610  012721  000000G                 MOV     #JMP0,(R1)+     ;CATCH JUMPS TO LOCATION 0
003614  012721  000000G                 MOV     #TRP4,(R1)+     ;TRAP 4
003620  005021                          CLR     (R1)+
003622  012721  000000G                 MOV     #TRP10,(R1)+    ;TRAP 10
003626  005021                          CLR     (R1)+
003630  012721  000000G                 MOV     #TRP14,(R1)+    ;TRAP 14 (BREAKPOINTS)
003634  010021                          MOV     R0,(R1)+
003636  012721  000000G                 MOV     #TRP20,(R1)+    ;IOT TRAP
003642  005021                          CLR     (R1)+
003644  012721  000000G                 MOV     #TRP24,(R1)+    ;POWER FAIL
003650  010021                          MOV     R0,(R1)+
003652  012721  000000G                 MOV     #EMTENT,(R1)+   ;EMT
003656  005021                          CLR     (R1)+
003660  012721  000000G                 MOV     #TRP34,(R1)+    ;TRAP
003664  005021                          CLR     (R1)+
003666  012737  000000G 000114          MOV     #MEMPAR,@#114   ;MEMORY PARITY TRAP
003674  010037  000116                  MOV     R0,@#116
003700  012737  000000G 000244          MOV     #FPTRAP,@#244   ;TRAP 244 -- FLOATING POINT TRAP
003706  010037  000246                  MOV     R0,@#246        ;Enter FPU trap at priority 7
003712  012737  000000G 000250          MOV     #TRP250,@#250   ;TRAP 250 -- MEMORY MANAGEMENT TRAP
003720  005037  000252                  CLR     @#252
                                ;
                                ;  Initialize the system mapped region.
                                ;
003724  010546                          MOV     R5,-(SP)        ;SAVE THE CURRENT CONTENTS OF R5
003726  012705  000006                  MOV     #6,R5           ;INITIALIZE TO THE FIRST REGION
003732  004737  000000G                 CALL    MAPSYS          ;CALL THE SYSTEM MAPPING ROUTINE
003736  012605                          MOV     (SP)+,R5        ;RESTORE THE PREVIOUS CONTENTS OF R5
                                ;
                                ;  Set up Unibus mapping if needed
                                ;
003740  004737  004616'                 CALL    SETUMP          ;SET UP UNIBUS MAPPING
                                ;
                                ;  Initialize time-sharing lines.
                                ;
003744  004737  005234'                 CALL    LININI          ;INIT LINES & SET UP VECTORS
                                ;
                                ;  Enable memory management
                                ;  (The kernel-mode mapping registers are already set up)
                                ;
003750  052737  000000G 000000G         BIS     #MMENBL,@#SR0MMR;Turn on memory management
003756  105737  000000G                 TSTB    SR3FLG          ;Does machine have memory management reg 3?
003762  001415                          BEQ     4$              ;Br if register does not exist (no ext. mem.)
003764  023727  000000G 010000          CMP     PHYMEM,#4096.   ;Does machine have at least 256Kb phys memory?
003772  103411                          BLO     4$              ;Br if not
003774  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;Set extended memory on
004002  105737  000000G                 TSTB    MEM256          ;Will TSX-Plus use at least 256Kb?
004006  001403                          BEQ     4$              ;Br if not
004010  052737  000000G 000000G         BIS     #IOMAP,@#SR3MMR ;Turn on 22-bit memory management for I/O
                                ;
                                ;  Initialize the memory allocation table
                                ;
004016  013737  000000G 000000G 4$:     MOV     MAPPAR,@#KPAR5  ;Map to memory allocation table
004024  013702  000000G                 MOV     LOMAP,R2        ;Point to 1st user-page entry
004030  105022                  8$:     CLRB    (R2)+           ;Say page is free
004032  020237  000000G                 CMP     R2,HIMAP        ;Done all user pages?
004036  103774                          BLO     8$              ;Loop if not
004040  112712  000000G                 MOVB    #MA$SYS,(R2)    ;Set flag marking start of system pages
                                ;
                                ;  Set up I/O device interrupt vectors.
                                ;
004044  004737  004706'                 CALL    DEVVEC          ;SET UP DEVICE INTERRUPT VECTORS
                                ;
                                ;  If we are running on a Professional, initiailize the PI handler
                                ;
                                .IF     NE,PROASM
                                        TSTB    PROFLG          ;Are we running on a Professional?
                                        BEQ     3$              ;Br if not
                                        CALL    PROHAN          ;Initialize the PI handler
                                        CALL    PIDVEN          ;Make device table entry for PI
                                .ENDC   ;NE,PROASM
                                ;
                                ;  Initialize interrupt stack area
                                ;
004050  013702  000000G         3$:     MOV     INTSND,R2       ;Point to base of stack area
004054  012700  123456                  MOV     #123456,R0      ;Get initialization value
004060  010022                  12$:    MOV     R0,(R2)+        ;Initialize the interrupt stack area
004062  020237  000000G                 CMP     R2,INTSTK       ;Finished?
004066  103774                          BLO     12$             ;Loop if not
004070  112737  177777  000000G         MOVB    #-1,STKLVL      ;Say we are not running on interrupt stack
                                ;
                                ; Enter TSEXEC to complete initialization
                                ;
004076  000137  000000G                 JMP     INIJMP          ;ENTER INITIALIZATION ROUTINE IN TSEXEC
                                ;
                                ;  Abort the initialization
                                ;
004102  013737  000042' 000000G INISTP: MOV     CLK100,@#CLKVEC ;Restore RT-11 clock vector
004110  013737  000044' 000004          MOV     RTTRP4,@#4      ;Restore trap 4 vector
004116  013737  000046' 000000G         MOV     RTMNVC,@#RMON   ;Restore RT-11 monitor pointer
004124                          9$:     .EXIT                   ;RETURN TO RT-11
```

```
                                                .SBTTL  LODINI -- Load a segment over TSINIT
                                        ;-----------------------------------------------------------------------------
                                        ;  LODINI is called to read an overlay segment over TSINIT.
                                        ;
                                        ;  Inputs:
                                        ;   R2 = Pointer to OSTABL entry for segment to be loaded.
                                        ;   R5 = 64-byte block # where segment is to be loaded.
                                        ;
 004126  010146                         LODINI: MOV     R1,-(SP)
 004130  010346                                 MOV     R3,-(SP)
 004132  010446                                 MOV     R4,-(SP)
                                        ;
                                        ;  Get pointer to linker-built overlay entry
                                        ;
 004134  016201  000004                         MOV     OS$OVL(R2),R1   ;Get pointer to linker-built table
                                        ;
                                        ;  Determine how much code to read from the segment
                                        ;
 004140  016204  000000                         MOV     OS$SIZ(R2),R4   ;Get # 64-byte blks allocated for segment
 004144  072427  000005                         ASH     #5,R4           ;Convert to # words
 004150  020461  000000G                        CMP     R4,O.SIZ(R1)    ;Compare with original segment code size
 004154  101402                                 BLOS    1$              ;Br if segment was truncated by init
 004156  016104  000000G                        MOV     O.SIZ(R1),R4    ;Get code size
                                        ;
                                        ;  Read the segment into memory
                                        ;
 004162  010503                         1$:     MOV     R5,R3           ;Get 64-byte block #
 004164  072327  000006                         ASH     #6,R3           ;Convert to byte address
 004170                                         .READW  #AREA,#17,R3,R4,O.BLK(R1)
 004224  103406                                 BCS     10$             ;Br if error on read
                                        ;
                                        ;  Store the physical address of the segment into the overlay descriptor
                                        ;
 004226  010561  000000G                        MOV     R5,O.PAR(R1)    ;Remember physical address of segment
                                        ;
                                        ;  Finished
                                        ;
 004232  012604                                 MOV     (SP)+,R4
 004234  012603                                 MOV     (SP)+,R3
 004236  012601                                 MOV     (SP)+,R1
 004240  000207                                 RETURN
                                        ;
                                        ;  Error on read
                                        ;
 004242                                 10$:    .PRINT  #TSXHD
 004250                                         .PRINT  #RDERR
 004256                                         .EXIT
```

```
                                           .SBTTL  INIOVL -- Load system overlays over TSINIT
                                  ;----------------------------------------------------------------------
                                  ;  INIOVL is called to load into memory those system overlays that
                                  ;  are to be placed over the TSINIT code.
                                  ;
                                  ;  Inputs:
                                  ;   Overlay segment information is in OSTABL.
                                  ;
 004260  010246                   INIOVL: MOV     R2,-(SP)
 004262  010546                           MOV     R5,-(SP)
                                  ;
                                  ;  Initialize pointer to start of memory area for overlays
                                  ;
 004264  013705  000140'                  MOV     OVLBAS,R5       ;Start of area for overlays
 004270  072527  177772                   ASH     #-6,R5          ;Convert to 64-byte #
 004274  042705  176000                   BIC     #176000,R5      ;Clear possible propagated sign bits
                                  ;
                                  ;  Begin loop to load each overlay that goes over TSINIT
                                  ;
 004300  012702  000516'                  MOV     #OSTABL,R2      ;Point to 1st overlay segment entry
 004304  005762  000002           1$:     TST     OS$FLG(R2)      ;Does this segment go over TSINIT?
 004310  001406                           BEQ     2$              ;Br if not
 004312  004737  004470'                  CALL    KEYSEG          ;Remember base of some segments
 004316  004737  004126'                  CALL    LODINI          ;Load the segment
 004322  066205  000000                   ADD     OS$SIZ(R2),R5   ;Advance memory pointer
 004326  062702  000006           2$:     ADD     #OS$$SZ,R2      ;Point to entry for next segment
 004332  020237  000744'                  CMP     R2,OSLAST       ;Finished all segments?
 004336  103762                           BLO     1$              ;Loop if not
                                  ;
                                  ;  Initialize entry point vector for TSTIOX segment
                                  ;
 004340  013702  000000G                  MOV     TIOBAS,R2       ;Get addr of base of TSTIOX
 004344  072227  000006                   ASH     #6,R2           ;Convert to byte address
 004350  012705  000000G                  MOV     #TIOVEC,R5      ;Point to entry point vector
 004354  004737  004432'                  CALL    ENTVEC          ;Set up entry point vector
                                  ;
                                  ;  Initialize entry point vector for TSCASH segment
                                  ;
 004360  013702  000000G                  MOV     CSHBAS,R2       ;Get addr of base of TSCASH
 004364  001406                           BEQ     3$              ;Br if TSCASH not loaded
 004366  072227  000006                   ASH     #6,R2           ;Convert to byte address
 004372  012705  000000G                  MOV     #CSHVEC,R5      ;Point to entry point vector
 004376  004737  004432'                  CALL    ENTVEC          ;Set up entry point vector
                                  ;
                                  ;  Initialize entry point vector for TSLOCK segment
                                  ;
 004402  013702  000000G          3$:     MOV     LOKBAS,R2       ;Get addr of base of TSLOCK
 004406  001406                           BEQ     9$              ;Br if TSLOCK not loaded
 004410  072227  000006                   ASH     #6,R2           ;Convert to byte address
 004414  012705  000000G                  MOV     #LOKVEC,R5      ;Point to entry point vector
 004420  004737  004432'                  CALL    ENTVEC          ;Set up entry point vector
                                  ;
                                  ;  Finished
                                  ;
 004424  012605                   9$:     MOV     (SP)+,R5
 004426  012602                           MOV     (SP)+,R2
 004430  000207                           RETURN
```

```
                                  .SBTTL  ENTVEC -- Set up entry point vector for overlay
                          ;-----------------------------------------------------------------------
                          ;  ENTVEC is called to set up addresses in an entry point vector for
                          ;  overlay segments such as TSCASH that are loaded at addresses different
                          ;  from where they are linked.
                          ;
                          ;  Inputs:
                          ;   R2 = Address of base of segment.
                          ;   R5 = Pointer to vector that is to be initialized (word with -1 terminates)
                          ;
004432  010246            ENTVEC: MOV     R2,-(SP)
004434  010446                    MOV     R4,-(SP)
004436  010546                    MOV     R5,-(SP)
004440  010204                    MOV     R2,R4           ;Get addr of base of segment
004442  062704  000004            ADD     #4,R4           ;Point to start of vector in segment
004446  005722                    TST     (R2)+           ;Get value to use to relocate offsets
004450  012415            1$:     MOV     (R4)+,(R5)      ;Get offset to entry point within segment
004452  060225                    ADD     R2,(R5)+        ;Convert to absolute address
004454  005715                    TST     (R5)            ;Any more words to initialize?
004456  001774                    BEQ     1$              ;Br if yes
                          ;
                          ;  Finished
                          ;
004460  012605                    MOV     (SP)+,R5
004462  012604                    MOV     (SP)+,R4
004464  012602                    MOV     (SP)+,R2
004466  000207                    RETURN
```

```
                                         .SBTTL  KEYSEG -- Remember memory position of system overlays
                                ;--------------------------------------------------------------------------
                                ;  KEYSEG is called to remember the physical memory position of some
                                ;  key system overlay segments.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to segment entry in OSTABL overlay table.
                                ;   R5 = Base 64-byte block physical memory for segment.
                                ;
004470  010446                  KEYSEG: MOV     R4,-(SP)
                                ;
                                ;  Get the name of the segment out of the linker-built segment block
                                ;
004472  016200  000004                  MOV     OS$OVL(R2),R0   ;Point to linker-built entry
004476  016004  000000G                 MOV     O.ADR(R0),R4    ;Get the name of the segment
                                ;
                                ;  See if this is a segment whose address we want to remember
                                ;
004502  020427  052077                  CMP     R4,#R50MSG      ;Is this the TSMSG segment?
004506  001003                          BNE     1$              ;Br if not
004510  010537  000000G                 MOV     R5,MSGBAS       ;Remember base of TSMSG segment
004514  000436                          BR      8$
004516  020427  110466          1$:     CMP     R4,#R50WIN      ;Is this the TSWIN segment?
004522  001003                          BNE     3$              ;Br if not
004524  010537  000000G                 MOV     R5,WINBAS       ;Remember base of TSWIN segment
004530  000430                          BR      8$
004532  020427  103112          3$:     CMP     R4,#R50USR      ;Is this the TSUSR segment?
004536  001003                          BNE     4$              ;Br if not
004540  010537  000000G                 MOV     R5,USRBAS       ;Remember base of TSUSR segment
004544  000422                          BR      8$
004546  020427  046543          4$:     CMP     R4,#R50LOK      ;Is this the TSLOCK segment?
004552  001003                          BNE     5$              ;Br if not
004554  010537  000000G                 MOV     R5,LOKBAS       ;Remember base of TSLOCK segment
004560  000414                          BR      8$
004562  020427  012700          5$:     CMP     R4,#R50CSH      ;Is this the TSCASH segment?
004566  001003                          BNE     6$              ;Br if not
004570  010537  000000G                 MOV     R5,CSHBAS       ;Remember base of TSCASH segment
004574  000406                          BR      8$
004576  020427  077167          6$:     CMP     R4,#R50TIO      ;Is this the TSTIOX segment?
004602  001003                          BNE     8$              ;Br if not
004604  010537  000000G                 MOV     R5,TIOBAS       ;Remember base of module
004610  000400                          BR      8$
                                ;
                                ;  Finished
                                ;
004612  012604                  8$:     MOV     (SP)+,R4
004614  000207                          RETURN
```

```
                                                .IF     NE,<PROASM-1>   ;Assemble for PDP-11
                                                .SBTTL  SETUMP -- Set up Unibus mapping if needed
                                        ;-----------------------------------------------------------------------------
                                        ;  SETUMP is called to set up the Unibus map registers for 11/44 and
                                        ;  11/70 systems which more than 256Kb of memory.
                                        ;  If Unibus mapping is needed, the Unibus map registers # 0-4 are
                                        ;  initialized for a 1-to-1 mapping with the low 40Kb of memory
                                        ;  so that I/O to system buffers in the low memory area can be done without
                                        ;  having to do Unibus mapping.
                                        ;
                                        ;  Outputs:
                                        ;   UBUSMP:  1==>Do Unibus mapping;  0==>Don't do Unibus mapping.
                                        ;
004616  010246                          SETUMP: MOV     R2,-(SP)
004620  010346                                  MOV     R3,-(SP)
004622  013746  000004                          MOV     @#4,-(SP)       ;SAVE TRAP VECTOR
004626  012737  004674' 000004                  MOV     #9$,@#4         ;CATCH TRAPS
                                        ;
                                        ;  See if this is a type of maching that needs unibus mapping
                                        ;
004634  105737  000000G                         TSTB    UBUSMP          ;Is UNIBUS mapping needed?
004640  001415                                  BEQ     9$              ;Br if not
                                        ;
                                        ;  Unibus mapping is needed
                                        ;  Load unibus map registers # 0-4 to point to low 48Kb of memory.
                                        ;
004642  012705  000000G                 2$:     MOV     #UMRADR,R5      ;POINT TO CONTROL REGISTER FOR UNIBUS MAP 0
004646  005004                                  CLR     R4              ;START MAPPING TO BOTTOM OF MEMORY
004650  012700  000005                          MOV     #5,R0           ;LOAD 5 MAP REGISTERS
004654  010425                          1$:     MOV     R4,(R5)+        ;SET LOW-ORDER VALUE IN MAP REGISTER
004656  005025                                  CLR     (R5)+           ;CLEAR HIGH-ORDER VALUE IN MAP REGISTER
004660  062704  020000                          ADD     #8192.,R4       ;ADVANCE MEMORY ADDRESS
004664  077005                                  SOB     R0,1$           ;LOOP IF MORE MAP REGISTERS TO LOAD
                                        ;
                                        ;  Turn on Unibus mapping
                                        ;
004666  052737  000000G 000000G                 BIS     #IOMAP,@#SR3MMR ;ENABLE UNIBUS MAPPING
                                        ;
                                        ;  Finished
                                        ;
004674  012637  000004                  9$:     MOV     (SP)+,@#4       ;RESTORE TRAP VECTOR
004700  012605                                  MOV     (SP)+,R5
004702  012604                                  MOV     (SP)+,R4
004704  000207                                  RETURN
                                                .IFF    ;NE,<PROASM-1>  ;Following code for Pro-only assembly
                                        ;
                                        ;  Define dummy SETUMP routine for Pro
                                        ;
                                        SETUMP: RETURN
                                                .ENDC   ;NE,<PROASM-1>
                                        .SBTTL  DEVVEC -- Set up device vectors
                               ;-----------------------------------------------------------------------
                               ;  DEVVEC is called to set up device interrupt vectors for handlers
                               ;  that have been loaded.
                               ;
 004706  010146                DEVVEC: MOV      R1,-(SP)
 004710  010346                        MOV      R3,-(SP)
 004712  010546                        MOV      R5,-(SP)
 004714  013746  000000G               MOV      @#KPAR5,-(SP)    ;Save PAR 5 mapping
                               ;
                               ;  Begin loop to set up vectors for each device
                               ;
 004720  012701  000002                MOV      #2,R1            ;Get index # of 1st device after TT
 004724  016103  000000G       1$:     MOV      HANENT(R1),R3    ;Get handler entry point address
 004730  020327  000006                CMP      R3,#6            ;Is this a real device?
 004734  101436                        BLOS     6$               ;Br if not
                               ;
                               ;  See if we need to map PAR 5 to this handler
                               ;
 004736  016100  000000G               MOV      HANPAR(R1),R0    ;Get PAR 5 base for this handler
 004742  001402                        BEQ      2$               ;Br if this is not a mapped handler
 004744  010037  000000G               MOV      R0,@#KPAR5       ;Map PAR 5 to the handler
                               ;
                               ;  Clear CQE and LQE in handler header
                               ;
 004750  005023                2$:     CLR      (R3)+            ;Clear LQE (4th word in handler)
 004752  005013                        CLR      (R3)             ;Clear CQE (5th word in handler)
 004754  162703  000010                SUB      #10,R3           ;Point to 1st word of handler
                               ;
                               ;  Set up interrupt vectors for this handler
                               ;
 004760  005005                        CLR      R5               ;Assume vector base address is 0
 004762  005713                        TST      (R3)             ;Any vectors to set up?
 004764  001422                        BEQ      6$               ;Br if no vectors to set up
 004766  002403                        BLT      5$               ;Br if multiple-vector
 004770  004737  005060'               CALL     SETVEC           ;Set up the vector
 004774  000416                        BR       6$               ;Finished
                               ;
                               ;  Multiple vectors.
                               ;
 004776  012300                5$:     MOV      (R3)+,R0         ;Get offset to list of vector info
 005000  006300                        ASL      R0               ;Get byte offset to vector list
 005002  060003                        ADD      R0,R3            ;Get absolute address of vector table
 005004  005713                        TST      (R3)             ;Is this a PRO device with floating vectors?
 005006  002005                        BGE      7$               ;Br if not
 005010  005723                        TST      (R3)+            ;Point to word with device ID
 005012  012346                        MOV      (R3)+,-(SP)      ;Get device ID
 005014  004777  000000G               CALL     @RPRVEC          ;Get base vector location for device
 005020  012605                        MOV      (SP)+,R5         ;This is base of vector locations
 005022  004737  005060'       7$:     CALL     SETVEC           ;Set up the vector
 005026  005713                        TST      (R3)             ;Any more vectors to set up?
 005030  003374                        BGT      7$               ;Br if yes
                               ;
                               ;  See if there are more devices to set up.
                               ;
 005032  062701  000002        6$:     ADD      #2,R1            ;Advance device table index
 005036  020137  000000G               CMP      R1,NUMDEV        ;More to do?
```

```
005042  101730                                  BLOS    1$                   ;Br if yes
                                        ;
                                        ;  Finished
                                        ;
005044  012637  000000G                         MOV     (SP)+,@#KPAR5
005050  012605                                  MOV     (SP)+,R5
005052  012603                                  MOV     (SP)+,R3
005054  012601                                  MOV     (SP)+,R1
005056  000207                                  RETURN
```

```
                                 .SBTTL  SETVEC -- Set up an interrupt vector for a device
                         ;-----------------------------------------------------------------------
                         ;  SETVEC is called to set up one interrupt vector for a device.
                         ;
                         ;  Inputs:
                         ;   R1 = Device index number.
                         ;   R3 = Pointer into device handler to 3 word cells:
                         ;         1. Address of interrupt vector.
                         ;         2. Offset to interrupt entry point in handler.
                         ;         3. PS for interrupt.
                         ;   R5 = Base address to add to vector locations.
                         ;
                         ;  Outputs:
                         ;   R3 = Points beyond 3 word info block in handler.
                         ;
                         ;  Size of interrupt catching routine compiled for interrupts to
                         ;  mapped handlers:
                         ;
        000032           MPIVSZ  =       26.             ;Amt of code compiled for mapped ints
                         ;
005060  010446           SETVEC: MOV     R4,-(SP)
                         ;
                         ;  See if this is a mapped handler
                         ;
005062  005761  000000G          TST     HANPAR(R1)      ;Is this a mapped handler
005066  001010                   BNE     1$              ;Br if yes
                         ;
                         ;  This is an unmapped handler.
                         ;  Vector interrupts directly to the handler.
                         ;
005070  012300                   MOV     (R3)+,R0        ;Get address of interrupt vector
005072  060500                   ADD     R5,R0           ;Add base address to vector location
005074  010310                   MOV     R3,(R0)         ;Store address of cell in handler
005076  062320                   ADD     (R3)+,(R0)+     ;Add offset to interrupt entry point
005100  012310                   MOV     (R3)+,(R0)      ;Set PS for interrupt
005102  052710  000340           BIS     #340,(R0)       ;Make sure priority = 7
005106  000450                   BR       9$
                         ;
                         ;  This is a mapped handler.
                         ;  Vector the interrupt to a routine that performs the following functions:
                         ;   1. Save the current PAR 5 mapping.
                         ;   2. Map PAR 5 to the handler.
                         ;   3. Push a dummy PC and PS on stack that will send return from handler
                         ;      to a routine that will restore the PAR 5 mapping.
                         ;   4. Jump into the handler interrupt entry point.
                         ;
005110  013704  000122'  1$:     MOV     XMVBAS,R4       ;Point to area where we store interrupt rtn
005114  012300                   MOV     (R3)+,R0        ;Get address of interrupt vector
005116  060500                   ADD     R5,R0           ;Add base address to interrupt location
005120  010420                   MOV     R4,(R0)+        ;Direct interrupt to our routine
005122  012724  013746           MOV     #013746,(R4)+   ; [ MOV @#KPAR5,-(SP) ]
005126  012724  000000G          MOV     #KPAR5,(R4)+
005132  012724  012737           MOV     #012737,(R4)+   ; [ MOV #par5val,@#KPAR5 ]
005136  016124  000000G          MOV     HANPAR(R1),(R4)+
005142  012724  000000G          MOV     #KPAR5,(R4)+
005146  012724  012746           MOV     #012746,(R4)+   ; [ MOV #340,-(SP) ]
005152  012724  000340           MOV     #340,(R4)+
005156  012724  012746                      MOV     #012746,(R4)+   ; [ MOV #HANXIT,-(SP) ]
005162  012724  000000G                     MOV     #HANXIT,(R4)+
005166  012724  000257                      MOV     #000257,(R4)+   ; [ CCC - Clear all condition codes ]
005172  016314  000002                      MOV     2(R3),(R4)      ; [ SEx - Set condition codes specified in PS]
005176  042714  177760                      BIC     #^C17,(R4)
005202  052724  000260                      BIS     #260,(R4)+
005206  012724  000137                      MOV     #000137,(R4)+   ; [ JMP @#handler_entry ]
005212  010314                              MOV     R3,(R4)         ; Store address of int entry point
005214  062324                              ADD     (R3)+,(R4)+
005216  012310                              MOV     (R3)+,(R0)      ;Set PS for interrupt entry
005220  052710  000340                      BIS     #340,(R0)       ;Make sure priority = 7
                                    ;
                                    ;   Save address beyond end of compiled interrupt catcher routine
                                    ;
005224  010437  000122'                     MOV     R4,XMVBAS       ;Save address beyond end of routine
                                    ;
                                    ;   Finished
                                    ;
005230  012604                      9$:     MOV     (SP)+,R4
005232  000207                              RETURN
```

```
.IF     NE,PROASM
        .SBTTL  PIDVEN -- Make device table entry for PI device
;-----------------------------------------------------------------------------
;  If we are running on a Professional computer, PIDVEN is called to
;  make an entry in the device tables for the PI device.
;
PIDVEN: MOV     R1,-(SP)
;
;  Increase number of defined devices and get device table entry index
;  to use for the PI device.
;
        ADD     #2,NUMDEV       ;One more device
        MOV     NUMDEV,R1       ;Get device table index
;
;  Set up information about the PI device
;
        MOV     #R50PI,PNAME(R1) ;Set device name
        MOV     #<DS$SFN!DI$PI>,DVSTAT(R1) ;Set device status flags
        CLR     DVFLAG(R1)      ;Clear other flags
        CLR     DEVSIZ(R1)      ;Clear device size
        MOV     #PIHAN+6,HANENT(R1);Set handler entry point (4th word)
        MOV     #PROSIZ,HANSIZ(R1) ;Set handler size
;
;  Finished
;
        MOV     (SP)+,R1
        RETURN
.ENDC   ;NE,PROASM
```

```
                                        .IF     NE,<PROASM-1>   ;If assembling for PDP-11
                                        .SBTTL  LININI -- Initialize time-sharing lines
                                ;----------------------------------------------------------------------
                                ;  LININI is called to initialize the time-sharing lines.
                                ;  This consists of setting up interrupt vectors and setting control
                                ;  flags in the status registers.
                                ;
005234  010146                  LININI: MOV     R1,-(SP)
005236  010246                          MOV     R2,-(SP)
005240  010346                          MOV     R3,-(SP)
005242  010446                          MOV     R4,-(SP)
005244  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set up interrupt vectors for DL11 lines
                                ;
005246  012701  000002                  MOV     #2,R1           ;Index for 1st line
005252  012704  000340                  MOV     #340,R4         ;Priority 7 PS
005256  032761  000000G 000000G 1$:     BIT     #$DEAD,LSW3(R1) ;Is this line uninstalled?
005264  001027                          BNE     8$              ;Br if yes
005266  032761  000000G 000000G         BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
005274  001423                          BEQ     8$              ;Br if not
005276  026127  000000G 000000G         CMP     LCDTYP(R1),#CDX$DL ;Is this a DL11 line?
005304  001017                          BNE     8$              ;Br if not
                                ;
                                ;  DL-11 line
                                ;
005306  016105  000000G                 MOV     INVEC(R1),R5    ;GET ADDRESS OF INPUT VECTOR
005312  012702  000000G                 MOV     #INRECV,R2      ;END OF RECEIVING VECTOR
005316  012703  177772G                 MOV     #<OTRECV-6>,R3  ;START OF INPUT INTERRUPT ENTRY POINTS
005322  010100                          MOV     R1,R0           ;GET LINE NUMBER
005324  006300                          ASL     R0              ;4 BYTES PER INPUT INTERRUPT ENTRY POINT
005326  160002                          SUB     R0,R2           ;GET ADDRESS OF INPUT INTERRUPT ENTRY POINT
005330  060100                          ADD     R1,R0           ;6 BYTES PER OUTPUT INTERRUPT ENTRY POINT
005332  060003                          ADD     R0,R3           ;GET ADDRESS OF OUTPUT INTERRUPT ENTRY POINT
005334  010225                          MOV     R2,(R5)+        ;SET PC FOR INPUT INTERRUPT ENTRY POINT
005336  010425                          MOV     R4,(R5)+        ;SET PS FOR INPUT INTERRUPT
005340  010325                          MOV     R3,(R5)+        ;SET PC FOR OUTPUT INTERRUPT
005342  010425                          MOV     R4,(R5)+        ;SET PS FOR OUTPUT INTERRUPT
                                ;
                                ;  Try next line
                                ;
005344  062701  000002          8$:     ADD     #2,R1           ;ADVANCE LINE INDEX NUMBER
005350  020127  000000G                 CMP     R1,#LSTHL       ;MORE TO DO?
005354  101740                          BLOS    1$              ;BR IF YES
                                ;
                                ;  Initialize multiplexers.
                                ;
005356  012701  000000G         SETMUX: MOV     #LSTMX,R1       ;Get last mux index #
005362  001423                          BEQ     SETLIN          ;Br if there are no mux lines
005364  026127  000000G 000000G 3$:     CMP     MXTYPE(R1),#CDX$DZ ;Is this a DZ11, DH11, or DHV11?
005372  001412                          BEQ     1$              ;Br if DZ11
005374  026127  000000G 000000G         CMP     MXTYPE(R1),#CDX$VH ;Is this a DHV11?
005402  001003                          BNE     2$              ;Br if not
005404  004737  006154'                 CALL    VHINIT          ;Initialize a DHV11
005410  000405                          BR      4$
005412  004737  006104'         2$:     CALL    DHINIT          ;Initialize a DH11
005416  000402                          BR      4$
```

```
005420  004737  005714'          1$:     CALL    DZINIT             ;Initialize a DZ11
005424  162701  000002           4$:     SUB     #2,R1              ;More to enable?
005430  001355                           BNE     3$                 ;Br if yes
                                 ;
                                 ;  Enable all lines
                                 ;
005432  012701  000000G          SETLIN: MOV     #LSTHL,R1          ;INDEX # OF LAST REAL LINE
005436  032761  000000G 000000G  4$:     BIT     #$DEAD,LSW3(R1)    ;IS THIS LINE INSTALLED?
005444  001057                           BNE     2$                 ;BR IF NOT
005446  032761  000000G 000000G          BIT     #$HARD,LSW3(R1)    ;Is this line connected to hardware?
005454  001453                           BEQ     2$                 ;Br if not
005456  032761  000000G 000000G          BIT     #$PHONE,ILSW2(R1);IS THIS A DIAL-UP LINE?
005464  001403                           BEQ     3$                 ;BR IF NOT
005466  052761  000000G 000000G          BIS     #$NOIN,LSW3(R1)    ;IGNORE INPUT TILL DIAL UP OCCURS
005474  016105  000000G          3$:     MOV     LCDTYP(R1),R5      ;Get comm device type code
005500  016100  000000G                  MOV     LMXNUM(R1),R0      ;IS THIS A DL-11 OR MUX LINE?
005504  001423                           BEQ     1$                 ;BR IF DL-11
005506  020527  000000G                  CMP     R5,#CDX$DZ         ;Is this a DZ11 or DH11?
005512  001411                           BEQ     6$                 ;Br if DZ11
005514  020527  000000G                  CMP     R5,#CDX$VH         ;Is this a DH11 or DHV11?
005520  001403                           BEQ     7$                 ;Br if DHV11
                                 ;
                                 ;  DH11 line
                                 ;
005522  004737  005626'                  CALL    DHLPRM             ;Set line parameters for DH11 line
005526  000426                           BR      2$
                                 ;
                                 ;  DHV11 line
                                 ;
005530  004737  005664'          7$:     CALL    VHLPRM             ;Set line parameters for DHV11 line
005534  000423                           BR      2$
                                 ;
                                 ;  DZ-11 line
                                 ;
005536  016102  000000G          6$:     MOV     LMXLN(R1),R2       ;Get line # within mux group
005542  052702  017030                   BIS     #017030,R2         ;Set line enable flags
005546  010270  000000G                  MOV     R2,@MXLPR(R0)      ;Enable the line
005552  000414                           BR      2$
                                 ;
                                 ;  DL-11 line
                                 ;
005554  016102  000000G          1$:     MOV     TSR(R1),R2         ;ADDRESS OF TRANSMITTER STATUS REGISTER
005560  011203                           MOV     (R2),R3            ;CLEAR TRANSMITTER STATUS REGISTER
005562  005012                           CLR     (R2)
005564  016102  000000G                  MOV     RBR(R1),R2         ;ADDRESS OF RECEIVER BUFFER REGISTER
005570  011203                           MOV     (R2),R3            ;CLEAR RECEIVER BUFFER REGISTER
005572  016102  000000G                  MOV     RSR(R1),R2         ;ADDRESS OF RECEIVER STATUS REGISTER
005576  005012                           CLR     (R2)
005600  012712  000000G                  MOV     #RDINT,(R2)        ;ENABLE RECEIVER INTERRUPTS
                                 ;
                                 ;  Do next line
                                 ;
005604  162701  000002           2$:     SUB     #2,R1
005610  003312                           BGT     4$
                                 ;
                                 ;  Finished
                                 ;
```

```
005612  012605                          MOV     (SP)+,R5
005614  012604                          MOV     (SP)+,R4
005616  012603                          MOV     (SP)+,R3
005620  012602                          MOV     (SP)+,R2
005622  012601                          MOV     (SP)+,R1
005624  000207                          RETURN
```

```
                                                .SBTTL  DHLPRM -- Set line parameters for a DH11 line
                                        ;------------------------------------------------------------------------------
                                        ;  DHLPRM is called to set up the line parameters for a DH11 line.
                                        ;
                                        ;  Inputs:
                                        ;   R1 = Physical line index number.
                                        ;
005626                                  DHLPRM:
                                        ;
                                        ;  Enable DM11 for this line
                                        ;
005626  016100  000000G                         MOV     LMXNUM(R1),R0   ;Get mux index number
005632  005760  000000G                         TST     DM$CSR(R0)      ;Does this DH11 have DM11 modem control?
005636  001411                                  BEQ     2$              ;Br if not
005640  142770  000000G 000000G                 BICB    #MF$LIN,@DM$CSR(R0) ;Clear line # field in DM11 CSR
005646  156170  000000G 000000G                 BISB    LMXLN(R1),@DM$CSR(R0);Select line of interest
005654  012770  000000G 000000G                 MOV     #MF$LE,@DM$LSR(R0);Enable the line
                                        ;
                                        ;  Finished
                                        ;
005662  000207                          2$:     RETURN
```

```
                                        .SBTTL  VHLPRM -- Set line parameter values for DHV11 line
                                ;-----------------------------------------------------------------------------
                                ;  Set the line parameter values for a DHV11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;
005664                          VHLPRM:
                                ;
                                ;  Enable the line
                                ;
005664  016100  000000G                 MOV     LMXNUM(R1),R0   ;Get mux index number
005670  042770  000000G 000000G         BIC     #VF$LIN,@VH$CSR(R0) ;Clear line # field in mux CSR
005676  156170  000000G 000000G         BISB    LMXLN(R1),@VH$CSR(R0) ;Set our line #
005704  012770  000000G 000000G         MOV     #<VF$RE>,@VH$LCR(R0) ;Enable the line
                                ;
                                ;  Finished
                                ;
005712  000207                          RETURN
```

```
                                                .SBTTL  DZINIT -- Initialize a DZ11 multiplexer
                                        ;---------------------------------------------------------------------------
                                        ;  DZINIT is called to initialize a DZ11 multiplexer.
                                        ;
                                        ;  Inputs:
                                        ;   R1 = Mux index number.
                                        ;
005714                                  DZINIT:
                                        ;
                                        ;  See if this DZ11 is installed
                                        ;
005714  005761  000000G                         TST     MXCSR(R1)       ;Is this DZ-11 installed?
005720  001416                                  BEQ     4$              ;Br if not
                                        ;
                                        ;  Set up interrupt vector connections for this MUX
                                        ;
005722  004737  005760'                         CALL    MUXVEC          ;Set up interrupt vectors for this DZ11
                                        ;
                                        ;  Start up the mux operation
                                        ;
005726  052771  000000G 000000G                 BIS     #ZCLR,@MXCSR(R1);Do master clear on DZ-11
005734  032771  000000G 000000G 1$:             BIT     #ZCLR,@MXCSR(R1);Wait for clear to finish
005742  001374                                  BNE     1$
005744  017100  000000G                 2$:     MOV     @MXRBUF(R1),R0  ;Clear silo
005750  100775                                  BMI     2$
005752  105071  000000G                         CLRB    @MXDTR(R1)      ;Disable all data sets
                                        ;
                                        ;  Finished
                                        ;
005756  000207                          4$:     RETURN
```

```
                                         .SBTTL  MUXVEC -- Set up interrupt vectors for a multiplexer
                                 ;-----------------------------------------------------------------------
                                 ;  MUXVEC is called to set up the interrupt vector connections for
                                 ;  a DZ11, DH11, or DHV11 multiplexer.
                                 ;
                                 ;  Inputs:
                                 ;   R1 = Mux index number.
                                 ;
 005760  010246                  MUXVEC: MOV     R2,-(SP)
 005762  010346                          MOV     R3,-(SP)
 005764  010546                          MOV     R5,-(SP)
                                 ;
                                 ;  Set interrupt vector for mux
                                 ;
 005766  016105  000000G                 MOV     MXVEC(R1),R5     ;Get address of input interrupt vector
 005772  012702  000000G                 MOV     #INMXV,R2        ;End of receiving vector
 005776  012703  177772G                 MOV     #<OTMXV-6>,R3    ;Output interrupt table
 006002  010100                          MOV     R1,R0            ;Get mux index number
 006004  006300                          ASL     R0               ;4 bytes per line in input int table
 006006  160002                          SUB     R0,R2            ;Get address of input entry point
 006010  060100                          ADD     R1,R0            ;6 bytes per mux in output entry point table
 006012  060003                          ADD     R0,R3            ;Get address of output int entry point
 006014  010225                          MOV     R2,(R5)+         ;Set PC for input interrupt
 006016  012725  000340                  MOV     #340,(R5)+       ;Set PS for output interrupt
 006022  010325                          MOV     R3,(R5)+         ;Set PC for output interrupt
 006024  012715  000340                  MOV     #340,(R5)        ;Set PS for output interrupt
                                 ;
                                 ;  Now store an instruction sequence of the form:
                                 ;
                                 ;       JSR     R5,@#interrupt_routine
                                 ;       .WORD   mux_index
                                 ;
                                 ;  to catch mux output interrupts and vector them to the interrupt routine.
                                 ;
 006030  012723  004537                  MOV     #004537,(R3)+    ;JSR R5,@#x
 006034  012700  000000G                 MOV     #DZOINT,R0       ;Assume this is a DZ11
 006040  026127  000000G 000000G         CMP     MXTYPE(R1),#CDX$DZ ;Is this a DZ11?
 006046  001410                          BEQ     1$               ;Br if yes
 006050  012700  000000G                 MOV     #VHOINT,R0       ;Assume this is a DHV11
 006054  026127  000000G 000000G         CMP     MXTYPE(R1),#CDX$VH ;Is this a DHV11?
 006062  001402                          BEQ     1$               ;Br if yes
 006064  012700  000000G                 MOV     #DHOINT,R0       ;Get interrupt routine for DH11's
 006070  010023                  1$:     MOV     R0,(R3)+         ;Store address of interrupt routine
 006072  010113                          MOV     R1,(R3)          ;Store mux index number
                                 ;
                                 ;  Finished
                                 ;
 006074  012605                  9$:     MOV     (SP)+,R5
 006076  012603                          MOV     (SP)+,R3
 006100  012602                          MOV     (SP)+,R2
 006102  000207                          RETURN
```

```
                                        .SBTTL  DHINIT -- Initialize a DH11 multiplexer
                                ;----------------------------------------------------------------------------
                                ;  DHINIT is called to initialize a DH11 multiplexer
                                ;
                                ;  Inputs:
                                ;   R1 = Mux index number
                                ;
006104                          DHINIT:
                                ;
                                ;  See if this DH11 is installed
                                ;
006104  005761  000000G                 TST     MH$SCR(R1)      ;Is this DH11 installed?
006110  001420                          BEQ     9$              ;Br if not
                                ;
                                ;  Connect interrupt vector to DH11
                                ;
006112  004737  005760'                 CALL    MUXVEC          ;Set up interrupt vectors for DH11
                                ;
                                ;  Clear the multiplexer
                                ;
006116  012771  000000G 000000G         MOV     #HF$MC,@MH$SCR(R1) ;Set the master-clear flag
006124  032771  000000G 000000G 1$:     BIT     #HF$MC,@MH$SCR(R1) ;Wait for the master clear to be completed
006132  001374                          BNE     1$
                                ;
                                ;  Clear the DM11 scanner
                                ;
006134  016100  000000G                 MOV     DM$CSR(R1),R0   ;Is there an associated DM11?
006140  001404                          BEQ     3$              ;Br if not
006142  012710  000000G                 MOV     #MF$CS,(R0)     ;Clear the scanner
006146  052710  000000G                 BIS     #MF$CM,(R0)     ;Clear the multiplexer
006152                          3$:
                                ;
                                ;  Finished
                                ;
006152                          9$:
006152  000207                          RETURN
```

```
                                        .SBTTL  VHINIT -- Initialize a DHV11 multiplexer
                                ;-----------------------------------------------------------------------------
                                ;  Perform initialization for a DHV11 mux.
                                ;
                                ;  Inputs:
                                ;   R1 = Mux index number.
                                ;
006154                          VHINIT:
                                ;
                                ;  See if this DHV11 is installed
                                ;
006154  005761  000000G                 TST     VH$CSR(R1)      ;Is this DHV11 installed?
006160  001414                          BEQ     9$              ;Br if not
                                ;
                                ;  Connect interrupt vector to DHV11
                                ;
006162  004737  005760'                 CALL    MUXVEC          ;Set up interrupt vectors
                                ;
                                ;  Clear the multiplexer
                                ;
006166  012771  000000G 000000G         MOV     #VF$MR,@VH$CSR(R1) ;Reset the multiplexer
006174  032771  000000G 000000G 1$:     BIT     #VF$MR,@VH$CSR(R1) ;Wait for reset to finish
006202  001374                          BNE     1$
                                ;
                                ;  Clean out the FIFO buffer in the mux
                                ;
006204  017100  000000G         2$:     MOV     @MXRBUF(R1),R0  ;Get contents of receiver buffer register
006210  002775                          BLT     2$              ;Loop until RBUF empty
                                ;
                                ;  Finished
                                ;
006212  000207                  9$:     RETURN
```

```
;------------------------------------------------------------------------
;  End of code that can be omitted for Pro-only systems
;
        .IFF    ;NE,<PROASM-1>  ;Begin code for Pro only
;
;  This code is assembled only for Pro systems.
;  T/S line init routines for Pro only.
;
LINCHK:
DHLPRM:
VHLPRM:
DZINIT:
MUXVEC:
DHINIT:
VHINIT:
        RETURN
;
;  LININI routine for Pro systems
;
LININI: MOV     R1,-(SP)
        MOV     #LSTLIN,R1      ;Get # of last line
;
;  Determine if this line is connected to hardware
;
1$:     CALL    LINTYP          ;Determine the type of this line
        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
        BEQ     2$              ;Br if not
;
;  Call Pro line initialization routine
;
        CALL    PROLIN          ;Initialize Pro line
;
;  Do some special init for phone lines
;
        BIT     #$PHONE,ILSW2(R1);Is this a dialup line?
        BEQ     2$              ;Br if not
        BIS     #$NOIN,LSW3(R1) ;Ignore input till dial up occurs
;
;  Check next line
;
2$:     SUB     #2,R1           ;Get index # of next line
        BGT     1$              ;Loop if more lines to do
;
;  Finished
;
        MOV     (SP)+,R1
        RETURN
;
;  End of Pro-only code
;
        .ENDC   ;NE,<PROASM-1>
```

```
                                          .SBTTL  LINTYP -- Determine the type of a line
                                  ;---------------------------------------------------------------------------
                                  ;  LINTYP is called to determine if the current line is a time-sharing line
                                  ;  a CL line, or a non-hardware connected line.
                                  ;
                                  ;  Inputs:
                                  ;   R1 = Line index number
                                  ;
006214  020127  000000G           LINTYP: CMP     R1,#LSTPL       ;Is this a time-sharing line?
006220  101422                            BLOS    1$              ;Br if yes
006222  020127  000000G                   CMP     R1,#FSTIOL      ;Is this a CL line?
006226  103004                            BHIS    2$              ;Br if yes
006230  012761  177777   000000G          MOV     #-1,LCLUNT(R1)  ;Say line not in use as a CL line
006236  000432                            BR      9$
                                  ;
                                  ;  This is a CL line
                                  ;
006240  016100  000000G           2$:     MOV     LCLUNT(R1),R0   ;Get the CL unit index number
006244  010160  000000G                   MOV     R1,CL$LIX(R0)   ;Say which line is assoc with this CL unit
006250  012761  000000G  000000G          MOV     #CLOTIR,LOUTIR(R1) ;Set terminal output interrupt routine
006256  012761  000000G  000000G          MOV     #CLINCP,LINIR(R1)  ;Set terminal input interrupt routine
006264  000414                            BR      8$
                                  ;
                                  ;  This is a time-sharing line
                                  ;
006266  012761  177777   000000G  1$:     MOV     #-1,LCLUNT(R1)  ;Say line is not in use as a CL unit
006274  012761  177777   000000G          MOV     #-1,LXCL(R1)    ;Line is not cross-connected to CL unit
006302  012761  000000G  000000G          MOV     #NEDCHR,LOUTIR(R1) ;Set terminal output interrupt routine
006310  012761  000000G  000000G          MOV     #TTINCP,LINIR(R1)  ;Set terminal input interrupt routine
006316  052761  000000G  000000G  8$:     BIS     #$HARD,LSW3(R1) ;This line is connected to hardware
                                  ;
                                  ;  Finished
                                  ;
006324  000207                    9$:     RETURN
```

```
                                        .SBTTL  * * * Initialization done with RT-11 running * * *
                                ;----------------------------------------------------------------------------
                                ;  Initialization at start of execution of TSX.
                                ;
                                ;  The initialization done in this section uses the running RT-11 system
                                ;  to perform functions for it.
                                ;
006326                          INITGO:
                                ;
                                ;  Save some RT-11 pointers in case we abort the initialization
                                ;
006326  013737  000004  000044'         MOV     @#4,RTTRP4      ;Save trap 4 vector
006334  013737  000000G 000046'         MOV     @#RMON,RTMNVC   ;RT-11 monitor pointer
006342  013737  000000G 000042'         MOV     @#CLKVEC,CLK100 ;Clock vector (defined in TSGEN at 100)
                                ;
                                ;  Get the current time of day which we will use later to make sure
                                ;  the line time clock is working.
                                ;
006350                                  .GTIM   #AREA,#SYTIMH   ;Get the current time of day
                                ;
                                ;  Trap ^C for later test so we can restore clock vector
                                ;
006370                                  .SCCA   #AREA,#CCAFLG   ;Catch control-C
                                ;
                                ;  Check for TSGEN size overflow
                                ;
006410  012700  000000G                 MOV     #GENTOP,R0      ;Get top of TSGEN
006414  162700  037776                  SUB     #<40000-2>,R0   ;Will TSKMON have problems?
006420  003422                          BLE     15$             ;Continue if not
006422  010046                          MOV     R0,-(SP)        ;Save overflow size
006424                                  .PRINT  #TSXHD          ;Print error message
006432                                  .PRINT  #TOOBIG
006440                                  .PRINT  #REDUCE
006446  012600                          MOV     (SP)+,R0        ;Recover amount of overflow
006450  004737  027754'                 CALL    PRTDEC
006454                                  .PRINT  #BYTES
006462  000137  004102'                 JMP     INISTP
                                ;
                                ;  Initialize the system stack (below 1000)
                                ;
006466  012701  000000G         15$:    MOV     #SSEND,R1       ;Point to bottom of stack
006472  012700  123456                  MOV     #123456,R0      ;Get initialization value
006476  010021                  13$:    MOV     R0,(R1)+        ;Initialize the stack
006500  020127  000000G                 CMP     R1,#SS          ;Reached top of the stack area?
006504  103774                          BLO     13$             ;Loop if not
006506  010106                          MOV     R1,SP           ;Run on system stack
                                ;
                                ;  Make sure we are not already running under TSX.
                                ;
006510                                  .SERR                   ;DON'T DIE ON ERRORS
006516  012700  000176'                 MOV     #GTLIN,R0       ;TSX EMT TO GET LINE NUMBER
006522  104375                          EMT     375             ;TRY A TSX EMT
006524  103410                          BCS     1$              ;BR IF NOT UNDER TSX
006526                                  .PRINT  #TSXHD          ;ALREADY UNDER TSX
006534                                  .PRINT  #TSXRUN
006542  000137  004102'                 JMP     INISTP
006546                          1$:     .HERR                   ;RENABLE FATAL ERRORS
```

```
                                       ;
                                       ;  Make sure this machine has memory management facilities.
                                       ;
006554                                         .TRPSET #AREA,#NOXM      ;CATCH TRAPS
006574  005737  000000G                        TST     @#SROMMR         ;TRY TO ACCESS MEMORY MANAGEMENT REGISTER
006600                                         .TRPSET #AREA,#0         ;Release trap control
                                       ;
                                       ;  Request all available memory from RT-11.
                                       ;
006616                                         .SETTOP #-2              ;REQUEST ALL AVAILABLE MEMORY
006624  010037  000132'                        MOV     R0,TOPMEM        ;REMEMBER WHERE TOP OF MEMORY IS
006630  020027  000000G                        CMP     R0,#VPAR5        ;TSX CANNOT EXTEND ABOVE PAR5 BASE ADDRESS
006634  101402                                 BLOS    3$               ;BR IF RT-11 IS BELOW THAT
006636  012700  000000G                        MOV     #VPAR5,R0        ;SET PAR5 BASE AS UPPER LIMIT ON TSX SIZE
006642  010037  000236'                3$:     MOV     R0,MEMLIM        ;TSX MAY NOT EXCEED THIS UPPER LIMIT
                                       ;
                                       ;  Lock USR in memory for speed
                                       ;  (Set USR to swap over TSEMT to get out of the way)
                                       ;
006646  012705  177776'                        MOV     #TSINIT-2,R5     ;GET THE BASE OF TSINIT
006652                                         .GVAL   #AREA,#374       ;GET SIZE OF RT-11 USR MODULE
006672  160005                                 SUB     R0,R5            ;ALLOCATE SPACE BELOW TSINIT FOR USR
006674  010537  000046                         MOV     R5,@#46          ;SET USR TO SWAP OVER TSEMT
006700                                 5$:     .LOCK                    ;LOCK USR IN MEMORY
                                       ;
                                       ;  Determine if we are to run system with the system debugger
                                       ;
006702  032737  000000G 000000G                BIT     #CHAIN,@#JSWLOC  ;WERE WE CHAINED TO?
006710  001406                                 BEQ     10$              ;BR IF NOT
006712  023727  000510  057164                 CMP     @#510,#R50ODT    ;SHOULD WE RUN UNDER ODT?
006720  001002                                 BNE     10$              ;BR IF NOT
006722  005237  000034'                        INC     ODTFLG           ;SET FLAG SAYING DEBUGGER WANTED
                                       ;
                                       ;  Call Pro TSX initialization only if assembling for the Pro
                                       ;  Jump to INISTP if checking fails.
                                       ;
006726                                 10$:
                                       .IF     NE,PROCID                ;** Do if assembling for pro only **
                                               CALL    INSCHK           ;PERFORM VERIFICATION AND DECRYPTION FOR PRO
                                       .ENDC   ;NE,PROCID
                                       ;
                                       ;  Allocate non-initialized buffer space over TSINIT.
                                       ;
006726  012705  000000'                        MOV     #TSINIT,R5       ;Allocate buffer space over TSINIT
006732  004737  013060'                        CALL    ALOCBF           ;Do allocation
006736  004737  013524'                        CALL    ALCSLO           ;Allocate silo buffers for lines
006742  020527  006326'                        CMP     R5,#INITGO       ;Are we beyond code that takes over control?
006746  103002                                 BHIS    12$              ;Br if yes
006750  012705  006326'                        MOV     #INITGO,R5       ;Advance up to initial code
                                       ;
                                       ;  Allocate the interrupt stack over TSINIT
                                       ;  If we are running on a Pro, allocate buffer for the PI handler
                                       ;  initialization code over the interrupt stack area.
                                       ;
        001274                         PIINSZ  =       700.             ;Space needed for PI init code
006754  010537  000000G                12$:    MOV     R5,INTSND        ;Ptr to base of interrupt stack
006760  062705  000002                         ADD     #2,R5            ;Always leave last word of stack for flag val
```

```
006764  013701  000000G                         MOV     @#RMON,R1       ;Get pointer to RT-11 RMON base
006770  032761  000000G 000370                  BIT     #CW$PRO,370(R1) ;Are we running on a PRO?
006776  001407                                  BEQ     11$             ;Br if not
007000  105237  000000G                         INCB    PROFLG          ;Set flag saying this is a PRO-350
007004  010537  000150'                         MOV     R5,PROBUF       ;Save pointer to buffer area
007010  062705  001274                          ADD     #PIINSZ,R5      ;Allocate space for buffer
007014  000402                                  BR      14$
007016  062705  000000G                 11$:    ADD     #INTSSZ,R5      ;Allocate space for interrupt stack
007022  010537  000000G                 14$:    MOV     R5,INTSTK       ;Address of top of interrupt stack
                                        ;
                                        ;  Allocate space for those overlays that go over TSINIT
                                        ;
007026  004737  023372'                         CALL    OVLPOS          ;Determine how much space to alloc for overlay
                                        ;
                                        ;  Note: from this point onward we are carrying the address of the
                                        ;  base of the free memory area in R5.
                                        ;
007032  020527  030074'                         CMP     R5,#INITOP      ;Have we allocated up to top of TSINIT?
007036  103002                                  BHIS    4$              ;Br if yes
007040  012705  030074'                         MOV     #INITOP,R5      ;Advance to top of TSINIT
                                        ;
                                        ;  Allocate a 2048. byte work buffer
                                        ;
007044  004737  010340'                 4$:     CALL    ALCWRK          ;Allocate work buffer
                                        ;
                                        ;  Allocate empty Region Control Blocks for use by handlers
                                        ;
007050  004737  010374'                         CALL    ALCHRB
                                        ;
                                        ;  If we were started in debug mode, load ODT.
                                        ;
                                                .IF     EQ,PROCID       ;Don't allow ODT for production PRO version
007054  005737  000034'                         TST     ODTFLG          ;Are we to load system debugger?
007060  001402                                  BEQ     2$              ;Br if not
007062  004737  026314'                         CALL    GETODT          ;Load ODT and start it
                                                .ENDC   ;EQ,PROCID
                                        ;
                                        ;  Initialize memory management registers for 1-to-1 mapping but
                                        ;  leave memory management turned off
                                        ;
007066  004737  016436'                 2$:     CALL    MEMINI          ;Initialize memory management
                                        ;
                                        ;  Extract information from RT-11 configuration and sysgen words.
                                        ;
007072  013701  000000G                         MOV     @#RMON,R1       ;GET POINTER TO BASE OF RMON
007076  016102  000300                          MOV     300(R1),R2      ;GET RT-11 CONFIGURATION WORD
007102  042702  000000C                         BIC     #CW$GDH+CW$BTH+CW$LGS,R2 ;RESET A FEW FLAGS
007106  052702  000000C                         BIS     #CW$FB+CW$FGJ+CW$USR+CW$XM,R2 ;SET A FEW FLAGS
007112  010237  000000G                         MOV     R2,CONFIG       ;INITIALIZE OUR CONFIGURATION WORD
                                        ;  Now get extended configutation word.
007116  016137  000370  000000G                 MOV     370(R1),CONFG2  ;EXTENDED CONFIGURATION WORD
007124  052737  000000G 000000G                 BIS     #CW$ESP,CONFG2  ;SET EXIT NO SWAP FLAG
007132  123727  000000G 000000G                 CMPB    VBUSTP,#QBUS    ;Is this a Q-bus machine?
007140  001003                                  BNE     25$             ;Br if not
007142  052737  000000G 000000G                 BIS     #CW$QBS,CONFG2  ;Set QBUS flag
                                        ;  And sysgen option word.
007150  016102  000372                  25$:    MOV     372(R1),R2
```

```
007154  042702  000000C                 BIC     #SG$ELG+SG$PAR+SG$MTS,R2
007160  052702  000000C                 BIS     #SG$MMU+SG$MTM+SG$IOT+SG$TSX,R2
007164  010237  000000G                 MOV     R2,SYSGEN       ;INITIALIZE OUR SYSGEN WORD
                                ;
                                ;   If a system version number was specified, use it.
                                ;   Else, get version number from RT-11, but limit to default.
                                ;
007170  013700  000000G                 MOV     SYSVER,R0       ;Has user specified version to emulate?
007174  001015                          BNE     30$             ;If so, keep SYSVER
007176  012737  000314' 000156'         MOV     #RTVDEF,RTVPTR  ;If using RT version, set cutoff
007204                                  .GVAL   #AREA,#276      ;GET RT-11 SYSTEM VERSION NUMBER
007224  010037  000000G                 MOV     R0,SYSVER       ;SET AS TSX-PLUS VERSION NUMBER
                                ;
                                ;   Now scan the known version number table and try to locate a match.
                                ;   R0 contains version # in low byte, udpate # in high byte
                                ;
007230  012702  000264'         30$:    MOV     #RTVER,R2       ;Get ptr to first entry in table
007234  120062  000000          31$:    CMPB    R0,RT$VER(R2)   ;Does main version match?
007240  001005                          BNE     32$             ;Br if not
007242  000300                          SWAB    R0              ;Main version matches, get update to low byte
007244  120062  000001                  CMPB    R0,RT$UPD(R2)   ;Does the update match also?
007250  001425                          BEQ     34$             ;Its a match! Use this entry
007252  000300                          SWAB    R0              ;Get SYSVER back to low byte
007254  062702  000003          32$:    ADD     #RTV$SZ,R2      ;If not, step up to the next entry
007260  020227  000322'                 CMP     R2,#RTVEND      ;Past end of table?
007264  103763                          BLO     31$             ;Keep checking if not
                                ;
                                ;   Couldn't find version in tables.  If we picked it up from RT, reset
                                ;   everything to the default.  If it was user-specified, then keep
                                ;   SYSVER, but use last entry ptr.
                                ;
007266  023727  000156' 000322'         CMP     RTVPTR,#RTVEND  ;Was a limit specified? (Got ver. from RT?)
007274  103404                          BLO     33$             ;Br if so
                                ;   Unknown user-specified version, keep user-specified SYSVER,
                                ;   but use ptr to last known version
007276  012737  000317' 000156'         MOV     #<RTVEND-RTV$SZ>,RTVPTR ;Use latest known defaults
007304  000414                          BR      36$
                                ;   Got version from RT, but don't recognize it, reset SYSVER and use defaults
                                ;   RTVPTR was already set to default RTVDEF when we got RT version
007306  113737  000314' 000000G 33$:    MOVB    RTVDEF+RT$VER,SYSVER ;Set SYSVER to default
007314  113737  000315' 000000G         MOVB    RTVDEF+RT$UPD,SYSUPD ;and update
007322  000405                          BR      36$
                                ;
                                ;   Version was identified in tables.  RTVPTR contains limiting version ptr
                                ;   (RTVDEF if got vers from RT, RTVEND if user specified version)
                                ;
007324  020237  000156'         34$:    CMP     R2,RTVPTR       ;Is it past limit?
007330  103366                          BHIS    33$             ;If so, keep limit, and go limit SYSVER
007332  010237  000156'                 MOV     R2,RTVPTR       ;If not, use what we found
                                ;
                                ;   Now set some information that depends on the emulated version number
                                ;
007336  105737  000000G         36$:    TSTB    CLVERS          ;Use table value for CL version number?
007342  001005                          BNE     38$             ;Br if not, use value supplied in TSGEN
007344  013702  000156'                 MOV     RTVPTR,R2       ;Get ptr to version table
007350  116237  000002  000000G         MOVB    CL$VER(R2),CLVERS ;Auto set CLVERS from version table
007356  105737  000000G         38$:    TSTB    LDVERS          ;Auto-select LD translation table type?
```

```
    007362  001011                                  BNE     39$             ;Br if not, use value supplied in TSGEN
    007364  112737  000001  000000G                 MOVB    #1,LDVERS       ;Assume old translation table format
    007372  023727  000156' 000314'                 CMP     RTVPTR,#RT54    ;At or beyond 5.4?
    007400  103402                                  BLO     39$             ;Br if not, retain old format
    007402  105237  000000G                         INCB    LDVERS          ;LD translation table format changed at 5.4
    007406                                  39$:
```

```
                               ;
                               ;   Set up a few clock constants based on clock frequency.
                               ;   See if we have a 50 or 60 Hz clock
                               ;
007406 032737  000000G 000000G INICLK: BIT     #CW$50H,CONFIG  ;50 or 60 Hz clock?
007414 001017                          BNE     2$              ;Br if 50 Hz
                               ;
                               ;   60 Hz clock
                               ;
007416 012737  000074  000000G         MOV     #60.,TK1SEC     ;Clock ticks per 1 second
007424 012737  000036  000000G         MOV     #30.,TK5VAL     ;Clock ticks per 0.5 seconds
007432 012737  000264  000000G         MOV     #180.,TK3SVL    ;Clock ticks per 3 seconds
007440 012737  000006  000000G         MOV     #6.,TK1VAL      ;Clock ticks per 0.1 seconds
007446 012700  001130                  MOV     #600.,R0        ;Get # clock ticks per 10 seconds
007452 000416                          BR      8$
                               ;
                               ;   50 Hz clock
                               ;
007454 012737  000062  000000G 2$:     MOV     #50.,TK1SEC     ;Clock ticks per 1 second
007462 012737  000031  000000G         MOV     #25.,TK5VAL     ;Clock ticks per 0.5 seconds
007470 012737  000226  000000G         MOV     #150.,TK3SVL    ;Clock ticks per 3 seconds
007476 012737  000005  000000G         MOV     #5.,TK1VAL      ;Clock ticks per 0.1 seconds
007504 012700  000764                  MOV     #500.,R0        ;Get # clock ticks per 10 seconds
                               ;
                               ;   Set number of clock ticks per day
                               ;
007510 012702  020700          8$:     MOV     #8640.,R2       ;(# seconds per day) / 10.
007514 070200                          MUL     R0,R2           ;Get # clock ticks per day
007516 010237  000000G                 MOV     R2,DATIMH       ;High-order value
007522 010337  000000G                 MOV     R3,DATIML       ;Low-order value
                               ;
                               ;   Do a fast check to make sure specified T/S line addresses are ok.
                               ;
007526 004737  010450'         CKLIN:  CALL    LINCHK          ;CHECK T/S LINE ADDRESSES
                               ;
                               ;   Do PRO-350 system initialization
                               ;
                               .IF     NE,PROASM
                                       BIT     #CW$PRO,CONFG2  ;Are we running on a PRO-350?
                                       BEQ     INIDEV          ;Br if not
                                       CALL    PROINI          ;Do PRO-350 initialization
                                       MOV     #PIDRIV,PIDPTR  ;Set up pointer to clock driven PI routine
                               .ENDC   ;NE,PROASM
                               ;
                               ;   Make entry in device handler table for TT device.
                               ;
007532 012737  100040  000000G INIDEV: MOV     #R50TT,PNAME    ;PERMANENT NAME "TT"
007540 012737  000000G 000000G         MOV     #DI$TT,DVSTAT   ;SET DEVICE STATUS FLAGS FOR TT
007546 005037  000000G                 CLR     DVFLAG
007552 005037  000000G                 CLR     DEVSIZ
007556 012737  000002  000000G         MOV     #2,HANENT       ;SET UP HANENT SO HANDLER LOOKS RESIDENT
007564 005037  000000G                 CLR     NUMDEV          ;IT IS DEVICE # 0
                               ;
                               ;   Make device table entry for LD (logical disk) device
                               ;
007570 012737  177777  000000G         MOV     #-1,LDDEVX      ;ASSUME LD SUPPORT NOT WANTED
007576 105737  000000G                 TSTB    VLDSYS          ;IS LD SUPPORT GENNED IN?
```

```
007602  001425                                  BEQ     6$              ;BR IF NOT
007604  062737  000002  000000G                 ADD     #2,NUMDEV       ;ONE MORE DEVICE
007612  013701  000000G                         MOV     NUMDEV,R1       ;GET DEVICE TABLE INDEX
007616  010137  000000G                         MOV     R1,LDDEVX       ;REMEMBER INDEX NUMBER FOR LD DEVICE
007622  012761  045640  000000G                 MOV     #R50LD,PNAME(R1) ;SET DEVICE NAME ("LD")
007630  012761  000000C 000000G                 MOV     #<DS$DIR+DS$SFN+DS$VSZ+DI$LD>,DVSTAT(R1);SET DEV STATUS FLAGS
007636  012761  000000G 000000G                 MOV     #DX$EBA,DVFLAG(R1);Say buffers must be on even byte boundaries
007644  005061  000000G                         CLR     DEVSIZ(R1)
007650  012761  000002  000000G                 MOV     #2,HANENT(R1)   ;SAY HANDLER IS RESIDENT
                                        ;
                                        ;  Make device table entry for CL (communications line) device
                                        ;
007656  005727  000000G                 6$:     TST     #CLTOTL         ;Are there any communications lines?
007662  001402                                  BEQ     8$              ;Br if not
007664  004737  014652'                         CALL    CLINIT          ;Initialize CL handler
                                        ;
                                        ;  Disable clock interrupts.
                                        ;
007670  012737  000002  000000          8$:     MOV     #2,@#0          ;LOAD RTI IN LOCATION 0
007676  005037  000000G                         CLR     @#CLKVEC        ;ATTACH CLOCK INTERRUPT TO 0
007702  032737  000000G 000000G                 BIT     #CW$PRO,CONFG2  ;ARE WE RUNNING ON A PRO?
007710  001402                                  BEQ     1$              ;BR IF NOT
007712  005037  000230                          CLR     @#230           ;380 CLOCK INTERRUPT VECTOR
                                        ;
                                        ;  Set up memory parity control
                                        ;
007716  004737  017642'                 1$:     CALL    PARSET          ;SET UP MEMORY PARITY CONTROL
                                        ;
                                        ;  Determine how much memory is installed on machine
                                        ;
007722  004737  016542'                         CALL    MEMTST          ;FIND OUT HOW MUCH PHYSICAL MEMORY THERE IS
                                        ;
                                        ;  Set up information about the size of the job context area
                                        ;
007726  004737  017172'                         CALL    CXTALC          ;Determine size of job context area
                                        ;
                                        ;  Load TSX-Plus device handlers that go in low memory
                                        ;
007732  004737  017730'                         CALL    GETHNL          ;Load low memory handlers
                                        ;
                                        ;  Reserve space for interrupt vector intercept routines for mapped handlers
                                        ;
007736  010537  000122'                         MOV     R5,XMVBAS       ;Save address of base of area for XM vectors
007742  013701  000124'                         MOV     NMXHAN,R1       ;Get # mapped handlers
007746  006301                                  ASL     R1              ;Reserve room for 2 interrupts per handler
007750  062701  000000G                         ADD     #NXIVMH,R1      ;Add # requested extra interrupt vectors
007754  070127  000032                          MUL     #MPIVSZ,R1      ;Calc space needed for interrupt entry code
007760  060105                                  ADD     R1,R5           ;Advance the address of free memory
                                        ;
                                        ;  Set up device index number and unit number for "SY:" device.
                                        ;
007762  004737  027324'                         CALL    SETSY           ;SET UP INFO ABOUT SY DEVICE
                                        ;
                                        ;  Open channel to TSKMON and set up information about it.
                                        ;
007766  004737  014350'                         CALL    OPNKMN          ;OPEN CHANNEL TO TSKMON
                                        ;
                                        ;  Set up information about the IND program
                                        ;
007772  004737  015316'                         CALL    INDINI          ;INITIALIZE FOR IND PROGRAM
                                        ;
                                        ;  Initialize the TSXUCL data file
                                        ;
007776  004737  016060'                         CALL    UCLINI
                                        ;
                                        ;  Set name of device that UCL program is to be run from
                                        ;
010002  013737  000000G 000000G                 MOV     SYNAME,UCLNAM   ;SET DEVICE NAME FOR UCL PROGRAM
                                        ;
                                        ;  Initialize spooling system
                                        ;
010010  004737  011730'                         CALL    SPLINI          ;INITIALIZE SPOOLING SYSTEM
                                        ;
                                        ;  Open system swap file
                                        ;
010014  105737  000000G                         TSTB    VSWPFL          ;IS JOB SWAPPING ALLOWED?
010020  001402                                  BEQ     3$              ;BR IF NOT
010022  004737  010772'                         CALL    OPNSWP          ;OPEN THE SYSTEM SWAP FILE
                                        ;
                                        ;  Open swap file used for PLAS regions
                                        ;
010026  004737  011424'                 3$:     CALL    OPNRSF          ;Open PLAS region swap file
                                        ;
                                        ;  Set up information about which devices need to have their I/O mapped
                                        ;
010032  004737  023150'                         CALL    SETMIO          ;Set up information about mapped I/O
                                        ;
                                        ;  We are finished allocating low-memory buffer space.
                                        ;
010036  010500                                  MOV     R5,R0           ;ENSURE WE DON'T OVERFLOW 40KB
010040  004737  027630'                         CALL    CHKMEM          ;ABORT IF > 40KB OR INTO RT-11
                                        ;
                                        ;  From this point on carry the free memory address in R5
                                        ;  as a 64-byte block # in physical memory.
                                        ;
010044  010537  000000G                         MOV     R5,UMSYTP       ;SAVE ADDRESS OF NON-EXTENDED SYSTEM TOP
010050  062705  000077                          ADD     #77,R5          ;BOUND UP TO 64-BYTE BOUNDARY
010054  072527  177772                          ASH     #-6,R5          ;CONVERT TO 64-BYTE BLOCK #
010060  042705  176000                          BIC     #176000,R5      ;KILL SIGN EXTENSION
                                        ;
                                        ;  Allocate buffer space that is not contrained to 40Kb TSX-Plus region.
                                        ;
010064  004737  013744'                         CALL    ALBFX           ;ALLOCATE EXTENDED BUFFERS
                                        ;
                                        ;  We will now do some allocation from the top of physical memory downward.
                                        ;  Save the base of free memory in R4 and get the top of free memory to R5.
                                        ;
010070  010504                                  MOV     R5,R4           ;Save the base of free memory in R4
010072  010437  000136'                         MOV     R4,FMEMLO       ;Save pointer above top of alloc low memory
010076  013705  000134'                         MOV     FMEMHI,R5       ;Get 64-byte block # of free high memory
                                        ;
                                        ;  Load any mapped system code
                                        ;
010102  004737  024002'                         CALL    GETMAP          ;LOAD USR, EMT, MSG, LOCK, SPOOL, etc.
```

```
                                       ;
                                       ;  Load any shared run-time systems
                                       ;
 010106  005727  000000G                       TST     #NUMRDB         ;Do we need to load any shared run-times?
 010112  001415                                BEQ     4$              ;Br if not
 010114  012701  000000G                       MOV     #RDB,R1         ;Point to 1st run-time descriptor block
 010120  010502                                MOV     R5,R2           ;Save initial memory pointer
 010122  004737  025464'               5$:     CALL    GETSRT          ;Load a shared run-time system
 010126  062701  000000G                       ADD     #RT$$SZ,R1      ;Point to next shared run-time descriptor
 010132  020127  000000G                       CMP     R1,#RDBEND      ;Are there more to load?
 010136  103771                                BLO     5$              ;Br if yes
 010140  160502                                SUB     R5,R2           ;Compute amt of space used by run-times
 010142  010237  000000G                       MOV     R2,SRTSIZ       ;Save total run-time size
 010146                                4$:
                                       .IF     NE,PROASM
                                       ;
                                       ;  If we are running on a Pro, load the PI handler like a shared run-time
                                       ;
                                               TSTB    PROFLG          ;Are we running on a Pro?
                                               BEQ     10$             ;Br if not
                                               MOV     #PISRT,R1       ;Point to dummy shared run-time block for PI
                                               MOV     R5,R2           ;Save current memory pointer
                                               CALL    GETSRT          ;Load PI handler like a shared run-time
                                               SUB     R5,R2           ;Calculate amt of space used by PI handler
                                               ADD     R2,MHNSIZ       ;Count in mapped-handler size
                                       .ENDC   ;NE,PROASM
                                       ;
                                       ;  Load any mapped handlers
                                       ;
 010146  004737  021244'               10$:    CALL    GETHNH          ;Load mapped handlers
                                       ;
                                       ;  Allocate space for data cache buffers and control tables
                                       ;
 010152  004737  026076'                       CALL    CSHBUF          ;Allocate space for data cache
                                       ;
                                       ;  We have finished allocating all of the memory used by the system.
                                       ;  Allocate and initialize a memory map table that will be used to
                                       ;  show which pages are available for user jobs.
                                       ;
 010156  004737  017332'                       CALL    MAPALC          ;Allocate memory map table
                                       ;
                                       ;  Set up info about maximum memory space available to jobs
                                       ;
 010162  004737  017516'                       CALL    SETJSZ          ;SET JOB SIZE INFO
                                       ;
                                       ;  Set up date and time
                                       ;
 010166  013702  000000G                       MOV     SYTIML,R2       ;Save time we got at start of init
 010172                                        .GTIM   #AREA,#SYTIMH   ;SET TIME OF DAY
 010212                                        .DATE                   ;GET DATE
 010220  010037  000000G                       MOV     R0,SYSDAT       ;SET SYSTEM DATE
 010224  020237  000000G                       CMP     R2,SYTIML       ;Make sure some time has elapsed
 010230  001010                                BNE     11$             ;Br if clock is running
 010232                                        .PRINT  #TSXHD          ;Print error message heading
 010240                                        .PRINT  #NOCLOK         ;Print clock-not-working message
 010246  000137  004102'                       JMP     INISTP          ;Abort initialization
                                       ;
```

```
                                        ;  Unlock the USR so that TSEMT will be swapped back in.
                                        ;
010252                                  11$:    .UNLOCK                 ;RELEASE USR
                                        ;
                                        ;  Read back into memory that part of the resident portion of TSX
                                        ;  that we overlayed with our work buffer.
                                        ;
010254  013702  000154'                         MOV     WRKSIZ,R2       ;Get size of work buffer
010260  006202                                  ASR     R2              ;Convert to # words
010262  013703  000152'                         MOV     WRKBUF,R3       ;Get address of work buffer area
010266  000241                                  CLC                     ;Convert to block # in TSX.SAV file
010270  006003                                  ROR     R3
010272  000303                                  SWAB    R3
010274                                          .READW  #AREA,#17,WRKBUF,R2,R3 ;Read back TSX over work buffer
                                        ;
                                        ;  See if user requested control-C abort
                                        ;
010330  004737  027670'                         CALL    CCATST          ;JUMP TO INISTP IF ^C^C BEFORE THIS POINT
                                        ;
                                        ;  Jump to code at end of TSINIT which takes over control from RT-11
                                        ;
010334  000137  003420'                         JMP     TAKOVR
```

```
                                          .SBTTL  * * *  Subroutines  * * *
                                          .SBTTL  ALCWRK -- Allocate a work buffer
                                  ;----------------------------------------------------------------------
                                  ;  Allocate a 2048. byte work buffer over a resident portion of TSX.
                                  ;  This area will be restored from the TSX.SAV disk file after we
                                  ;  are finished using the work area.
                                  ;
                                  ;  Outputs:
                                  ;   WRKBUF = Address of base of work buffer.
                                  ;   WRKSIZ = Size of work buffer (2048).
                                  ;
010340  010246                    ALCWRK: MOV     R2,-(SP)
                                  ;
                                  ;  Get address of start of area where buffer can go and then bound
                                  ;  up to a block boundary.
                                  ;
010342  012702  000000G                   MOV     #EXCBUF,R2      ;Get address of base of buffer area
010346  062702  000777                    ADD     #777,R2         ;Bound up to block boundary
010352  042702  000777                    BIC     #777,R2         ;Set to block boundary
010356  010237  000152'                   MOV     R2,WRKBUF
010362  012737  004000  000154'           MOV     #2048.,WRKSIZ
                                  ;
                                  ;  Finished
                                  ;
010370  012602                            MOV     (SP)+,R2
010372  000207                            RETURN
```

```
                                        .SBTTL  ALCHRB -- Allocate Region Control Blocks for handlers
                                ;-------------------------------------------------------------------------
                                ;  This routine allocates and initializes empty Region Control Blocks for
                                ;  use by device handlers.  Handler XM regions not supported on Pro/TSX-Plus.
                                ;
                                        .IF     NE,<PROASM-1>   ;Only for 11's
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to start of memory area where RCB's are to be built.
                                ;
                                ;  Outputs:
                                ;   R5 = Pointer past end of RCB area.
                                ;
010374  010246                  ALCHRB: MOV     R2,-(SP)
                                ;
                                ;  Get count of # RCB's to build
                                ;
010376  013700  000000G                 MOV     NDVRCB,R0       ;Get # RCB's to build for handlers
                                ;
                                ;  Store pointer to start of RCB area and store -1 at beginning of area
                                ;
010402  010537  000000G                 MOV     R5,HANRCB       ;Start of RCB area
010406  010537  000000G                 MOV     R5,HANRCO       ;Store offset relative to MONVEC
010412  162737  000000G 000000G         SUB     #MONVEC,HANRCO  ;Convert address to offset
010420  012725  177777                  MOV     #-1,(R5)+       ;Store -1 at start of area
                                ;
                                ;  Allocate and initialize to zero the RCB's
                                ;
010424  012702  000005          1$:     MOV     #5.,R2          ;Each RCB has 5 words
010430  005025                  2$:     CLR     (R5)+           ;Zero the RCB
010432  077202                          SOB     R2,2$
                                ;
                                ;  See if there are more RCB's to build
                                ;
010434  005300                          DEC     R0              ;More RCB's to initialize?
010436  003372                          BGT     1$              ;Loop if yes
                                ;
                                ;  Store -1 at end of RCB area
                                ;
010440  012725  177777                  MOV     #-1,(R5)+       ;Mark end of RCB list
                                ;
                                ;  Finished
                                ;
010444  012602                          MOV     (SP)+,R2
010446  000207                          RETURN

                                        .IFF    ;NE,<PROASM-1>  ;Dummy for Pro-only
                                ALCHRB: RETURN
                                        .ENDC   ;NE,<PROASM-1>
```

```
                                                .IF     NE,<PROASM-1>   ;If not assembling for Pro only
                                                .SBTTL  LINCHK -- Check validity of T/S line
                                        ;-------------------------------------------------------------------------------
                                        ;  LINCHK is called to check the validity of specified T/S line
                                        ;  vector and status register addresses.
                                        ;  If an uninstalled line is detected this routine aborts if
                                        ;  INIABT=1 or sets the $DEAD flag for the line if INIABT=0.
                                        ;
010450  010146                          LINCHK: MOV     R1,-(SP)
010452  010246                                  MOV     R2,-(SP)
010454  010346                                  MOV     R3,-(SP)
010456  010446                                  MOV     R4,-(SP)
010460  013746  000004                          MOV     @#4,-(SP)       ;SAVE ORIGINAL TRAP VECTOR
                                        ;
                                        ;  Take over trap control
                                        ;
010464  012737  010664' 000004                  MOV     #6$,@#4         ;CATCH TRAPS
                                        ;
                                        ;  Loop through the test for each line.
                                        ;
010472  012701  000000G                         MOV     #LSTLIN,R1      ;NUMBER OF LAST LINE
                                        ;
                                        ;  Determine if this is a primary line or an I/O line and set the
                                        ;  addresses of the interrupt service routines.
                                        ;
010476  004737  006214'                 1$:     CALL    LINTYP          ;Determine the type of this line
010502  032761  000000G 000000G                 BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
010510  001440                                  BEQ     31$             ;Br if not
010512  016103  000000G                         MOV     LMXNUM(R1),R3   ;IS THIS A DL-11 OR MULTIPLEXER LINE?
010516  001411                                  BEQ     2$              ;BR IF DL-11
010520  016302  000000G                         MOV     MXCSR(R3),R2    ;GET DZ11 OR DH11 STATUS REGISTER ADDRESS
010524  001403                                  BEQ     11$             ;BR IF ALREADY MARKED AS DEAD
010526  016304  000000G                         MOV     MXVEC(R3),R4    ;GET MUX INTERRUPT VECTOR ADDRESS
010532  000407                                  BR      3$
010534  004737  010636'                 11$:    CALL    4$              ;MARK LINE AS DEAD
010540  000424                                  BR      31$             ;CONTINUE CHECKING TERMINALS
010542  016102  000000G                 2$:     MOV     RSR(R1),R2      ;GET DL-11 STATUS REGISTER ADDRESS
010546  016104  000000G                         MOV     INVEC(R1),R4    ;GET DL-11 INTERRUPT VECTOR ADDRESS
                                        ;  Check validity of status register address.
010552  020227  160000                  3$:     CMP     R2,#160000      ;IS IT IN I/O PAGE?
010556  103445                                  BLO     LINTRP          ;ERROR IF NOT
010560  032702  000007                          BIT     #7,R2           ;IS IT ON 8-BYTE BOUNDARY?
010564  001042                                  BNE     LINTRP          ;ERROR IF NOT
010566  005712                                  TST     @R2             ;TRY TO ACCESS IT AND SEE IF WE TRAP
                                        ;  Check validty of interrupt vector address.
010570  020427  000060                          CMP     R4,#60          ;CAN'T BE BELOW 60
010574  103445                                  BLO     BADVEC
010576  020427  000500                          CMP     R4,#500         ;OR ABOVE 500
010602  103042                                  BHIS    BADVEC
010604  032704  000007                          BIT     #7,R4           ;MUST BE ON 8-BYTE BOUNDARY
010610  001037                                  BNE     BADVEC
                                        ;  This line looks good.  Check next.
010612  162701  000002                  31$:    SUB     #2,R1           ;MORE TO CHECK?
010616  003327                                  BGT     1$              ;BR IF YES
                                        ;
                                        ;  Finished -- all lines look ok.
                                        ;
```

```
010620  012637  000004                  MOV     (SP)+,@#4       ;RESTORE TRAP VECTOR
010624  012604                          MOV     (SP)+,R4
010626  012603                          MOV     (SP)+,R3
010630  012602                          MOV     (SP)+,R2
010632  012601                          MOV     (SP)+,R1
010634  000207                          RETURN
                                ;
                                ;   See if we should abort or just mark the line as dead.
                                ;
010636  105737  000000G         4$:     TSTB    VINABT          ;DOES HE WANT TO ABORT?
010642  001013                          BNE     LINTRP          ;YES
010644  052761  000000G 000000G         BIS     #$DEAD,LSW3(R1) ;MARK LINE AS DEAD
010652  005703                          TST     R3              ;IS THIS A DL11 OR A MUX LINE?
010654  001402                          BEQ     5$              ;BR IF DL11
010656  005063  000000G                 CLR     MXCSR(R3)       ;MARK DZ OR DH AS DEAD
010662  000207                  5$:     RETURN

                                ;
                                ;   Trap occured while trying to access status register.
                                ;
010664  004737  010636'         6$:     CALL    4$              ;REPORT ERROR OR MARK AS DEAD LINE
010670  000002                          RTI                     ;RETURN TO LINE CHECKING
                                ;
                                ;   Error: Invalid status register address.
                                ;   R1 = Line number,  R2 = status register address
                                ;
010672                          LINTRP: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
010700                                  .PRINT  #BADLIN
010706  000407                          BR      ERP
                                ;
                                ;   Error: Invalid interrupt vector address.
                                ;   R1 = Line number,  R4 = interrupt vector address
                                ;
010710                          BADVEC: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
010716                                  .PRINT  #BDVMSG
010724  010402                          MOV     R4,R2           ;GET VECTOR ADDRESS TO R2
010726  010200                  ERP:    MOV     R2,R0           ;GET ADDRESS TO R0
010730  004737  027704'                 CALL    PRTOCT          ;PRINT OCTAL VALUE
010734                                  .PRINT  #CRLF
010742                                  .PRINT  #BDLMSG         ;LINE # =
010750  010100                          MOV     R1,R0           ;GET LINE NUMBER
010752  006200                          ASR     R0              ;# 1
010754  004737  027754'                 CALL    PRTDEC          ;PRINT LINE NUMBER
010760                                  .PRINT  #CRLF
010766  000137  004102'                 JMP     INISTP          ;ABORT INITIALIZATION
                                        .ENDC   ;NE,<PROASM-1>
                                        .SBTTL  OPNSWP -- Open system swap file
                                ;---------------------------------------------------------------------------
                                ;  OPNSWP is called to open the TSX job swap file.
                                ;  It also assigns swap file slots for each line.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of base of free memory region
                                ;
                                ;  Outputs:
                                ;   SWPCHN = Set up for access to swap file.
                                ;   LSWPBK(i) = Starting block number in swap file for swap area for line.
                                ;
010772  010146                  OPNSWP: MOV     R1,-(SP)
010774  010246                          MOV     R2,-(SP)
010776  010346                          MOV     R3,-(SP)
                                ;
                                ;  Load RT-11 handler for swap device.
                                ;
011000  013700  000000G                 MOV     SWDBLK,R0           ;Get name of device
011004  004737  027514'                 CALL    RTFTCH              ;Fetch the RT-11 handler
011010  103546                          BCS     11$                 ;Br if invalid device
                                ;
                                ;  Compute the maximum number of slots in swap file that we could need
                                ;
011012  012703  000000C                 MOV     #NSL+NDL,R3         ;Get # virtual lines and detached jobs
011016  012701  000000G                 MOV     #LSTPL,R1           ;Get index to last primary line
011022  032761  000000G 000000G 1$:     BIT     #$DEAD,LSW3(R1) ;Is this line installed?
011030  001001                          BNE     5$                  ;Br if not
011032  005203                          INC     R3                  ;Count another primary line
011034  162701  000002          5$:     SUB     #2,R1               ;Get next line index
011040  003370                          BGT     1$                  ;Loop if more lines to check
011042  020337  000000G                 CMP     R3,VSWPSL           ;Compare with # slots specified
011046  002002                          BGE     6$                  ;Br if VSWPSL value is ok
011050  010337  000000G                 MOV     R3,VSWPSL           ;Reduce number of slots in swap file
                                ;
                                ;  Determine how many blocks are needed for each slot in swap file.
                                ;
011054  013703  000000G         6$:     MOV     VHIMEM,R3           ;GET # BLOCKS NEEDED FOR LARGEST JOB SIZE
011060  006303                          ASL     R3
011062  063703  000000G                 ADD     CXTPAG,R3           ;ADD # BLOCKS NEEDED FOR JOB CONTEXT AREA
011066  010337  000000G                 MOV     R3,SLTSIZ           ;Save size of swap file slot
                                ;
                                ;  Compute the total number of blocks needed for the swap file.
                                ;
011072  070337  000000G                 MUL     VSWPSL,R3           ;Multiply by # slots in swap file
                                ;
                                ;  R3 now contains total number of blocks needed in swap file.
                                ;  See if swap file already exists on disk.
                                ;
011076                          4$:     .LOOKUP #AREA,#1,#SWDBLK;DOES SWAP FILE EXIST NOW?
011116  103415                          BCS     2$                  ;BR IF NOT
                                ;  Swap file exists.  See if it is the right size.
011120  020003                          CMP     R0,R3               ;IS SWAP FILE THE RIGHT SIZE?
011122  001453                          BEQ     3$                  ;BR IF YES
                                ;  Old swap file is not of correct size.
                                ;  Delete it and open a new swap file.
011124                                  .CLOSE  #1
```

```
011132                                   .DELETE #AREA,#1,#SWDBLK;DELETE THE OLD SWAP FILE
                         ;
                         ;  Create a new swap file.
                         ;
011152                   2$:     .ENTER  #AREA,#1,#SWDBLK,R3 ;CREATE A NEW SWAP FILE
011176  103443                   BCS     9$                  ;BR IF SOME ERROR ON OPEN
                         ;
                         ;  Swap file has been created.
                         ;  Write to last block to reserve full space in file then close
                         ;  and reopen the channel using a .lookup.
                         ;
011200  005303                   DEC     R3                  ;GET # OF LAST BLOCK IN FILE
011202                           .WRITW  #AREA,#1,#TSINIT,#256.,R3 ;WRITE TO LAST BLOCK IN FILE
011240  005203                   INC     R3                  ;GET BACK # BLOCKS IN FILE
011242                           .CLOSE  #1                  ;CLOSE FILE WE CREATED
011250  000712                   BR      4$                  ;NOW GO REOPEN USING A .LOOKUP
                         ;
                         ;  Swap file has been successfully opened using a .lookup.
                         ;  Now copy channel status to TSX swap channel.
                         ;
011252  012700  000000G  3$:     MOV     #SWPCHN,R0          ;POINT TO SWAP CHANNEL BLOCK
011256  013702  000000G          MOV     SWDBLK,R2           ;GET DEVICE NAME
011262  004737  027134'          CALL    SETCHN              ;SET UP SWAP CHANNEL INFO
                         ;
                         ;  Release the RT-11 device handler
                         ;
011266                           .RELEAS #SWDBLK             ;Release RT-11 device handler
                         ;
                         ;  Finished
                         ;
011276  012603                   MOV     (SP)+,R3
011300  012602                   MOV     (SP)+,R2
011302  012601                   MOV     (SP)+,R1
011304  000207                   RETURN
                         ;
                         ;  Error: Cannot open swap file
                         ;
011306                   9$:     .PRINT  #TSXHD              ;PRINT ERROR MESSAGE
011314                           .PRINT  #BADOPN
011322  004737  011350'          CALL    SPNEED              ;Print info about number of blocks needed
                         ;
                         ;  Error: Invalid device specification.
                         ;
011326  010001           11$:    MOV     R0,R1               ;Save device name
011330                           .PRINT  #TSXHD              ;Print error message
011336                           .PRINT  #BADOPN
011344  004737  011376'          CALL    BADDEV              ;Print invalid device specification
                         ;
                         ;  Error: Number of contiguous blocks required.
                         ;
011350                   SPNEED: .PRINT  #CONSPC             ;Print contiquous blocks needed
011356  010300                   MOV     R3,R0               ;GET # BLOCKS NEEDED FOR FILE
011360  004737  027754'          CALL    PRTDEC              ;DISPLAY # BLOCKS NEEDED
011364                           .PRINT  #CRLF
011372  000137  004102'          JMP     INISTP              ;ABORT INITIALIZATION
                         ;
                         ;  Bad file specification.
```

```
                                        ;
    011376                              BADDEV: .PRINT   #CFHMSG         ;Print invalid device specification
    011404  010100                              MOV      R1,R0           ;Get the rad50 device name
    011406  004737  030020'                     CALL     PRTR50          ;Print rad50 device name
    011412                                      .PRINT   #CRLF           ;Print carriage return/line feed
    011420  000137  004102'                     JMP      INISTP          ;Abort initialization
```

```
                                          .SBTTL  OPNRSF -- Open PLAS region swap file
                                  ;-----------------------------------------------------------------------
                                  ;  OPNRSF is called to open the swap file used for PLAS regions.
                                  ;
                                  ;  Inputs:
                                  ;   R5 = Address of base of free memory area.
                                  ;
                                  ;  Outputs:
                                  ;   SEGCHN = Set up to access swap file.
                                  ;
011424  010346                    OPNRSF: MOV     R3,-(SP)
                                  ;
                                  ;  Return if this is a non-swapping system or if region swap file is
                                  ;  not wanted.
                                  ;
011426  105737  000000G                   TSTB    VSWPFL          ;Is this a non-swapping system?
011432  001513                            BEQ     9$              ;Br if yes
011434  005737  000000G                   TST     VPLAS           ;Is a PLAS swap file wanted?
011440  001510                            BEQ     9$              ;Br if not
                                  ;
                                  ;  Load RT-11 device handler for swap device
                                  ;
011442  013700  000000G                   MOV     RSFBLK,R0       ;Get name of device
011446  004737  027514'                   CALL    RTFTCH          ;Try to fetch the RT-11 device handler
011452  103515                            BCS     11$             ;Br if error on handler fetch
011454  013703  000000G                   MOV     VPLAS,R3        ;Get # blocks in PLAS swap file
                                  ;
                                  ;  See if PLAS swap file already exists on disk
                                  ;
011460                            4$:     .LOOKUP #AREA,#1,#RSFBLK ;Try to find existing PLAS swap file
011500  103416                            BCS     2$              ;Br if file does not now exist
                                  ;
                                  ;  PLAS swap file exists.
                                  ;  See if it is the correct size.
                                  ;
011502  020037  000000G                   CMP     R0,VPLAS        ;Is swap file of the correct size?
011506  001453                            BEQ     3$              ;Br if yes
                                  ;
                                  ;  Old PLAS swap file is not of correct size.
                                  ;  Delete it and open a new swap file.
                                  ;
011510                                    .CLOSE  #1              ;Close and delete the old file
011516                                    .DELETE #AREA,#1,#RSFBLK;Delete the old file
                                  ;
                                  ;  Create new swap file
                                  ;
011536                            2$:     .ENTER  #AREA,#1,#RSFBLK,VPLAS ;Create a new PLAS swap file
011564  103440                            BCS     10$             ;Br if cannot create new file
                                  ;
                                  ;  New swap file has been created.
                                  ;  Write to last block to reserve full file size
                                  ;  and then close and reopen with a lookup.
                                  ;
011566  005303                            DEC     R3              ;Get # of last block in file
011570                                    .WRITW  #AREA,#1,#TSINIT,#256.,R3
011626                                    .CLOSE  #1
011634  000711                            BR      4$              ;Go back and lookup file
```

```
                                        ;
                                        ;  Swap file has been successfully opened using lookup.
                                        ;  Copy channel status to TSX channel block.
                                        ;
011636  012700  000000G                 3$:     MOV     #SEGCHN,R0      ;Point to TSX PLAS swap channel
011642  013703  000000G                         MOV     RSFBLK,R3       ;Get device name
011646  004737  027134'                         CALL    SETCHN          ;Set up TSX channel block
                                        ;
                                        ;  Release the RT-11 device handler
                                        ;
011652                                          .RELEAS #RSFBLK         ;Release RT-11 device handler
                                        ;
                                        ;  Finished
                                        ;
011662  012603                          9$:     MOV     (SP)+,R3
011664  000207                                  RETURN
                                        ;
                                        ;  Error -- Cannot open PLAS swap file
                                        ;
011666                                  10$:    .PRINT  #TSXHD          ;Print error prefix
011674                                          .PRINT  #RSFERR         ;Print error message
011702  004737  011350'                         CALL    SPNEED          ;Print information about amt of space needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
011706  010001                          11$:    MOV     R0,R1           ;Save device name
011710                                          .PRINT  #TSXHD          ;Print error message
011716                                          .PRINT  #RSFERR
011724  004737  011376'                         CALL    BADDEV          ;Print invalid device specification
```

```
                                       .SBTTL  SPLINI -- Initialize spooling system
                                ;-----------------------------------------------------------------------
                                ;  SPLINI performs the initialization of the spooling system.
                                ;  Inputs:
                                ;   R5 = Current base of free memory area.
                                ;
011730  005727  000000G         SPLINI: TST     #SPLND          ;Are there any spooled devices?
011734  001401                          BEQ     13$             ;Br if not
011736  000401                          BR      10$             ;Initialize the spooled devices
011740  000207                  13$:    RETURN
                                ;
                                ;  There are some spooled devices
                                ;
011742  010146                  10$:    MOV     R1,-(SP)
011744  010246                          MOV     R2,-(SP)
011746  010346                          MOV     R3,-(SP)
011750  010446                          MOV     R4,-(SP)
011752  010546                          MOV     R5,-(SP)
                                ;
                                ;  Open each spooled device
                                ;
011754  105037  000000G                 CLRB    NSPLDV          ;INIT COUNT OF # ACTUAL SPOOLED DEVICES
011760  012701  000000G                 MOV     #SDCB,R1        ;POINT TO 1ST SDCB
011764  012703  000000G                 MOV     #SPLDEV,R3      ;POINT TO TABLE OF RAD50 DEV NAMES
011770  012704  000000G                 MOV     #SPLANM,R4      ;POINT TO TABLE OF ASCII DEV NAMES
011774  004737  013002'         2$:     CALL    FORCE0          ;FORCE UNIDENTIFIED UNIT #S TO 0
012000  011302                          MOV     (R3),R2         ;GET RAD50 NAME OF SPOOLED DEVICE
012002  010261  000000G                 MOV     R2,SDNAME(R1)   ;SET NAME IN SDCB
012006  010100                          MOV     R1,R0           ;GET ADDRESS OF SDCB
012010  062700  000000G                 ADD     #SDANAM,R0      ;POINT TO CELL FOR ASCII NAME
012014  112420                          MOVB    (R4)+,(R0)+     ;MOVE IN ASCII DEVICE NAME
012016  112420                          MOVB    (R4)+,(R0)+
012020  112420                          MOVB    (R4)+,(R0)+
012022  020227  000000G                 CMP     R2,#DMYDEV      ;Is this a dummy entry for later patching?
012026  001451                          BEQ     1$              ;Br if yes -- Ignore it
012030  010200                          MOV     R2,R0           ;Get name to R0
012032  004737  012704'                 CALL    CVTDVU          ;Convert name to device # and unit #
012036  010061  000000G                 MOV     R0,SDDVU(R1)    ;Store device # and unit # in SDCB
012042  010200                          MOV     R2,R0           ;Get device name
012044  004737  012600'                 CALL    CHKCLD          ;See if this is a CL device?
012050  103406                          BCS     14$             ;Br if not
012052  004737  012514'                 CALL    SPLCLD          ;Set up for spooling to CL device
012056  103414                          BCS     3$              ;Br if invalid unit
012060  105237  000000G                 INCB    NSPLDV          ;Count # of actual spooled devices
012064  000432                          BR      1$              ;Process next device
012066  010100                  14$:    MOV     R1,R0           ;Get address of SDCB
012070  062700  000000G                 ADD     #SDCHAN,R0      ;Point to channel block within SDCB
012074  004737  027046'                 CALL    OPNCHN          ;Set TSX-Plus channel block open to device
012100  103403                          BCS     3$              ;Br if did not recognize device
012102  105237  000000G                 INCB    NSPLDV          ;Count # actual spooled devices
012106  000421                          BR      1$              ;GO PROCESS NEXT DEVICE
                                ;
                                ;  Error on opening spooled device
                                ;  Determine if we should print an error message or simply
                                ;  mark the spooled device as unavailable.
                                ;
012110  012761  000000G 000000G 3$:     MOV     #DMYDEV,SDNAME(R1);SAY THIS DEVICE IS NOT SPOOLED
```

```
     58 012116  105737  000000G                TSTB    VINABT          ;ABORT OR CONTINUE ON ERRORS?
     59 012122  001413                         BEQ     1$              ;BR TO IGNORE DEVICE AND CONTINUE INIT
     60 012124                                 .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     61 012132                                 .PRINT  #BDSPOP
     62 012140  010200                         MOV     R2,R0           ;GET RAD50 DEVICE NAME
     63 012142  004737  030020'                CALL    PRTR50          ;PRINT DEVICE NAME
     64 012146  000137  004102'                JMP     INISTP          ;ABORT INITIALIZATION
     65                                 ;
     66                                 ;  Process next spooled device
     67                                 ;
     68 012152  062701  000000G         1$:     ADD     #SDCBSZ,R1      ;POINT TO NEXT SDCB
     69 012156  005723                         TST     (R3)+           ;POINT TO NEXT DEVICE NAME
     70 012160  020327  000000G                CMP     R3,#SPLDVN      ;OPENED ALL SPOOLED DEVICES?
     71 012164  103703                         BLO     2$              ;BR IF MORE TO DO
     72                                 ;
     73                                 ;  Open the spool file
     74                                 ;
     75 012166  105737  000000G                TSTB    NSPLDV          ;ARE THERE ANY ACTUAL SPOOLED DEVICES?
     76 012172  001521                         BEQ     12$             ;BR IF THERE ARE NO ACTUAL SPOOLED DEVICES
     77 012174  013700  000000G                MOV     SPLBLK,R0       ;Get name of device for spool file
     78 012200  004737  027514'                CALL    RTFTCH          ;Fetch the RT-11 device handler
     79 012204  103532                         BCS     11$             ;Br if cannot fetch handler
     80 012206  013702  000000G                MOV     NSPLBL,R2       ;GET # BLOCKS TO ALLOCATE FOR FILE
     81 012212  005202                         INC     R2              ;Add 1 extra block
     82                                 ;
     83                                 ;  See if spool file already exists
     84                                 ;
     85 012214                          5$:     .LOOKUP #AREA,#1,#SPLBLK;SEE IF SPOOL FILE ALREADY EXISTS
     86 012234  103415                         BCS     6$              ;BR IF IT DOES NOT EXIST
     87 012236  020002                         CMP     R0,R2           ;IS IT THE RIGHT SIZE?
     88 012240  001453                         BEQ     7$              ;BR IF YES
     89 012242                                 .CLOSE  #1              ;IT EXISTS BUT IS OF WRONG SIZE
     90 012250                                 .DELETE #AREA,#1,#SPLBLK;DELETE CURRENT FILE AND OPEN NEW ONE
     91                                 ;
     92                                 ;  Open new spool file
     93                                 ;
     94 012270                          6$:     .ENTER  #AREA,#1,#SPLBLK,R2;CREATE A NEW SPOOL FILE
     95 012314  103456                         BCS     8$              ;BR IF ERROR ON ENTER
     96                                 ;  Write to last block in file to reserve full file space
     97 012316  010203                         MOV     R2,R3           ;Get # of blocks in file
     98 012320  005303                         DEC     R3              ;Get # of last block in file
     99 012322                                 .WRITW  #AREA,#1,#TSINIT,#256.,R3
    100                                 ;  Now close and reopen using a lookup
    101 012360                                 .CLOSE  #1              ;CLOSE SPOOL FILE
    102 012366  000712                         BR      5$              ;GO BACK AND REOPEN USING LOOKUP
    103                                 ;
    104                                 ;  Spool file has been successfully opened with a lookup.
    105                                 ;  Save the channel status
    106                                 ;
    107 012370  012700  000000G         7$:     MOV     #SPLCHN,R0      ;SAVE CHANNEL STATUS HERE
    108 012374  013702  000000G                MOV     SPLBLK,R2       ;GET DEVICE NAME
    109 012400  004737  027134'                CALL    SETCHN          ;SAVE CHANNEL STATUS
    110 012404                                 .RELEAS #SPLBLK         ;Release the RT-11 device handler
    111                                 ;
    112                                 ;  Set number of free public blocks in spool file
    113                                 ;
    114 012414  113703  000000G                MOVB    NSPLDV,R3       ;Get # spooled devices
```

```
012420  070327  000000G                  MUL     #PVSPBL,R3        ;Times number of private blocks per dev
012424  005403                           NEG     R3
012426  063703  000000G                  ADD     NSPLBL,R3                 ;Get # public spool blocks
012432  010337  000000G                  MOV     R3,NFRESB         ;This is number of public free spool blocks
                                 ;
                                 ;   Finished
                                 ;
012436  012605                   12$:    MOV     (SP)+,R5
012440  012604                           MOV     (SP)+,R4
012442  012603                           MOV     (SP)+,R3
012444  012602                           MOV     (SP)+,R2
012446  012601                           MOV     (SP)+,R1
012450  000207                   9$:     RETURN
                                 ;
                                 ;   Error: Cannot open spool file.
                                 ;
012452                           8$:     .PRINT  #TSXHD            ;PRINT ERROR MESSAGE
012460                                   .PRINT  #BOSF
012466  000137  004102'                  JMP     INISTP            ;ABORT INITIALIZATION
                                 ;
                                 ;   Error: Invalid device specification.
                                 ;
012472  010001                   11$:    MOV     R0,R1             ;Save device name
012474                                   .PRINT  #TSXHD            ;Print error message
012502                                   .PRINT  #BOSF
012510  004737  011376'                  CALL    BADDEV            ;Print invalid device specification
```

```
                                   .SBTTL  SPLCLD -- Set up spooling to a CL device
                           ;----------------------------------------------------------------------
                           ;  SPLCLD is called to set up a spool device control block when
                           ;  spooling is being directed to a Communication Line (CL) device.
                           ;
                           ;  Inputs:
                           ;   R0 = CL unit number
                           ;   R1 = Address of SDCB
                           ;
                           ;  Outputs:
                           ;   C-flag set ==> Invalid CL unit
                           ;
012514  010546             SPLCLD: MOV     R5,-(SP)
                           ;
                           ;  Make sure CL unit number is valie
                           ;
012516  010005                     MOV     R0,R5            ;Get CL unit number
012520  020527  000000G            CMP     R5,#CLTOTL       ;Is this a valid unit #
012524  103022                     BHIS    8$               ;Br if invalid
                           ;
                           ;  Set up channel control block in SDCB
                           ;
012526  005061  000000C            CLR     SDCHAN+C.SBLK(R1);Starting block # = 0
012532  020527  000007             CMP     R5,#7            ;Is this a CL or C1 unit?
012536  101405                     BLOS    1$               ;Br if CL unit
012540  162705  000010             SUB     #8.,R5           ;Remove C1 unit # bias
012544  013700  000000G            MOV     C1DEVX,R0        ;Get C1 device index number
012550  000402                     BR      2$
012552  013700  000000G    1$:     MOV     CLDEVX,R0        ;Get CL device index number
012556  010061  000000C    2$:     MOV     R0,SDCHAN+C.CSW(R1);Set device index number
012562  110561  000000C            MOVB    R5,SDCHAN+C.DEVQ(R1);Set unit #
                           ;
                           ;  We successfully set up a CL unit
                           ;
012566  000241                     CLC                      ;Signal success on error
012570  000401                     BR      9$
                           ;
                           ;  We cannot open this CL unit
                           ;
012572  000261             8$:     SEC                      ;Signal error
                           ;
                           ;  Finished
                           ;
012574  012605             9$:     MOV     (SP)+,R5
012576  000207                     RETURN
```

```
                                .SBTTL  CHKCLD -- See if a device name is a CL or C1 unit
                        ;---------------------------------------------------------------------------
                        ;  Determine if a rad50 device and unit name is a CL or C1 device.
                        ;
                        ;  Inputs:
                        ;   R0 = Rad50 device spec
                        ;
                        ;  Outputs:
                        ;   C-flag set     ==> Not a CL or C1 unit
                        ;   C-flag cleared ==> This is a CL or C1 unit
                        ;   R0 = CL unit number (0-15)
                        ;
012600                  CHKCLD:
                        ;
                        ;  See if this is a CL unit
                        ;
012600  020027  012240          CMP     R0,#R50CL       ;Is name "CL"?
012604  001411                  BEQ     1$              ;Br if yes
012606  020027  012276          CMP     R0,#R50CL0      ;Is name in the range CL0 to CL7?
012612  103432                  BLO     8$              ;Br if not
012614  020027  012305          CMP     R0,#R50CL7
012620  101005                  BHI     2$
012622  162700  012276          SUB     #R50CL0,R0      ;Get CL unit number
012626  000422                  BR      7$
012630  005000          1$:     CLR     R0              ;CL = CL0
012632  000420                  BR      7$
                        ;
                        ;  See if this is a C1 unit
                        ;
012634  020027  013630  2$:     CMP     R0,#R50C1       ;Is name "C1"?
012640  001413                  BEQ     3$              ;Br if yes
012642  020027  013666          CMP     R0,#R50C10      ;Is name in the range C10 to C17?
012646  103414                  BLO     8$              ;Br if not
012650  020027  013675          CMP     R0,#R50C17
012654  101011                  BHI     8$
012656  162700  013666          SUB     #R50C10,R0      ;Get unit number
012662  062700  000010          ADD     #8.,R0          ;Bias by 8 for C1 units
012666  000402                  BR      7$
012670  012700  000010  3$:     MOV     #8.,R0          ;C1 = CL8
                        ;
                        ;  This is a CL or C1 unit
                        ;
012674  000241          7$:     CLC                     ;Signal success on return
012676  000401                  BR      9$
                        ;
                        ;  This is not a CL or C1 unit
                        ;
012700  000261          8$:     SEC                     ;Signal failure on return
                        ;
                        ;  Finished
                        ;
012702  000207          9$:     RETURN
```

```
                                        .SBTTL  CVTDVU -- Convert device name to dev index and unit #
                                ;-----------------------------------------------------------------------
                                ;  CVTDVU is called to convert a RAD50 device name into the corresponding
                                ;  device index number and unit number.
                                ;
                                ;  Inputs:
                                ;   R0 = RAD50 device name.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Conversion successful.
                                ;   C-flag set     ==> Unable to find device name in tables.
                                ;   R0 = Device index number (low byte), device unit number (high byte).
                                ;
012704  010246                  CVTDVU: MOV     R2,-(SP)
012706  010346                          MOV     R3,-(SP)
                                ;
                                ;  Split the unit number off of the full device name
                                ;
012710  010003                          MOV     R0,R3           ;Get full device name
012712  005002                          CLR     R2              ;Set up for divide
012714  071227  000050                  DIV     #50,R2          ;Split name and unit (R0=name, R1=unit)
012720  005703                          TST     R3              ;Was a unit number specified?
012722  001402                          BEQ     1$              ;Br if not
012724  162703  000036                  SUB     #36,R3          ;Convert unit number to binary value
012730  010300                  1$:     MOV     R3,R0           ;Get unit number
012732  000300                          SWAB    R0              ;Position to high-order byte
                                ;
                                ;  Look up the device name to get the device index
                                ;
012734  070227  000050                  MUL     #50,R2          ;Now get the device name without unit number
012740  013702  000000G                 MOV     NUMDEV,R2       ;Get index number of last device
012744  020362  000000G         2$:     CMP     R3,PNAME(R2)    ;Search for device in name table
012750  001407                          BEQ     3$              ;Br if found it
012752  162702  000002                  SUB     #2,R2           ;Try next device
012756  002372                          BGE     2$              ;Loop if more to check
                                ;
                                ;  Error, cannot find device name in tables
                                ;
012760  012700  177777                  MOV     #-1,R0          ;Set device # = unit # = -1
012764  000261                          SEC                     ;Signal error on return
012766  000402                          BR      9$
                                ;
                                ;  Found the device in the tables
                                ;
012770  050200                  3$:     BIS     R2,R0           ;Combine device # and unit #
012772  000241                          CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
012774  012603                  9$:     MOV     (SP)+,R3
012776  012602                          MOV     (SP)+,R2
013000  000207                          RETURN
```

```
                                         .SBTTL  FORCE0 -- Force a 2-char dev name to unit 0
                                 ;---------------------------------------------------------------------
                                 ;  Inputs: R3 points to a RAD50 device name
                                 ;
                                 ;  Outputs: If the 3rd char of the device name pointed to by R3 is
                                 ;       blank, then it is changed to 0
                                 ;
 013002  010346                  FORCE0: MOV     R3,-(SP)
 013004  010446                          MOV     R4,-(SP)
 013006  010546                          MOV     R5,-(SP)
 013010  011305                          MOV     (R3),R5         ;MOVE CURRENT DEV NAME TO R5
 013012  005004                          CLR     R4              ;SET UP FOR DIVIDE
 013014  071427  000050                  DIV     #50,R4          ;SEPARATE INTO NAME AND UNIT
 013020  005705                          TST     R5              ;WAS 3RD CHAR BLANK?
 013022  001012                          BNE     9$              ;RETURN IF NOT
 013024  010405                          MOV     R4,R5           ;GET HIGH 2 CHARS
 013026  005004                          CLR     R4              ;SET UP FOR ANOTHER DIVIDE
 013030  071427  000050                  DIV     #50,R4          ;SEPARATE 1 & 2 CHARS
 013034  005704                          TST     R4              ;WAS CHAR 1 BLANK?
 013036  001404                          BEQ     9$              ;EMPTY OR INVALID DEV NAME!
 013040  005705                          TST     R5              ;WAS CHAR 2 BLANK?
 013042  001402                          BEQ     9$              ;1-CHAR DEV NAME SHOULD BE INVALID???
 013044  062713  000036                  ADD     #^R  0,(R3)     ;FORCE TO UNIT 0
 013050  012605                  9$:     MOV     (SP)+,R5
 013052  012604                          MOV     (SP)+,R4
 013054  012603                          MOV     (SP)+,R3
 013056  000207                          RETURN
```

```
                                   .SBTTL  ALOCBF -- Allocate buffer space
                           ;-----------------------------------------------------------------------
                           ;  ALOCBF is called to allocate space for buffers.  The allocated space
                           ;  is not initialized but simply reserved.
                           ;
                           ;  Inputs:
                           ;   R5 = Start of area to allocate buffer space in.
                           ;
                           ;  Outputs:
                           ;   R5 = Address beyond end of buffer area.
                           ;   CHNBAS = Address of base of I/O channel space.
                           ;   CHNEND = Address past end of I/O channel space.
                           ;
013060  010146             ALOCBF: MOV     R1,-(SP)
                           ;
                           ;  Assign space for I/O queue elements.
                           ;
013062  010537  000000G            MOV     R5,FREIOQ       ;START OF I/O QUEUE SPACE
013066  062705  000000C            ADD     #IOQSIZ*NUMIOQ,R5;RESERVE SPACE FOR I/O QUEUE ELEMENTS
                           ;
                           ;  Assign space for shared PLAS region control blocks
                           ;
013072  010537  000000G            MOV     R5,SHRRCB       ;Start of area for RCB's
013076  013701  000000G            MOV     VNGR,R1         ;Get number of RCB's wanted
013102  020137  000000G            CMP     R1,VMXWIN       ;Must have one for each display window
013106  103002                     BHIS    13$             ;Br if ok
013110  013701  000000G            MOV     VMXWIN,R1       ;Force one for each window
013114  070127  000000G    13$:    MUL     #RC$$SZ,R1      ;Multiply by size of each block
013120  060105                     ADD     R1,R5           ;Allocate space for RCB's
013122  010537  000000G            MOV     R5,SHRRCN       ;Address of end of region
                           ;
                           ;  Assign space for fork blocks
                           ;
013126  012700  000000C            MOV     #<NUMFRK-FRKGEN>,R0 ;Get # fork blocks to allocate
013132  003404                     BLE     11$             ;Br if none to allocate
013134  010537  000000G            MOV     R5,FRKINI       ;Set pointer to start of area
013140  062705  000000C            ADD     #<<NUMFRK-FRKGEN>*FQ$$SZ>,R5 ;Reserve space for fork blocks
                           ;
                           ;  Assign space for job monitoring control blocks
                           ;
013144  013701  000000G    11$:    MOV     VMXMON,R1       ;Any job monitoring blocks wanted?
013150  001405                     BEQ     10$             ;Br if not
013152  010537  000000G            MOV     R5,MONFQH       ;Start of job monitoring control blocks
013156  070127  000000G            MUL     #JM$$SZ,R1      ;Compute space needed for control blocks
013162  060105                     ADD     R1,R5           ;Allocate the space
                           ;
                           ;  Allocate space for system message buffers.
                           ;
013164  010537  000000G    10$:    MOV     R5,SNMSHD       ;HEAD OF SYSTEM MESSAGE BUFFER AREA
013170  062705  000000C            ADD     #<NMSNMB*SB$$SZ>,R5;RESERVE ROOM FOR MESSAGE BUFFERS
                           ;
                           ;  Allocate space for INSTALLed program table
                           ;
013174  010537  000000G            MOV     R5,INSTBL       ;Base of table
013200  013701  000000G            MOV     VNUIP,R1        ;# slots for user installed programs
013204  062701  000000G            ADD     #NSIP,R1        ;Add # slots for system programs
013210  070127  000000G            MUL     #II$$SZ,R1      ;Multiply by size of each slot
```

```
013214  060105                                  ADD     R1,R5           ;Allocate space for table
013216  010537  000000G                         MOV     R5,INSTBN       ;Pointer past end of INSTALL table
                                        ;
                                        ;       Allocate space for device mount entries
                                        ;
013222  010537  000000G                         MOV     R5,CSHDEV       ;Point to start of area
013226  012701  000000G                         MOV     #CD$$SZ,R1      ;Get size of each entry
013232  070137  000000G                         MUL     VMXCSH,R1       ;Multiply by number of entries
013236  060105                                  ADD     R1,R5           ;Reserve space for table
013240  010537  000000G                         MOV     R5,CSHDVN       ;Save pointer past end of table
                                        ;
                                        ;       Allocate space for data cache control blocks
                                        ;
013244  005737  000000G                         TST     CSHALC          ;Is data caching wanted?
013250  001404                                  BEQ     12$             ;Br if not
013252  010537  000000G                         MOV     R5,CCBHD        ;Head of free list area
013256  062705  000000C                         ADD     #NUMCCB*CC$$SZ,R5 ;Allocate space for control blocks
                                        ;
                                        ;       Allocate space for spool file control blocks
                                        ;
013262  013701  000000G                 12$:    MOV     NSPLFL,R1       ;Get # spool file control blocks needed
013266  001407                                  BEQ     1$              ;Br if none needed
013270  070127  000000G                         MUL     #SFCBSZ,R1      ;Compute space needed by control blocks
013274  010537  000000G                         MOV     R5,SFCB         ;Base of control block area
013300  060105                                  ADD     R1,R5           ;Allocate space for control blocks
013302  010537  000000G                         MOV     R5,SFCBND       ;End of control block area
                                        ;
                                        ;       Allocate space for a 16 byte vector for each multiplexer.
                                        ;       This vector is used to map from the mux line number to the
                                        ;       TSX-Plus logical line number.
                                        ;
013306  012701  000002                  1$:     MOV     #2,R1           ;START WITH FIRST MUX
013312  020127  000000G                 2$:     CMP     R1,#LSTMX       ;HAVE WE DONE ALL MUX'S?
013316  101007                                  BHI     5$              ;BR IF YES
013320  010561  000000G                         MOV     R5,MXLNT(R1)    ;SET ADDRESS OF START OF VECTOR
013324  062705  000020                          ADD     #16.,R5         ;RESERVE SPACE FOR VECTOR
013330  062701  000002                          ADD     #2,R1           ;ADVANCE TO NEXT MUX
013334  000766                                  BR      2$
                                        ;
                                        ;       Allocate buffers to hold characters for DMA transfers to DH11 multiplexers
                                        ;
013336  012701  000002                  5$:     MOV     #2,R1           ;Start with first line
013342  020127  000000G                 6$:     CMP     R1,#LSTPL       ;Is this a primary time-sharing line?
013346  101403                                  BLOS    3$              ;Br if yes
013350  020127  000000G                         CMP     R1,#FSTIOL      ;Is this a CL line?
013354  103422                                  BLO     7$              ;Br if not
013356  026127  000000G 000000G         3$:     CMP     LCDTYP(R1),#CDX$DH ;Is this line connected to a DH11?
013364  001404                                  BEQ     8$              ;Br if yes
013366  026127  000000G 000000G                 CMP     LCDTYP(R1),#CDX$VH ;Is this line connected to a DHV11?
013374  001012                                  BNE     7$              ;Br if not
013376  010561  000000G                 8$:     MOV     R5,LDHB1B(R1)   ;Set address of start of buffer 1
013402  010561  000000G                         MOV     R5,LDHB1P(R1)   ;Initialize pointer into buffer 1
013406  062705  000000G                         ADD     #DHBFSZ,R5      ;Reserve space for buffer
013412  010561  000000G                         MOV     R5,LDHB2B(R1)   ;Set address of start of buffer 2
013416  062705  000000G                         ADD     #DHBFSZ,R5      ;Reserve space for buffer
013422  062701  000002                  7$:     ADD     #2,R1           ;Get # of next line
013426  020127  000000G                         CMP     R1,#LSTHL       ;Have we checked all lines?
```

```
013432  101743                          BLOS    6$              ;Br if not
013434  005205                          INC     R5              ;Bound up to next word
013436  042705  000001                  BIC     #1,R5
                                ;
                                ;  Allocate space for tables that keep track of space in swap file
                                ;
013442  105737  000000G                 TSTB    VSWPFL          ;Is this a swapping system?
013446  001415                          BEQ     14$             ;Br if not
013450  013700  000000G                 MOV     VSWPSL,R0       ;Get # slots in swap file
013454  006300                          ASL     R0              ;Allocate 2 bytes per slot
013456  010537  000000G                 MOV     R5,SWPPOS       ;Start of table with starting block #'s
013462  060005                          ADD     R0,R5           ;Allocate space
013464  010537  000000G                 MOV     R5,SWPJOB       ;Start of table with job #'s
013470  060005                          ADD     R0,R5           ;Allocate space
013472  010537  000000G                 MOV     R5,SCPFHD       ;Pointer to area with command packets
013476  062705  000000C                 ADD     #NSCP*SP$$SZ,R5 ;Allocate space for swap command packets
                                ;
                                ;  Allocate a 512-byte buffer to use to access job context blocks
                                ;
013502  010537  000000G         14$:    MOV     R5,CXTBUF       ;Set address of buffer
013506  062705  001400                  ADD     #1400,R5        ;Reserve space for the buffer
                                ;
                                ;  Make sure TSX is not too big
                                ;
013512  010500                          MOV     R5,R0           ;GET CURRENT MEMORY ADDRESS
013514  004737  027630'                 CALL    CHKMEM          ;CHECK FOR SPACE OVERFLOW
                                ;
                                ;  Finished
                                ;
013520  012601                          MOV     (SP)+,R1
013522  000207                          RETURN
```

```
                                        .SBTTL  ALCSLO -- Allocate silo buffers for lines
                                ;-----------------------------------------------------------------------
                                ;  Allocate the silo buffers that are used to hold characters as they
                                ;  are received from serial lines.
                                ;
                                ;  Inputs:
                                ;   R5 = Current pointer to start of free memory.
                                ;
                                ;  Outputs:
                                ;   R5 = New pointer to start of free memory.
                                ;
013524  010146                  ALCSLO: MOV     R1,-(SP)
013526  010246                          MOV     R2,-(SP)
                                ;
                                ;  Begin loop to check each line
                                ;
013530  012701  000000G                 MOV     #LSTHL,R1       ;Get index to last hardware line
                                ;
                                ;  Only allocate silo buffers for real lines
                                ;
013534  012702  000040          1$:     MOV     #32.,R2         ;Set minimum size for time-sharing lines
013540  020127  000000G                 CMP     R1,#LSTPL       ;Is this a primary line?
013544  101405                          BLOS    2$              ;Br if yes
013546  020127  000000G                 CMP     R1,#FSTIOL      ;Is this a CL line?
013552  103463                          BLO     9$              ;Br if not
013554  012702  000020                  MOV     #16.,R2         ;Set minimum size for CL lines
                                ;
                                ;  Determine how much space to allocate
                                ;
013560  016100  000000G         2$:     MOV     LHIRBA(R1),R0   ;Get requested size
013564  001002                          BNE     8$              ;Br if a size was specified
013566  013700  000000G                 MOV     VNCSLO,R0       ;Use default value
013572  020027  000000G         8$:     CMP     R0,#MAXSLO      ;Constrain size to 255 bytes
013576  101402                          BLOS    3$              ;Br if ok
013600  012700  000000G                 MOV     #MAXSLO,R0      ;Reduce size
013604  020002                  3$:     CMP     R0,R2           ;Compare with acceptable minimum
013606  103001                          BHIS    4$              ;Br if ok
013610  010200                          MOV     R2,R0           ;Use minimum size allowed
013612  010061  000000G         4$:     MOV     R0,LHIRBA(R1)   ;Set # bytes to be allocated for silo
                                ;
                                ;  Allocate the space
                                ;
013616  010561  000000G                 MOV     R5,LHIRBB(R1)   ;Set base address of buffer
013622  010561  000000G                 MOV     R5,LHIRBP(R1)   ;Init pointer where to store next char
013626  010561  000000G                 MOV     R5,LHIRBG(R1)   ;Init pointer where to get next char
013632  010061  000000G                 MOV     R0,LHIRBS(R1)   ;Set # free bytes in buffer
013636  060005                          ADD     R0,R5           ;Allocate space for buffer
013640  010561  000000G                 MOV     R5,LHIRBE(R1)   ;Set address beyond end of buffer
                                ;
                                ;  Set up control information about when to send XON and XOFF
                                ;
013644  006200                          ASR     R0              ;Get 1/2 of buffer size
013646  162700  000002                  SUB     #2,R0           ;Minus two characters
013652  116102  000000G                 MOVB    LHIRBC(R1),R2   ;Get specified size for XOFF point
013656  001002                          BNE     5$              ;Br if value specified
013660  113702  000000G                 MOVB    VNCXOF,R2       ;Try default
013664  020200                  5$:     CMP     R2,R0           ;Is specified value ok?
```

```
013666  101401                          BLOS    6$              ;Br if yes
013670  010002                          MOV     R0,R2           ;No, use size/2-2
013672  110261  000000G         6$:     MOVB    R2,LHIRBC(R1)   ;Set # of chars when XOFF sent
013676  116102  000001G                 MOVB    LHIRBC+1(R1),R2 ;Get specified size for XON point
013702  001002                          BNE     7$              ;Br if a value was specified
013704  113702  000000G                 MOVB    VNCXON,R2       ;Try default
013710  020200                  7$:     CMP     R2,R0           ;Is specified value ok?
013712  101401                          BLOS    10$             ;Br if ok
013714  010002                          MOV     R0,R2           ;No, use size/2-2
013716  110261  000001G         10$:    MOVB    R2,LHIRBC+1(R1) ;Set # of chars when XON sent
                                ;
                                ;  Do the next line
                                ;
013722  162701  000002          9$:     SUB     #2,R1           ;Get next line index number
013726  003302                          BGT     1$              ;Loop if more to do
                                ;
                                ;  Finished
                                ;
013730  005205                          INC     R5              ;Force R5 to be even
013732  042705  000001                  BIC     #1,R5
013736  012602                          MOV     (SP)+,R2
013740  012601                          MOV     (SP)+,R1
013742  000207                          RETURN
```

```
                                                 .SBTTL  ALBFX  -- Allocate buffers in extended memory region
                                        ;-------------------------------------------------------------------------
                                        ;  ALBFX is called to allocate space for buffers that are not constrained
                                        ;  to fit in the 40Kb region that TSX-Plus occupies.
                                        ;
                                        ;  Inputs:
                                        ;   R5 = 64-Byte address of base of free memory region.
                                        ;
                                        ;  Outputs:
                                        ;   R5 = Address above top of buffers allocated.
                                        ;
013744  010146                          ALBFX:  MOV     R1,-(SP)
013746  010246                                  MOV     R2,-(SP)
013750  010346                                  MOV     R3,-(SP)
                                        ;
                                        ;  Allocate character buffers for all lines
                                        ;  Note: Character buffer space will be accessed by mapping through PAR 6.
                                        ;
013752  012701  000002                          MOV     #2,R1             ;GET 1ST JOB INDEX NUMBER
013756  032761  000000G 000000G 3$:             BIT     #$DEAD,LSW3(R1)   ;IS THIS LINE INSTALLED?
013764  001047                                  BNE     2$                ;BR IF NOT -- DON'T ALLOCATE ANY BUFFER SPACE
013766  020127  000000G                         CMP     R1,#FSTDL         ;IS THIS A DETACHED JOB LINE?
013772  103403                                  BLO     1$                ;BR IF NOT
013774  020127  000000G                         CMP     R1,#LSTDL         ;DETACHED JOB LINE?
014000  101441                                  BLOS    2$                ;BR IF DETACHED JOB -- DON'T ALLOCATE  BUFFERS
014002  010561  000000G         1$:             MOV     R5,LTTPAR(R1)     ;SET PHYSICAL MEMORY PAR OFFSET FOR BUFFER
014006  012702  000000G                         MOV     #VPAR6,R2         ;GET VIRTUAL MEMORY ADDRESS FOR BASE OF PAR6
014012  010261  000000G                         MOV     R2,LINBUF(R1)     ;INPUT BUFFER STARTS AT BASE OF PAR6 REGION
014016  016100  000000G                         MOV     LINSIZ(R1),R0     ;GET # BYTES FOR INPUT BUFFER
014022  010061  000000G                         MOV     R0,LINSPC(R1)     ;SET # FREE BYTES IN INPUT BUFFER
014026  060002                                  ADD     R0,R2             ;ADVANCE VIRTUAL ADDRESS
014030  010261  000000G                         MOV     R2,LINEND(R1)     ;POINTS PAST END OF INPUT BUFFER
014034  010003                                  MOV     R0,R3             ;Get # bytes in input buffer
014036  062703  000007                          ADD     #7,R3             ;Bound up to multiple of 8
014042  072327  177775                          ASH     #-3,R3            ;Get # bytes needed in activation-flag buffer
014046  060302                                  ADD     R3,R2             ;Reserve space for activation-flag buffer
014050  060300                                  ADD     R3,R0             ;Accumulate total buffer space
014052  010261  000000G                         MOV     R2,LOTBUF(R1)     ;POINTS TO START OF OUTPUT BUFFER AREA
014056  066100  000000G                         ADD     LOTSIZ(R1),R0     ;ACCUMULATE # BYTES IN BOTH BUFFERS
014062  066102  000000G                         ADD     LOTSIZ(R1),R2     ;ADVANCE VIRTUAL ADDRESS COUNTER
014066  010261  000000G                         MOV     R2,LOTEND(R1)     ;SAVE ADDRESS OF END OF OUTPUT BUFFER
014072  062700  000077                          ADD     #77,R0            ;BOUND UP TO MULTIPLE OF 64 BYTES
014076  072027  177772                          ASH     #-6,R0            ;CONVERT TO # 64-BYTE BLOCKS ALLOCATED
014102  060005                                  ADD     R0,R5             ;ADVANCE PHYSICAL MEMORY PAR ADDRESS
014104  062701  000002          2$:             ADD     #2,R1             ;ADVANCE LINE NUMBER
014110  020127  000000G                         CMP     R1,#LSTSL         ;HAVE WE DONE ALL LINES YET?
014114  101720                                  BLOS    3$                ;BR IF MORE TO DO
                                        ;
                                        ;  Allocate space for shared file record locking data structures
                                        ;
014116  005737  000000G                         TST     VMXSF             ;Shared file support wanted?
014122  001451                                  BEQ     5$                ;Br if not
014124  013737  000000G 000000G                 MOV     VNUMDC,NUMDCD     ;Set number of data cache blocks
014132  013737  000000G 000000G                 MOV     VMXSFC,NUMCDB     ;Set number of free CDB's
014140  010537  000000G                         MOV     R5,LOKMEM         ;Set phys address of base of area
014144  012701  000000G                         MOV     #FF$$SZ,R1        ;Size of an FDB
014150  070137  000000G                         MUL     VMXSF,R1          ;Times number of FDB's
```

```
   014154  062701  000000C               ADD     #<NLINES*FW$$SZ>,R1 ;Space needed for wait blocks
   014160  013703  000000G               MOV     VMLBLK,R3          ;Max blocks a CDB may hold locked
   014164  006303                        ASL     R3                 ;Two bytes per entry
   014166  062703  000000G               ADD     #FC$LBN,R3         ;Add base size of a CDB
   014172  070337  000000G               MUL     VMXSFC,R3          ;Times number of shared file channels
   014176  060301                        ADD     R3,R1              ;Accumulate space needed
   014200  012703  000000G               MOV     #DC$$SZ,R3         ;Size of a data cache descriptor
   014204  070337  000000G               MUL     VNUMDC,R3          ;Times number of data cache entries
   014210  060301                        ADD     R3,R1              ;Reserve space for data cache descriptors
   014212  062701  000100                ADD     #64.,R1            ;Bound up to 64 byte unit
   014216  072127  177772                ASH     #-6,R1             ;Convert to # 64 byte units
   014222  042701  176000                BIC     #176000,R1         ;Clear sign extension
   014226  060105                        ADD     R1,R5              ;Reserve space for data structures
   014230  010537  000000G               MOV     R5,LOKCSH          ;Save pointer to start of cache buffer area
   014234  013701  000000G               MOV     VNUMDC,R1          ;# shared-file data cache blocks wanted
   014240  070127  000010                MUL     #8.,R1             ;8 64-byte blocks each (512 bytes each)
   014244  060105                        ADD     R1,R5              ;Reserve space for data cache buffers
                                 ;
                                 ;  Allocate space for mapped I/O buffers
                                 ;
   014246  105737  000000G       5$:     TSTB    MIOFLG             ;Are any mapped I/O buffers needed?
   014252  001415                        BEQ     7$                 ;Br if not
   014254  013701  000000G               MOV     MIOBHD,R1          ;Point to 1st mapped I/O control block
   014260  001412                        BEQ     7$                 ;Br if no more buffers needed
   014262  010561  000000G       6$:     MOV     R5,MI$SBP(R1)      ;Set address of base of buffer
   014266  113703  000000G               MOVB    VMIOSZ,R3          ;Get # blocks for buffer
   014272  072327  000003                ASH     #3,R3              ;Convert to # 64-byte pages
   014276  060305                        ADD     R3,R5              ;Allocate space for buffer
   014300  016101  000000G               MOV     MI$LNK(R1),R1      ;Get address of next control block
   014304  001366                        BNE     6$                 ;Loop if more to allocate
                                 ;
                                 ;  Allocate space for performance monitor data buffer if it is wanted.
                                 ;
   014306  013701  000000G       7$:     MOV     VPMSIZ,R1          ;DID USER GEN IN PERFORMANCE MONITOR FEATURE?
   014312  001412                        BEQ     9$                 ;BR IF NOT
   014314  006201                        ASR     R1                 ;CONVERT BYTES TO WORDS
   014316  010137  000000G               MOV     R1,PMCELS          ;SET INTO CELL IN TSEXEC
   014322  010537  000000G               MOV     R5,PMPAR           ;SET BASE ADDRESS OF PM BUFFER
   014326  062701  000037                ADD     #37,R1             ;BOUND SIZE UP TO 64-BYTE MULTIPLE
   014332  072127  177773                ASH     #-5,R1             ;CONVERT WORDS TO # 64-BYTE BLOCKS
   014336  060105                        ADD     R1,R5              ;ADVANCE FREE MEMORY POINTER
                                 ;
                                 ;  Finished
                                 ;
   014340  012603                9$:     MOV     (SP)+,R3
   014342  012602                        MOV     (SP)+,R2
   014344  012601                        MOV     (SP)+,R1
   014346  000207                        RETURN
```

```
                                        .SBTTL  OPNKMN -- Open channel to TSKMON
                                ;-------------------------------------------------------------------------
                                ;  OPNKMN is called to open an I/O channel to TSKMON SAV file and to
                                ;  set up information about TSKMON.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of base of free memory area
                                ;
                                ;  Outputs:
                                ;   KMNCHN = Saved status of channel to use to access TSKMON SAV file.
                                ;   KMNTOP = Top of memory address for TSKMON.
                                ;   KMNHI  = Top address of TSKMON - KMNBAS.
                                ;   KMNPGS = Number of 256-word memory pages needed for TSKMON & context area
                                ;   KMNSTK = Address of stack to use while TSKMON running.
                                ;   KMNSTR = Starting address of TSKMON.
                                ;
014350  010246                  OPNKMN: MOV     R2,-(SP)
                                ;
                                ;  Lookup TSKMON file.
                                ;
014352                                  .LOOKUP #AREA,#1,#KMNNAM ;TRY TO FILE KMON SAV FILE
014372  103517                          BCS     9$              ;BR IF NOT THERE
                                ;
                                ;  Read block 0 of save file and extract some information.
                                ;
014374  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
014400                                  .READW  #AREA,#1,R2,#256.,#0 ;READ BLOCK 0 OF SAV FILE
                                ;  Determine size of kmon
014434  016200  000050                  MOV     50(R2),R0       ;GET TOP ADDRESS OF KMON
014440  062700  000003                  ADD     #3,R0           ;BOUND UP TO NEXT WORD
014444  042700  000001                  BIC     #1,R0           ;FORCE EVEN
014450  010037  000000G                 MOV     R0,KMNTOP
                                ;  Determine number of 256-word memory pages needed while kmon running.
014454  162700  000000G                 SUB     #KMNBAS,R0      ;BASE ADDRESS OF KMON
014460  010037  000000G                 MOV     R0,KMNHI        ;TOP OF TSKMON - KMNBAS
014464  062700  000777                  ADD     #511.,R0        ;BOUND UP TO PAGE SIZE
014470  000241                          CLC
014472  006000                          ROR     R0              ;CVT TO # WORDS
014474  000300                          SWAB    R0              ;CVT TO # PAGES
014476  042700  177400                  BIC     #^C377,R0
014502  063700  000000G                 ADD     CXTPAG,R0       ;# PAGES NEEDED FOR JOB CONTEXT AREA
014506  010037  000000G                 MOV     R0,KMNPGS       ;# 256-wd pages needed for kmon + context area
                                ;  Determine Kmon stack pointer.
014512  016237  000042  000000G         MOV     42(R2),KMNSTK   ;INITIAL STACK POINTER FOR KMON
                                ;  Determine Kmon starting address.
014520  016237  000040  000000G         MOV     40(R2),KMNSTR   ;STARTING ADDRESS
                                ;
                                ;  Set up demo-system time limit
                                ;  (If this is a demo version of TSX-Plus, the number of minutes the system
                                ;   is to run before it crashes is stored in location 300 of TSKMON)
                                ;
014526  016237  000300  000000G         MOV     300(R2),DTLX    ;SET DEMO TIME-LIMIT
                                ;
                                ;  Now save status of channel so we can do a reopen when we need kmon.
                                ;
014534  012700  000000G                 MOV     #KMNCHN,R0      ;GET KMON CHANNEL SAVE AREA
014540  013702  000072'                 MOV     KMNNAM,R2       ;GET KMON DEVICE NAME
```

```
014544  004737  027134'                 CALL    SETCHN          ;SAVE CHANNEL STATUS
                                ;
                                ;  Lookup CCL.SAV and save channel status for it.
                                ;
014550                                  .LOOKUP #AREA,#1,#CCLNAM;LOOKUP SY:CCL.SAV
014570  103410                          BCS     8$              ;BR IF CAN'T FIND CCL
014572  012700  000000G                 MOV     #CCLSAV,R0      ;CHANNEL SAVE AREA
014576  013702  000102'                 MOV     CCLNAM,R2       ;DEVICE NAME
014602  004737  027134'                 CALL    SETCHN          ;SAVE CHANNEL STATUS
                                ;
                                ;  Finished
                                ;
014606  012602                  10$:    MOV     (SP)+,R2
014610  000207                          RETURN
                                ;
                                ;  Error:  We could not find SY:CCL.SAV
                                ;
014612                          8$:     .PRINT  #TSXHD
014620                                  .PRINT  #NOCCL          ;PRINT ERROR MESSAGE
014626  000137  004102'                 JMP     INISTP          ;ABORT INITIALIZATION
                                ;
                                ;  Error: We could not locate TSKMON SAV file.
                                ;
014632                          9$:     .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
014640                                  .PRINT  #NOKMON
014646  000137  004102'                 JMP     INISTP          ;ABORT INITIALIZATION
```

```
                                        .SBTTL  CLINIT -- Initialize CL handler
                                ;---------------------------------------------------------------------------
                                ;  Perform initialization for CL (Communication Line) handler
                                ;
                                ;  Inputs:
                                ;   R5 = Address of start of free memory area.
                                ;
                                ;  Outputs:
                                ;   R5 = Address of new start of free memory area.
                                ;
014652  010146                  CLINIT: MOV     R1,-(SP)
014654  010246                          MOV     R2,-(SP)
014656  010346                          MOV     R3,-(SP)
                                ;
                                ;  Initialize tables for each CL unit
                                ;
014660  005003                          CLR     R3              ;Accumulate ring buffer sizes in R3
014662  012701  000000C                 MOV     #2*<CLTOTL-1>,R1;Get index # of last CL unit
                                ;
                                ;  See if this CL unit is connected to hardware or is free to be
                                ;  connected later to a time-sharing line.
                                ;
014666  016102  000000G         1$:     MOV     CL$LIX(R1),R2   ;Is this CL unit associated with a line?
014672  001416                          BEQ     5$              ;Br if not
014674  012762  000000G 000000G         MOV     #$SXON,LSW10(R2);Send XON when we start the line
014702  010162  000000G                 MOV     R1,LCLUNT(R2)   ;Associate the CL unit with this line
014706  005762  000000G                 TST     RSR(R2)         ;Does this unit have a specified RSR addr?
014712  001006                          BNE     5$              ;Br if yes
014714  005762  000000G                 TST     LMXNUM(R2)      ;Is this a mux line?
014720  001003                          BNE     5$              ;Br if yes
014722  005061  000000G                 CLR     CL$EPS(R1)      ;Say no endstring buffer
014726  000472                          BR      4$              ;Line is not genned in
                                ;
                                ;  Allocate and set up pointers for the output ring buffers
                                ;
014730  010561  000000G         5$:     MOV     R5,CL$ORB(R1)   ;Start of output ring buffer
014734  010561  000000G                 MOV     R5,CL$ORP(R1)   ;Input character pointer
014740  010561  000000G                 MOV     R5,CL$ORG(R1)   ;Next available character pointer
014744  012700  000000G                 MOV     #CLORSZ,R0      ;Get default output ring buffer size
014750  005702                          TST     R2              ;Is this CL unit connected to a line?
014752  001402                          BEQ     6$              ;Br if not
014754  016200  000000G                 MOV     LOTSIZ(R2),R0   ;Get size of output ring buffer
014760  010061  000000G         6$:     MOV     R0,CL$ORA(R1)   ;Set size of output ring buffer
014764  010061  000000G                 MOV     R0,CL$ORS(R1)   ;Available space in ring buffer
014770  060005                          ADD     R0,R5           ;Point beyond end of ring buffer
014772  010561  000000G                 MOV     R5,CL$ORE(R1)   ;Address past end of ring buffer
014776  060003                          ADD     R0,R3           ;Accumulate size of output ring buffers
                                ;
                                ;  Allocate space for end-of-file string buffer
                                ;
015000  010561  000000G                 MOV     R5,CL$EPS(R1)   ;Set pointer to end-of-file string buffer
015004  005061  000000G                 CLR     CL$EPP(R1)      ;No string to print yet
015010  062705  000001G                 ADD     #<CLEOFS+1>,R5  ;Reserve space for buffer
015014  062703  000001G                 ADD     #<CLEOFS+1>,R3  ;Accumulate buffer space
                                ;
                                ;  Initialize end-of-file form-feed count
                                ;
```

```
 015020  005061  000000G                CLR     CL$EPN(R1)        ;Init ENDPAGE=0
                                ;
                                ;  Initialize option word
                                ;
 015024  012700  000000G                MOV     #<CO$DEF>,R0      ;Get default option flags
 015030  005702                         TST     R2                ;Is this CL unit connected with a line?
 015032  001421                         BEQ     7$                ;Br if not
 015034  016202  000000G                MOV     ILSW2(R2),R2      ;Get line options
 015040  032702  000000G                BIT     #$TAB,R2          ;Does hardware support tabs?
 015044  001402                         BEQ     2$                ;Br if not
 015046  052700  000000G                BIS     #CO$TAB,R0        ;Set hardware-tab flag
 015052  032702  000000G        2$:     BIT     #$FORM,R2         ;Does hardware support form feeds?
 015056  001402                         BEQ     3$                ;Br if not
 015060  052700  000000G                BIS     #CO$FF,R0         ;Set hardware-form-feed flag
 015064  032702  000000G        3$:     BIT     #$8BIT,R2         ;Does hardware want 8 bit support?
 015070  001402                         BEQ     7$                ;Br if not
 015072  052700  000000G                BIS     #CO$8BT,R0        ;Enable 8 bit support for CL line
 015076  010061  000000G        7$:     MOV     R0,CL$OPT(R1)     ;Set options for this CL line
                                ;
                                ;  Initialize page length
                                ;
 015102  012761  000102  000000G        MOV     #66.,CL$LEN(R1) ;Say page length = 66 lines
                                ;
                                ;  Initialize status flags
                                ;
 015110  005061  000000G                CLR     CL$STA(R1)        ;Initialize status flags
                                ;
                                ;  Do next line
                                ;
 015114  162701  000002         4$:     SUB     #2,R1             ;Get index # of next unit
 015120  002262                         BGE     1$                ;Loop if more units to initialize
                                ;
                                ;  Make a device table entry for "CL" device
                                ;
 015122  062737  000002  000000G        ADD     #2,NUMDEV         ;One more device
 015130  013701  000000G                MOV     NUMDEV,R1         ;Get device table index
 015134  010137  000000G                MOV     R1,CLDEVX         ;Remember index number of CL device
 015140  012761  012240  000000G        MOV     #R50CL,PNAME(R1) ;Set device name ("CL")
 015146  012761  000000G 000000G        MOV     #CLSTS,DVSTAT(R1) ;Set dev status flags
 015154  012761  000000C 000000G        MOV     #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ;Device info flags
 015162  005061  000000G                CLR     DEVSIZ(R1)        ;Clear device size
 015166  012761  000006G 000000G        MOV     #CLHEAD+6,HANENT(R1) ;Set handler entry point (4th word)
 015174  062703  000000G                ADD     #CLSIZE,R3        ;Get size of handler
 015200  062703  000000C                ADD     #<<CLTOTL*46.>+<NIOL*32.>>,R3 ;Add size of tables in TSGEN
 015204  010361  000000G                MOV     R3,HANSIZ(R1)     ;Set size of handler
 015210  005205                         INC     R5                ;Make sure free-memory pointer is even
 015212  042705  000001                 BIC     #1,R5
                                ;
                                ;  Make a device table entry for C1 if there are more than 8 CL units
                                ;
 015216  022727  000000G 000010         CMP     #CLTOTL,#8.       ;Are there more than 8 CL units?
 015224  101430                         BLOS    9$                ;Br if not -- Don't need C1
 015226  062737  000002  000000G        ADD     #2,NUMDEV         ;One more device
 015234  013701  000000G                MOV     NUMDEV,R1         ;Get device table index
 015240  010137  000000G                MOV     R1,C1DEVX         ;Remember index number of CL device
 015244  012761  013630  000000G        MOV     #R50C1,PNAME(R1) ;Set device name ("C1")
 015252  012761  000000G 000000G        MOV     #CLSTS,DVSTAT(R1) ;Set dev status flags
```

```
    015260  012761  000000C 000000G          MOV     #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ;Device info flags
    015266  005061  000000G                  CLR     DEVSIZ(R1)        ;Clear device size
    015272  012761  000006G 000000G          MOV     #CLHEAD+6,HANENT(R1) ;Set handler entry point (4th word)
    015300  012761  000004  000000G          MOV     #4.,HANSIZ(R1)    ;Set size of handler
                                     ;
                                     ;  Finished
                                     ;
    015306  012603                   9$:     MOV     (SP)+,R3
    015310  012602                           MOV     (SP)+,R2
    015312  012601                           MOV     (SP)+,R1
    015314  000207                           RETURN
```

```
                                        .SBTTL  INDINI -- Initialize IND program
                                ;-----------------------------------------------------------------------
                                ;  Perform initialization for IND program.
                                ;
                                ;  Outputs:
                                ;  If IND is available, the following information is set up:
                                ;   INDSAV = 5 word .SAVESTATUS block for SY:IND.SAV file
                                ;   INDDBL = Lowest block # within IND.SAV file of data overlay segment.
                                ;   INDDBS = Number of blocks used for data overlay segment.
                                ;   INDTSV = 5 word .SAVESTATUS block for SY:TSXIND.TSX file
                                ;
015316  010246                  INDINI: MOV     R2,-(SP)
015320  010346                          MOV     R3,-(SP)
                                ;
                                ;  Determine if IND support is wanted
                                ;
015322  005037  000000G                 CLR     INDSAV          ;ASSUME IND SUPPORT NOT WANTED
                                ;
                                ;  Lookup SY:IND.SAV file
                                ;
015326  013737  000000G 000220'         MOV     SYNAME,INDNAM   ;LOOK UP IND ON BOOT DEVICE
015334                                  .LOOKUP #AREA,#1,#INDNAM ;TRY TO FIND SY:IND.SAV
015354  103002                          BCC     4$              ;BR IF FOUND IND
015356  000137  016010'                 JMP     9$              ;IF CAN'T FIND IND, THEN NO IND SUPPORT
                                ;
                                ;  Set up information about IND overlay data segment
                                ;
015362  013703  000152'         4$:     MOV     WRKBUF,R3       ;Get pointer to work buffer
015366                                  .READW  #AREA,#1,R3,#256.,#0 ;READ IN BLOCK 0 OF SAV FILE
015422  016302  000064                  MOV     64(R3),R2       ;GET POINTER TO OVERLAY TABLE
015426                                  .READW  #AREA,#1,R3,#256.,#1 ;READ IN BLOCK 1 WITH OVERLAY TABLE
015464  162702  001000                  SUB     #1000,R2        ;GET ADDRESS OF OVERLAY TABLE REL TO BLOCK 1
015470  060302                          ADD     R3,R2           ;ADD BASE ADDRESS WHERE BLOCK 1 DATA IS
015472  011203                          MOV     (R2),R3         ;Get virtual address of segment 0
015474  020312                  5$:     CMP     R3,(R2)         ;Search for 1st segment with different addr
015476  001003                          BNE     6$              ;Br if found it (this is the data segment)
015500  062702  000006                  ADD     #6,R2           ;Point to overlay table entry for next seg
015504  000773                          BR      5$
015506  005722                  6$:     TST     (R2)+           ;Point to word with block # if SAV file
015510  012237  000000G                 MOV     (R2)+,INDDBL    ;GET BLOCK # IN SAV FILE OF DATA OVERLAY
015514  011202                          MOV     (R2),R2         ;GET # OF WORDS IN OVERLAY SEGMENT
015516  062702  000377                  ADD     #255.,R2        ;ROUND UP TO NEXT BLOCK
015522  000302                          SWAB    R2              ;CONVERT # WORDS TO # BLOCKS
015524  042702  177400                  BIC     #^C<377>,R2
015530  010237  000000G                 MOV     R2,INDDBS       ;SAVE # BLOCKS USED FOR DATA OVERLAY
                                ;
                                ;  Do .SAVESTATUS on channel opened to IND.SAV file so that we
                                ;  can do a reopen to access it from KMON.
                                ;
015534  012700  000000G                 MOV     #INDSAV,R0      ;GET ADDRESS OF SAVESTATUS BLOCK
015540  013702  000220'                 MOV     INDNAM,R2       ;GET RAD50 DEVICE NAME
015544  004737  027134'                 CALL    SETCHN          ;SAVE FILE STATUS
                                ;
                                ;  Determine how much space is needed for SY:TSXIND.TSX swap file
                                ;
015550  013703  000000G                 MOV     INDDBS,R3       ;GET # BLOCKS NEEDED PER JOB
015554  070327  000000C                 MUL     #<LSTSL/2>,R3   ;TIMES TOTAL NUMBER OF JOBS
```

```

                                ;
                                ;  Load Rt-11 device handler for ind swap file.
                                ;
015560  013700  000000G                 MOV     INDFIL,R0           ;Get name of the device
015564  004737  027514'                 CALL    RTFTCH              ;Try to fetch the RT-11 device handler
015570  103522                          BCS     11$                 ;Br if error on handler fetch
                                ;
                                ;  See if TSXIND file already exists
                                ;
015572                                  .LOOKUP #AREA,#1,#INDFIL ;DOES SY:TSXIND.TSX FILE EXIST NOW?
015612  103415                          BCS     1$                  ;BR IF NOT
015614  020003                          CMP     R0,R3               ;IS IT OF THE CORRECT SIZE?
015616  001462                          BEQ     2$                  ;BR IF YES
015620                                  .PURGE  #1                  ;FILE IS OF WRONG SIZE
015626                                  .DELETE #AREA,#1,#INDFIL;DELETE OLD FILE
                                ;
                                ;  File does not now exist
                                ;  Create new file
                                ;
015646                          1$:     .ENTER  #AREA,#1,#INDFIL,R3 ;CREATE NEW TSXIND FILE
015672  103451                          BCS     10$                 ;BR IF ERROR ON CREATE
015674  010302                          MOV     R3,R2               ;# BLOCKS IN FILE
015676  005302                          DEC     R2                  ;GET # OF LAST BLOCK IN FILE
015700                                  .WRITW  #AREA,#1,WRKBUF,#256.,R2 ;WRITE TO LAST BLOCK OF FILE
015736                                  .CLOSE  #1                  ;NOW CLOSE THE FILE
015744                                  .LOOKUP #AREA,#1,#INDFIL ;REOPEN TSXIND FILE WITH LOOKUP
                                ;
                                ;  Do .SAVESTATUS for SY:TSXIND.TSX file
                                ;
015764  012700  000000G         2$:     MOV     #INDTSV,R0          ;POINT TO SAVESTATUS BLOCK
015770  013702  000000G                 MOV     INDFIL,R2           ;GET RAD50 DEVICE NAME
015774  004737  027134'                 CALL    SETCHN              ;SAVE FILE INFO
016000                                  .RELEAS #INDFIL             ;Release device handler
                                ;
                                ;  Finished
                                ;
016010  012603                  9$:     MOV     (SP)+,R3
016012  012602                          MOV     (SP)+,R2
016014  000207                          RETURN
                                ;
                                ;  Error occurred while opening SY:TSXIND.TSX file
                                ;
016016                          10$:    .PRINT  #TSXHD              ;Print error message
016024                                  .PRINT  #INDOPN
016032  004737  011350'                 CALL    SPNEED              ;Print info about number of blocks needed
                                ;
                                ;  Error: Invalid device specification.
                                ;
016036  010001                  11$:    MOV     R0,R1               ;Save device name
016040                                  .PRINT  #TSXHD              ;Print error message
016046                                  .PRINT  #INDOPN
016054  004737  011376'                 CALL    BADDEV              ;Print invalid device specification
```

```
                                 .SBTTL  UCLINI -- Initialize TSXUCL data file
                         ;-----------------------------------------------------------------------
                         ;  UCLINI is called to initialize the TSXUCL data file which is used
                         ;  to store user-defined commands.
                         ;
                         ;  Outputs:
                         ;   TSXUCL data file is initialized.
                         ;   UCLBLK = Number of blocks in data file for each job.
                         ;
016060  010246           UCLINI: MOV     R2,-(SP)
016062  010346                   MOV     R3,-(SP)
                         ;
                         ;  Determine if TSXUCL data file is needed
                         ;
016064  105737  000000G          TSTB    VU$CL           ;Is TSXUCL being used at all?
016070  001536                   BEQ     9$              ;Br if not
016072  013702  000000G          MOV     VUCLMC,R2       ;Get maximum number of commands
016076  001533                   BEQ     9$              ;Br if none allowed
                         ;
                         ;  Determine number of blocks needed in data file for each job
                         ;
016100  012700  000000G          MOV     #UK$$SZ,R0      ;Size of each keyword descriptor
016104  062700  000000G          ADD     #US$$SZ,R0      ;Size of each command string descriptor
016110  070200                   MUL     R0,R2           ;Compute total # bytes for keywords+commands
016112  062703  000777G          ADD     #UC$$SZ+511.,R3 ;Add space for control information & round up
016116  005502                   ADC     R2              ;Propogate carry
016120  071227  001000           DIV     #512.,R2        ;Convert to # of blocks needed
016124  010237  000000G          MOV     R2,UCLBLK       ;Save number of blocks needed per job
                         ;
                         ;  Multiply by number of jobs to get total file size
                         ;
016130  070227  000000C          MUL     #<LSTSL/2>,R2   ;Times total number of jobs
                         ;
                         ;  Load Rt-11 device handler for ind swap file.
                         ;
016134  013700  000000G          MOV     UCLDAT,R0       ;Get name of the device
016140  004737  027514'          CALL    RTFTCH          ;Try to fetch the RT-11 device handler
016144  103523                   BCS     11$             ;Br if error on handler fetch
                         ;
                         ;  The total required file size is now in R3.
                         ;  See if the file already exists.
                         ;
016146                          .LOOKUP  #AREA,#1,#UCLDAT ;See if the file exists now
016166  103415                   BCS     1$              ;Br if file does not exist
016170  020003                   CMP     R0,R3           ;Is existing file of correct size?
016172  001446                   BEQ     2$              ;Br if yes -- use the old file
016174                           .PURGE  #1              ;Purge the channel
016202                           .DELETE #AREA,#1,#UCLDAT;Delete the old file
                         ;
                         ;  Create a new data file
                         ;
016222                   1$:     .ENTER  #AREA,#1,#UCLDAT,R3 ;Create new data file
016246  103452                   BCS     10$             ;Br if error creating the file
016250  005303                   DEC     R3              ;Get # of last block in the file
016252                           .WRITW  #AREA,#1,WRKBUF,#256.,R3 ;Write to last block of file
                         ;
                         ;  Translate possible logical device name to physical name and close
```

```
                                        ;  (Physical name is needed for TSXUCL program.)
                                        ;
    016310                              2$:     .CSTAT  #AREA,#1,#NFSBLK        ;GET CHANNEL STATUS INFORMATION
    016330  013702  000032'                     MOV     <NFSBLK+12>,R2  ;FETCH DEVICE NAME IN RAD50
    016334  063702  000030'                     ADD     <NFSBLK+10>,R2  ;ADD IN DEVICE UNIT NUMBER
    016340  062702  000036                      ADD     #^R  0,R2       ;CONVERT UNIT NUMBER TO RAD50
    016344  010237  000000G                     MOV     R2,UCLDAT       ;SET PHYSICAL NAME BACK INTO TSGEN CELL
    016350                                      .CLOSE  #1              ;Close the file
    016356                                      .RELEAS #UCLDAT         ;Release device handler
                                        ;
                                        ;  Finished
                                        ;
    016366  012603                      9$:     MOV     (SP)+,R3
    016370  012602                              MOV     (SP)+,R2
    016372  000207                              RETURN
                                        ;
                                        ;  Error creating the data file
                                        ;
    016374                              10$:    .PRINT  #TSXHD          ;Print error message
    016402                                      .PRINT  #UCLOPN
    016410  004737  011350'                     CALL    SPNEED          ;Print info about number of blocks needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
    016414  010001                      11$:    MOV     R0,R1           ;Save device name
    016416                                      .PRINT  #TSXHD          ;Print error message
    016424                                      .PRINT  #UCLOPN
    016432  004737  011376'                     CALL    BADDEV          ;Print invalid device specification
```

```
                                           .SBTTL  MEMINI -- Initialize memory management
                                ;-----------------------------------------------------------------------
                                ;  Initialize memory management registers for a 1-to-1 mapping.
                                ;  But leave memory management turned off.
                                ;
016436  010146                  MEMINI: MOV     R1,-(SP)
016440  010246                          MOV     R2,-(SP)
016442  010346                          MOV     R3,-(SP)
016444  010446                          MOV     R4,-(SP)
016446  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize all pages for a 1-to-1 mapping.
                                ;
016450  012700  000000G         12$:    MOV     #KPAR0,R0       ;Kernel mode PAR 0
016454  012701  000000G                 MOV     #UPAR0,R1       ;User mode PAR 0
016460  012702  000000G                 MOV     #KPDR0,R2       ;Kernel mode PDR 0
016464  012703  000000G                 MOV     #UPDR0,R3       ;User mode PDR 0
016470  012704  000010                  MOV     #8.,R4          ;Initialize 8 pages
016474  005005                          CLR     R5              ;Set initial PAR value
016476  010520                  2$:     MOV     R5,(R0)+        ;Set kernel PAR
016500  010521                          MOV     R5,(R1)+        ;Set user PAR value
016502  012722  077406                  MOV     #077406,(R2)+   ;Set kernel PDR
016506  012723  077406                  MOV     #077406,(R3)+   ;Set user PDR value
016512  062705  000200                  ADD     #200,R5         ;Advance block number
016516  077411                          SOB     R4,2$           ;Init all pages
                                ;
                                ;  Map kernel mode I/O page (160000) to 17760000.
                                ;
016520  012737  000000G 000000G         MOV     #IOPAGE,@#KPAR7 ;Map I/O page
                                ;
                                ;  Finished
                                ;
016526  012605                          MOV     (SP)+,R5
016530  012604                          MOV     (SP)+,R4
016532  012603                          MOV     (SP)+,R3
016534  012602                          MOV     (SP)+,R2
016536  012601                          MOV     (SP)+,R1
016540  000207                          RETURN
```

```
                                        .SBTTL  MEMTST -- Set up information about available memory space
                                ;-----------------------------------------------------------------------------
                                ;  MEMTST is called to set up information related to memory management.
                                ;  MEMTST performs the following functions:
                                ;   1. Determine how much memory is installed on machine.
                                ;   2. Load Kernel mode mapping registers.
                                ;
                                ;  Inputs:
                                ;   R5 = top of memory currently allocated for TSX and low memory buffers.
                                ;
                                ;  Outputs:
                                ;   PHYMEM = 64-byte block # above top of physical memory.
                                ;   FMEMHI = 64-byte block # above top of memory available for system.
                                ;   Kernel mode mapping registers loaded.
                                ;   Memory management is left turned off.
                                ;

                                ;
                                ;  Offset word to test for memory wrap - choose a location which will not
                                ;  effect RT-11 or TSX-Plus initialization.
                                ;
        000110                  TSTWRD = 110                            ;Offset word to test for memory wrap

016542  010146                  MEMTST: MOV     R1,-(SP)
016544  010246                          MOV     R2,-(SP)
016546  010346                          MOV     R3,-(SP)
016550  010446                          MOV     R4,-(SP)
016552  010546                          MOV     R5,-(SP)
016554  013746  000004                  MOV     @#4,-(SP)       ;Save illegal mem. ref. trap vector
                                ;
                                ;  Determine if this machine has a memory management register # 3.
                                ;  If it does not, then machine cannot possibly have more than 256Kb.
                                ;
016560  012737  017132' 000004          MOV     #TRCSET,@#4     ;Catch trap
016566  000240                          NOP                     ;Clean out 11/73 pipeline
016570  000240                          NOP                     ;Before attempting trap
016572  005737  000000G                 TST     @#SR3MMR        ;Try to access status register 3
016576  103411                          BCS     22$             ;Br if MMU 3 status register is non-existent
016600  105237  000000G                 INCB    SR3FLG          ;No trap.  We must have SR3
                                ;
                                ;  Now determine if it implements D-space (11/23 11/24 do not)
                                ;  If there was no SR3MMR, then it certainly doesn't!
                                ;
016604  000240                          NOP                     ;Clean out the pipeline again?
016606  000240                          NOP
016610  005737  000000G                 TST     @#UDDR0         ;It there a user D-space PDR0?
016614  103402                          BCS     22$             ;Br if not
016616  105237  000000G                 INCB    IDSFLG          ;No trap.  Must implement D-space
                                ;
                                ;  If we are running on a Professional, there is a register that tells
                                ;  us how much memory is installed on the machine.
                                ;
016622                          22$:
                                .IF     NE,PROASM
                                        TSTB    PROFLG          ;Are we running on a Professional?
                                        BEQ     26$             ;Br if not
                                        CLR     R5              ;Load byte without sign extension
```

```
                                                      BISB    @#173050,R5     ;Get 32Kb top of system RAM boundary
                                                      ASH     #9.,R5          ;Convert to # 64 byte blocks
                                                      SUB     #10,R5          ;Don't use the last 512 bytes of memory
                                                      BR      7$
                                              26$:
                                              .ENDC   ;NE,PROASM
                                              .IF     NE,<PROASM-1>           ;Assemble if could be on a PDP-11
                                              ;
                                              ;  We are not running on a Professional.
                                              ;  Test each page above TSX to see where the top of memory is.
                                              ;
016622  012737  017142' 000020                        MOV     #RTNKM,@#20     ;Use IOT instruction to get out of user mode
016630  005037  000022                                CLR     @#22
016634  052737  000000G 000000G                       BIS     #MMENBL,@#SROMMR;Enable memory management
016642  105737  000000G                               TSTB    SR3FLG          ;Does maching have mem management reg # 3?
016646  001403                                        BEQ     4$              ;Br if non-existent
016650  052737  000000G 000000G                       BIS     #EMMAP,@#SR3MMR ;Enable 22-bit extended memory
                                              ;
                                              ;  Map user page 7 to each successive 256-word block and attempt to access.
                                              ;
016656  012705  002000                        4$:     MOV     #1024.,R5       ;Start checking at 64Kb
016662  010537  000000G                       5$:     MOV     R5,@#UPAR7      ;Map user page 7 to page to be tested
016666  052737  000000G 000000G                       BIS     #UMODE,@#PSW    ;Go into user mode
016674  005737  160000                                TST     @#160000        ;Can we access the page?
                                              ;
                                              ;  Use IOT to get back into kernel mode.
                                              ;
016700  000004                                        IOT                     ;Return to kernel mode
016702  103405                                        BCS     6$              ;Br if memory is non-existent
016704  062705  000010                                ADD     #10,R5          ;Go try next page
016710  020527  177600                                CMP     R5,#177600      ;Don't enter I/O page
016714  103762                                        BLO     5$
                                              ;
                                              ;  Check for potential memory wrap (on 18-bit 256K byte computers).
                                              ;
016716  020527  010000                        6$:     CMP     R5,#10000       ;Is physical memory above 256K bytes
016722  101421                                        BLOS    7$              ;Br if below 256K bytes
016724  005037  000110                                CLR     @#TSTWRD        ;Clear physical location
016730  012737  010000  000000G                       MOV     #10000,@#UPAR7  ;Map to 256K byte boundary
016736  052737  000000G 000000G                       BIS     #UMODE,@#PSW    ;Go into user mode
016744  012737  177777  160110                        MOV     #-1,@#160000+TSTWRD     ;Store -1 at 256K physical location
016752  000004                                        IOT                     ;Return to kernel mode
016754  005737  000110                                TST     @#TSTWRD        ;Test physical location
016760  001402                                        BEQ     7$              ;Br if physical location is clear
016762  012705  010000                                MOV     #10000,R5       ;Constrain memory to 256K byte total
                                              .ENDC   ;NE,<PROASM-1>
                                              ;
                                              ;  Reached end of available memory.
                                              ;
016766  010537  000000G                       7$:     MOV     R5,PHYMEM       ;set physical memory size
016772  020537  000000G                               CMP     R5,MAPSIZ       ;Constrain kernel to user specified cutoff
016776  101402                                        BLOS    8$              ;Br if below user specified
017000  013705  000000G                               MOV     MAPSIZ,R5       ;Only use this much memory
017004  010537  000000G                       8$:     MOV     R5,$MEMSZ       ;Set # 64-byte blocks of total memory
017010  010537  000134'                               MOV     R5,FMEMHI       ;Save base 64-byte block # of top of free mem
                                              ;
                                              ;  Turn off memory management
```

```
                                   ;
017014  105737   000000G                    TSTB     SR3FLG          ;Do we have memory management reg # 3?
017020  001403                              BEQ      9$              ;Br if non-existent
017022  042737   000000G  000000G           BIC      #EMMAP,@#SR3MMR ;Disable extended memory management
017030  042737   000000G  000000G  9$:      BIC      #MMENBL,@#SROMMR;Turn off memory management
                                   ;
                                   ;   If this is a Q-bus machine with >256Kb then set EXTLSI flag in ICONFG
                                   ;
017036  023727   000134'  010000            CMP      FMEMHI,#4096.   ;Does machine have at least 256Kb?
017044  103411                              BLO      25$             ;Br if not
017046  105237   000000G                    INCB     MEM256          ;Remember machine has at least 256kb
017052  123727   000000G  000000G           CMPB     VBUSTP,#QBUS    ;Is this a Q-bus machine?
017060  001003                              BNE      25$             ;Br if not
017062  052737   000001   000242'           BIS      #EXTLSI,ICONFG  ;Set extended-LSI flag in ICONFG
                                   ;
                                   ;   See if this machine needs UNIBUS mapping
                                   ;
017070  123727   000000G  000000G  25$:     CMPB     VBUSTP,#UNIBUS  ;Is this a UNIBUS machine?
017076  001005                              BNE      29$             ;Br if not
017100  105737   000000G                    TSTB     MEM256          ;Does machine have at least 256kb of memory?
017104  001402                              BEQ      29$             ;Br if not
017106  105237   000000G                    INCB     UBUSMP          ;Say UNIBUS mapping is needed
                                   ;
                                   ;   Finished
                                   ;
017112  012637   000004            29$:     MOV      (SP)+,@#4       ;Reset trap vector
017116  012605                              MOV      (SP)+,R5
017120  012604                              MOV      (SP)+,R4
017122  012603                              MOV      (SP)+,R3
017124  012602                              MOV      (SP)+,R2
017126  012601                              MOV      (SP)+,R1
017130  000207                              RETURN
                                   ;
                                   ;   Trap - return with C-bit set.
                                   ;
017132  052766   000001   000002   TRCSET:  BIS      #1,2(SP)        ;Set c-bit for return
017140  000002                              RTI                      ;Return from trap
                                   ;
                                   ;   IOT - return at kernel mode with c-bit preserved.
                                   ;
017142  042766   000000G  000002   RTNKM:   BIC      #UMODE,2(SP)    ;Clear user mode - return to kernel
017150  000002                              RTI                      ;Return from trap
                                   ;
                                   ;   Error: System does not have memory management hardware.
                                   ;
017152                             NOXM:    .PRINT   #TSXHD          ;PRINT ERROR MESSAGE
017160                                      .PRINT   #NXMMSG
017166  000137   004102'                    JMP      INISTP          ;ABORT INITIALIZATION
```

```
                                          .SBTTL  CXTALC -- Set up info about job context area
                                  ;-----------------------------------------------------------------------
                                  ;  Set up information about the size of the job context area.
                                  ;
                                  ;  Outputs:
                                  ;   CXTWDS = Number of words needed for job context area.
                                  ;   CXTPAG = Number of 512-byte pages needed for context area.
                                  ;   CXTPDR = Value to load into PDR when mapping job context area.
                                  ;   CXTRMN = Address of simulated RMON in context area.
                                  ;   RMNPDR = Value to load into PDR to map to simulated RMON.
                                  ;
 017172                           CXTALC:
                                  ;
                                  ;  Get size of base portion of job context area
                                  ;
 017172  012700  000000G                  MOV     #CXTSIZ,RO        ;Get # bytes for base context area
                                  ;
                                  ;  Bound up to 64-byte boundary and add size of simulated RMON
                                  ;  which is allocated above the base job context data.
                                  ;
 017176  062700  000077                   ADD     #63.,RO           ;Bound up to 64 byte boundary
 017202  042700  000077                   BIC     #77,RO
 017206  010037  000000G                  MOV     RO,CXTRMN         ;Offset to start of simulated RMON
 017212  062737  000000G 000000G          ADD     #CXTBAS,CXTRMN    ;Add base virtual address of context area
 017220  062700  000001G                  ADD     #MVSIZ+1,RO       ;Add space for simulated RMON & channels
                                  ;
                                  ;  Save number of words needed for context area
                                  ;
 017224  006200                           ASR     RO                ;Convert to # words
 017226  010037  000000G                  MOV     RO,CXTWDS         ;This is # words for whole job context area
                                  ;
                                  ;  Compute PDR value to use to map to job context area
                                  ;
 017232  062700  000037                   ADD     #31.,RO           ;Bound up to # 32 word units
 017236  072027  177773                   ASH     #-5.,RO           ;Get # 32-word units for context area
 017242  000300                           SWAB    RO                ;Put # 32-word units in high-order byte
 017244  052700  000006                   BIS     #6,RO             ;Set PDR control flags
 017250  010037  000000G                  MOV     RO,CXTPDR         ;This is the PDR value
                                  ;
                                  ;  Compute # 512-byte pages needed for job context block
                                  ;
 017254  013700  000000G                  MOV     CXTWDS,RO         ;Get back # words for context area
 017260  062700  000377                   ADD     #255.,RO          ;Bound up to # 256-word blocks
 017264  072027  177770                   ASH     #-8.,RO           ;Get # 256-word pages for context area
 017270  010037  000000G                  MOV     RO,CXTPAG         ;# pages for job context area
                                  ;
                                  ;  Set up PDR value used when mapping to simulated RMON
                                  ;
 017274  012700  000000G                  MOV     #MVSIZ,RO         ;Get size of monitor vector table
 017300  062700  000077                   ADD     #63.,RO           ;Round up to # 32 word blocks
 017304  072027  177772                   ASH     #-6,RO            ;Cvt to # 32-word blocks
 017310  005300                           DEC     RO                ;Get # blocks - 1
 017312  000300                           SWAB    RO                ;Put # blocks in left byte
 017314  052700  000006                   BIS     #6,RO             ;Allow read and write access
 017320  042700  100261                   BIC     #100261,RO        ;Make sure unused PDR bits are zero
 017324  010037  000000G                  MOV     RO,RMNPDR         ;This is PDR value to map to sim. RMON
                                  ;
```

```
                                        ;  Finished
                                        ;
   017330  000207                               RETURN
```

```
                                                 .SBTTL  MAPALC -- Allocate memory usage table
                                         ;-----------------------------------------------------------------------
                                         ;  MAPALC is called to allocate a table that keeps track of which pages
                                         ;  of memory are currently in use by user jobs and which are free.
                                         ;  Each byte in the table corresponds to a 512-byte block of physical memory.
                                         ;  The portion of physical memory used by the system is not represented
                                         ;  in the memory allocation table.
                                         ;
                                         ;  Inputs:
                                         ;   R5     = 64-byte block number of top of free memory area.
                                         ;   FMEMLO = 64-byte block number of base of free memory area.
                                         ;
                                         ;  Outputs:
                                         ;   FMEMHI = 64-byte block number of top of free memory area.
                                         ;   MAPPAR = 64-byte block number used to map to the memory alloc table.
                                         ;   BASMAP = Virtual address of memory allocation table that would
                                         ;            correspond to physical address 0.  Note, the entries
                                         ;            in the allocation table between BASMAP and LOMAP are
                                         ;            actually not allocated.
                                         ;   LOMAP  = Virtual address of memory allocation table that corresponds
                                         ;            to 1st physical 512-byte page that is available to user jobs.
                                         ;            Note, LOMAP always contains 120000 because we access the
                                         ;            allocation table by mapping it through PAR 5.
                                         ;   HIMAP  = Virtual address of memory allocation table that corresponds
                                         ;            to 512-byte page above the top of the user area.
                                         ;
 017332  010246                          MAPALC: MOV     R2,-(SP)
 017334  010346                                  MOV     R3,-(SP)
 017336  010446                                  MOV     R4,-(SP)
                                         ;
                                         ;  Determine how many bytes will be required for the memory allocation table.
                                         ;  One byte in the table is required for each 512-byte physical page.
                                         ;
 017340  010503                                  MOV     R5,R3           ;Get 64-byte block # of top of free mem
 017342  072327  177775                          ASH     #-3,R3          ;Convert to 512-byte page #
 017346  042703  160000                          BIC     #160000,R3      ;Kill possible sign extension
 017352  013702  000136'                         MOV     FMEMLO,R2       ;Get 64-byte block # of base of free memory
 017356  062702  000007                          ADD     #7,R2           ;Round up
 017362  072227  177775                          ASH     #-3,R2          ;Convert to 512-byte page #
 017366  042702  160000                          BIC     #160000,R2      ;Kill possible sign extension
 017372  160203                                  SUB     R2,R3           ;Get # bytes needed for allocation table
 017374  003440                                  BLE     10$             ;Br if memory overflow
 017376  010304                                  MOV     R3,R4           ;Get # bytes for allocation table
 017400  062704  001000                          ADD     #512.,R4        ;Add 1 extra byte and round up to 512-byte
 017404  072427  177767                          ASH     #-9.,R4         ;Get # 512-byte units needed for alloc table
                                         ;
                                         ;  Set up virtual address pointers for the allocation table
                                         ;
 017410  012700  000000G                         MOV     #VPAR5,R0       ;We will map to alloc table through PAR 5
 017414  010037  000000G                         MOV     R0,LOMAP        ;Pointer to 1st entry in alloc table
 017420  160200                                  SUB     R2,R0           ;Get pseudo virtual address for page # 0
 017422  010037  000000G                         MOV     R0,BASMAP       ;This would point to alloc entry for page 0
 017426  012700  000000G                         MOV     #VPAR5,R0       ;Get back base address of table
 017432  060300                                  ADD     R3,R0           ;Add # bytes used by table
 017434  160400                                  SUB     R4,R0           ;Subtract space used by table itself
 017436  010037  000000G                         MOV     R0,HIMAP        ;Virtual address of 1st entry for system page
                                         ;
```

```
                                     ;  Allocate space for the allocation table
                                     ;
   017442  072427  000003                      ASH     #3,R4            ;Get # 64-byte units for alloc table
   017446  160405                              SUB     R4,R5            ;Compute physical 64-byte base for table
   017450  020537  000136'                     CMP     R5,FMEMLO        ;Did we run out of memory space?
   017454  101410                              BLOS    10$              ;Br if memory overflow
   017456  010537  000000G                     MOV     R5,MAPPAR        ;Use this value to map PAR 5 to alloc table
   017462  010537  000134'                     MOV     R5,FMEMHI        ;Save new top of free memory area
                                     ;
                                     ;  Finished
                                     ;
   017466  012604                              MOV     (SP)+,R4
   017470  012603                              MOV     (SP)+,R3
   017472  012602                              MOV     (SP)+,R2
   017474  000207                              RETURN
                                     ;
                                     ;  Error: Generated system is too large
                                     ;
   017476                            10$:      .PRINT  #TSXHD           ;Print error message heading
   017504                                      .PRINT  #PHSOVF          ;Physical memory overflow
   017512  000137  004102'                     JMP     INISTP           ;Abort the initialization
```

```
                                                 .SBTTL  SETJSZ -- Set up information about maximum job sizes
                                         ;-----------------------------------------------------------------------
                                         ;  SETJSZ is called to set up some information about the maximum
                                         ;  job sizes to be allowed. The maximum job size is chosen so that
                                         ;  we are guaranteed to be able to get at least one job logged on.
                                         ;
                                         ;  Inputs:
                                         ;   LOMAP = Address of 1st MEMMAP entry available to user jobs.
                                         ;   HIMAP = Address of 1st MEMMAP entry above top of user job area.
                                         ;
                                         ;  Outputs:
                                         ;   FREPGS = Total number of 512-byte pages available to user jobs.
                                         ;   MXJMEM = max # K bytes available to a job
                                         ;   DFJMEM = Default job memory size (kb)
                                         ;
017516  010546                           SETJSZ: MOV     R5,-(SP)
                                         ;
                                         ;  Determine total number of pages of memory available to user jobs
                                         ;
017520  013705  000000G                          MOV     HIMAP,R5        ;POINTER ABOVE LAST FREE PAGE ENTRY
017524  163705  000000G                          SUB     LOMAP,R5        ;GET TOTAL # OF FREE PAGES
017530  010537  000000G                          MOV     R5,FREPGS       ;# FREE PAGES AVAILABLE TO USER JOBS
                                         ;
                                         ;  Make sure there is enough free space to run TSKMON.
                                         ;
017534  020537  000000G                          CMP     R5,KMNPGS       ;COMPARE # FREE PAGES TO # PAGES FOR TSKMON
017540  103436                                   BLO     1$              ;BR IF INSUFFICIENT MEMORY TO RUN TSKMON
                                         ;
                                         ;  Set up max memory limit for jobs
                                         ;
017542  013700  000000G                          MOV     CXTPAG,R0       ;# PAGES NEEDED FOR JOB CONTEXT AREA
017546  010037  000000G                          MOV     R0,JCXPGS
017552  160005                                   SUB     R0,R5           ;LEAVE ROOM FOR JOB CONTEXT AREA
017554  006205                                   ASR     R5              ;Convert # pages to # KB
017556  020537  000000G                          CMP     R5,VHIMEM       ;COMPARE WITH TSGEN SPECIFIED MAX SIZE
017562  101402                                   BLOS    10$             ;BR IF CONSTRAINED BY PHYSICAL SIZE
017564  013705  000000G                          MOV     VHIMEM,R5       ;LIMIT BY VALUE SPECIFIED IN TSGEN
017570  010537  000000G                  10$:    MOV     R5,MXJMEM       ;MAX # K BYTES AVAILABLE TO A JOB
017574  010500                                   MOV     R5,R0
017576  072027  000012                           ASH     #10.,R0         ;CONVERT # KB TO BYTE ADDRESS
017602  001002                                   BNE     2$              ;BR IF DIDN'T OVERFLOW 64KB
017604  012700  177774                           MOV     #177774,R0      ;GET 64KB TOP ADDRESS
017610  010037  000000G                  2$:     MOV     R0,MXJADR       ;ADDRESS ABOVE TOP OF JOB
                                         ;
                                         ;  Set default memory size of jobs
                                         ;
017614  020537  000000G                          CMP     R5,VDFMEM       ;COMPARE TO DEFAULT SPECIFIED IN TSGEN
017620  101402                                   BLOS    11$             ;BR IF CONSTRAINED BY PHYSICAL LIMIT
017622  013705  000000G                          MOV     VDFMEM,R5       ;CONSTRAIN BY TSGEN PARAMETER
017626  010537  000000G                  11$:    MOV     R5,DFJMEM       ;SET DEFAULT JOB MEMORY SIZE
                                         ;
                                         ;  Finished
                                         ;
017632  012605                                   MOV     (SP)+,R5
017634  000207                                   RETURN
                                         ;
                                         ;  Error -- Insufficient memory space available to run TSKMON.
```

```
     58                                 ;
     59 017636  004737  027650'         1$:     CALL    SIZERR          ;Generated system is too big -- abort
```

```
                                        .IF     NE,<PROASM-1>   ;No parity control if PRO only
                                        .SBTTL  PARSET -- Setup memory parity control
                                ;--------------------------------------------------------------------------
                                ;  PARSET is called to set up memory parity control.
                                ;  Currently this consists of disabling memory parity.
                                ;
017642  005727  000000G         PARSET: TST     #MPARFL         ;Does he want to disable memory parity?
017646  001027                          BNE     20$             ;Br if not
017650  010246                          MOV     R2,-(SP)
017652  013746  000004                  MOV     @#4,-(SP)       ;Save contents of trap vector
                                ;
                                ;  Catch traps that occur when we access unimplemented parity registers
                                ;
017656  012737  017702' 000004          MOV     #2$,@#4         ;Send traps to 2$
017664  000240                          NOP                     ;Clean out instruction pipeline
017666  000240                          NOP
                                ;
                                ;  Disable parity for each block of memory
                                ;
017670  012702  000000G                 MOV     #MPAR0,R2       ;Point to 1st memory control register
017674  042712  000000G         1$:     BIC     #PARENL,(R2)    ;Disable memory parity
017700  000402                          BR      3$              ;We did not trap
017702  062706  000004          2$:     ADD     #4,SP           ;Clean trap PS and PC off of stack
017706  062702  000002          3$:     ADD     #2,R2           ;Point to next parity control register
017712  020227  000000G                 CMP     R2,#MPAR16      ;Have we cleared all registers?
017716  101766                          BLOS    1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
017720  012637  000004                  MOV     (SP)+,@#4       ;Restore trap vector
017724  012602                          MOV     (SP)+,R2
017726  000207                  20$:    RETURN
                                        .IFF    ;NE,<PROASM-1>
                                PARSET: RETURN
                                        .ENDC   ;NE,<PROASM-1>
                                        .SBTTL  GETHNL -- Load device handlers into memory
                                ;-----------------------------------------------------------------------
                                ;  GETHNL performs two functions:
                                ;   1. Set up information in the device tables about all devices.
                                ;   2. Load those handlers that reside in low memory.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of start of free memory.
                                ;
                                ;  Outputs:
                                ;   R5 = Address of new start of free memory.
                                ;   NMXHAN = Number of handlers to load into extended memory.
                                ;
017730  010146                  GETHNL: MOV     R1,-(SP)
017732  010246                          MOV     R2,-(SP)
017734  010446                          MOV     R4,-(SP)
                                ;
                                ;  Begin loop to check all handlers specified in TSGEN with DEVDEF.
                                ;
017736  005001                          CLR     R1                      ;Init device table index
017740  020127  000000C         1$:     CMP     R1,#<AHEND-AUTHAN>      ;Done all devices?
017744  103015                          BHIS    2$                      ;Br if yes
017746  016102  000000G                 MOV     AUTHAN(R1),R2           ;Get the name of the device
017752  001407                          BEQ     3$                      ;Ignore null devices
017754  020227  000000G                 CMP     R2,#DMYDEV              ;Is this a dummy device entry?
017760  001404                          BEQ     3$                      ;Skip it if yes
017762  016104  000000G                 MOV     DTYPE(R1),R4            ;Get flags specified in TSGEN
                                ;
                                ;  Load this handler
                                ;
017766  004737  020040'                 CALL    LDHAND                  ;Try to load handler into memory
                                ;
                                ;  Check next device
                                ;
017772  062701  000002          3$:     ADD     #2,R1                   ;Advance device index
017776  000760                          BR      1$                      ;See if more devices to load
                                ;
                                ;  Now see if there are spooled devices to contend with
                                ;
020000  012704  000000G         2$:     MOV     #SPLND,R4               ;Are there any spooled devices?
020004  001411                          BEQ     9$                      ;Br if not
020006  012701  000000G                 MOV     #SPLDEV,R1              ;Point to spooled device name table
020012  012102                  5$:     MOV     (R1)+,R2                ;Get the name of the next spooled device
020014  010446                          MOV     R4,-(SP)                ;Save device count
020016  005004                          CLR     R4                      ;Say no TSGEN flags for device
020020  004737  020040'                 CALL    LDHAND                  ;Load the handler
020024  012604                          MOV     (SP)+,R4                ;Recover the device count
020026  077407                          SOB     R4,5$                   ;Loop if more handlers to load
                                ;
                                ;  Finished
                                ;
020030  012604                  9$:     MOV     (SP)+,R4
020032  012602                          MOV     (SP)+,R2
020034  012601                          MOV     (SP)+,R1
020036  000207                          RETURN
```

```
                                        .SBTTL  LDHAND -- Load a device handler
                                ;--------------------------------------------------------------------
                                ;  LDHAND sets up the device tables for a handler and loads into memory
                                ;  those handlers that reside in low memory.
                                ;  The device interrupt vectors are NOT set up by LDHAND.
                                ;
                                ;  Inputs:
                                ;   R2 = Rad-50 name of device.
                                ;   R4 = TSX-Plus DX$xxx status flags for device from TSGEN.
                                ;   R5 = Address where handler is to be loaded.
                                ;
                                ;  Outputs:
                                ;   R5 = New free memory address.
                                ;   NUMDEV = Incremented by 2.
                                ;   PNAME(i) = Rad-50 name of device.
                                ;   ENTRY(i) = Handler entry point.
                                ;   DVSTAT(i) = Device status flags.
                                ;   DVFLAG(i) = TSX-Plus device status flags.
                                ;   HANPAR(i) = PAR offset if this is a mapped handler.
                                ;
020040  010446                  LDHAND: MOV     R4,-(SP)
                                ;
                                ;  Determine if we should ignore this device
                                ;
020042  004737  020212'                 CALL    INSCK1          ;Should we ignore this device?
020046  103457                          BCS     9$              ;Br if yes
                                ;
                                ;  The initial tests indicate that this handler should be loaded.
                                ;  Now open the handler file and perform some additional checks.
                                ;
020050  004737  020370'                 CALL    INSCK2          ;Perform some additional checks on handler
020054  103451                          BCS     8$              ;Br if we should not load this device
                                ;
                                ;  At this point channel 1 is open to the handler file and block 0
                                ;  of the handler is in WRKBUF.
                                ;  Set up information tables for this device.
                                ;
020056  004737  021004'                 CALL    STDVTB          ;Set up info in tables for this device
                                ;
                                ;  Determine if this handler is to be loaded into low memory or
                                ;  extended memory.
                                ;
020062  016400  000000G                 MOV     DVFLAG(R4),R0   ;Get TSX-Plus status flags for device
020066  032700  000000G                 BIT     #DX$NHM,R0      ;Are we never to map this handler?
020072  001036                          BNE     1$              ;Br if handler cannot be mapped
020074  032700  000000G                 BIT     #DX$MPH,R0      ;Is mapping wanted for this handler?
020100  001433                          BEQ     1$              ;Br if not
020102  032700  000000G                 BIT     #DX$IBH,R0      ;Does this handler have an internal I/O buff?
020106  001412                          BEQ     2$              ;Br if not
020110  105737  000000G                 TSTB    UBUSMP          ;Does this machine have a mapped UNIBUS?
020114  001025                          BNE     1$              ;Br if yes -- Don't map this handler
020116  032700  000000G                 BIT     #DX$MAP,R0      ;Does this handler require I/O mapping?
020122  001404                          BEQ     2$              ;Br if not
020124  032737  000001  000242'         BIT     #EXTLSI,ICONFG  ;Is this a Q-bus system with more than 256Kb?
020132  001016                          BNE     1$              ;Br if yes -- Don't map this handler
                                ;
                                ;  This handler can be mapped and will be loaded in extended memory
```

```
                                        ;
   020134  005237  000124'         2$:     INC     NMXHAN          ;Count # of mapped handlers
   020140  012764  000001  000000G         MOV     #1,HANPAR(R4)   ;Set flag saying handler should be mapped
                                        ;
                                        ;  Make sure size of mapped handler does not exceed 8KB
                                        ;
   020146  026427  000000G 020000          CMP     HANSIZ(R4),#8192. ;Is mapped handler too big?
   020154  101411                          BLOS    8$              ;Br if not too big
   020156  012700  002335'                 MOV     #HN2BIG,R0      ;Get error message address
   020162  004737  022102'                 CALL    HLERR           ;Abort initialization if iniabt
   020166  000404                          BR      8$
                                        ;
                                        ;  This handler must be loaded into low memory
                                        ;
   020170  005064  000000G         1$:     CLR     HANPAR(R4)      ;Say this handler is not mapped
   020174  004737  021114'                 CALL    LDHNLO          ;Load handler into low memory
                                        ;
                                        ;  Close the handler file
                                        ;
   020200                          8$:     .CLOSE  #1              ;Close the handler file
                                        ;
                                        ;  Finished
                                        ;
   020206  012604                  9$:     MOV     (SP)+,R4
   020210  000207                          RETURN
```

```
                                        .SBTTL  INSCK1 -- Determine if a handler should be installed
                                ;----------------------------------------------------------------------------
                                ;   INSCK1 is called to determe if a certain device handler should be
                                ;   loaded.
                                ;
                                ;   Inputs:
                                ;    R2 = Rad50 name of the device.
                                ;    R4 = Initial DX$xxx flags as specified in TSGEN.
                                ;
                                ;   Outputs:
                                ;    C-flag cleared ==> Load the handler.
                                ;    C-flag set     ==> Do not load the handler.
                                ;    R2 = Device name with unit number removed.
                                ;    R4 = DX$xxx combined with default flags for the device.
                                ;
020212  010146                  INSCK1: MOV     R1,-(SP)
                                ;
                                ;   Strip off any specified unit number
                                ;
020214  010201                          MOV     R2,R1           ;Get full device name
020216  005000                          CLR     R0              ;Set for divide
020220  071027  000050                  DIV     #50,R0          ;Split off last digit
020224  070027  000050                  MUL     #50,R0          ;Now correct for divide
020230  010102                          MOV     R1,R2           ;Get device name less 3rd digit
020232  010237  000146'                 MOV     R2,CURNAM       ;Set name of handler being loaded
                                ;
                                ;   See if this is a device such as DK, SY, or TT which we don't
                                ;   need to load as a device handler.
                                ;
020236  020227  045640                  CMP     R2,#R50LD       ;Is device name LD?
020242  001004                          BNE     1$              ;Br if not
020244  105737  000000G                 TSTB    VLDSYS          ;Is standard system LD support included?
020250  001044                          BNE     5$              ;Br if yes -- Don't load LD
020252  000417                          BR      3$              ;Load LD
020254  012701  000160'         1$:     MOV     #SKPDEV,R1      ;Point to table of devices to skip
020260  020221                  2$:     CMP     R2,(R1)+        ;Is this a device to be skipped?
020262  001437                          BEQ     5$              ;Br if yes
020264  005711                          TST     (R1)            ;Reached end of skip list?
020266  001374                          BNE     2$              ;Loop if not
                                ;
                                ;   See if we have already loaded the handler for this device
                                ;
020270  013701  000000G                 MOV     NUMDEV,R1       ;Get index for last device
020274  001406                          BEQ     3$              ;Br if no devices installed yet
020276  020261  000000G         4$:     CMP     R2,PNAME(R1)    ;See if this device is already installed
020302  001427                          BEQ     5$              ;Br if already installed
020304  162701  000002                  SUB     #2,R1           ;More installed devices to check?
020310  003372                          BGT     4$              ;Loop if yes
                                ;
                                ;   This handler is to be loaded.
                                ;   Get default TSX-Plus control flags for this device.
                                ;
020312  012701  000336'         3$:     MOV     #DVFLBS,R1      ;Point to start of table
020316  020261  000000          6$:     CMP     R2,DV$NAM(R1)   ;Search for device in the table
020322  001003                          BNE     7$              ;Br if this is not it
020324  056104  000002                  BIS     DV$FLG(R1),R4   ;Combine default flags
020330  000405                          BR      8$
```

```
020332  062701  000004          7$:     ADD     #DV$$SZ,R1      ;Point to next entry
020336  020127  000516'                 CMP     R1,#DVFLND      ;Checked all entries?
020342  103765                          BLO     6$              ;Loop if not
                                ;
                                ;  If this is a DMA device, set flag saying buffers must be on
                                ;  even byte boundaries.
                                ;
020344  032704  000000G         8$:     BIT     #DX$DMA,R4      ;Is this a DMA device?
020350  001402                          BEQ     10$             ;Br if not
020352  052704  000000G                 BIS     #DX$EBA,R4      ;Set even-buffer-boundary flag
                                ;
                                ;  Load this handler
                                ;
020356  000241                  10$:    CLC                     ;Set flag saying to load the handler
020360  000401                          BR      9$
                                ;
                                ;  Do not load this handler
                                ;
020362  000261                  5$:     SEC                     ;Set flag saying not to load the handler
                                ;
                                ;  Finished
                                ;
020364  012601                  9$:     MOV     (SP)+,R1
020366  000207                          RETURN
```

```
                                        .SBTTL  INSCK2 -- Additional checking for handler installation
                                ;-----------------------------------------------------------------------
                                ;  INSCK2 is called to determine if a device handler should be installed.
                                ;
                                ;  Inputs:
                                ;   R2 = Rad50 device name (without unit number).
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Load this handler.
                                ;   C-flag set     ==> Do not load this handler.
                                ;   If the handler is to be loaded, its block 0 is in WRKBUF and channel
                                ;   number 1 is opened to the handler file.
                                ;
020370  010146                  INSCK2: MOV     R1,-(SP)
020372  010246                          MOV     R2,-(SP)
020374  010346                          MOV     R3,-(SP)
020376  010446                          MOV     R4,-(SP)
020400  010546                          MOV     R5,-(SP)
020402  013746  000004                  MOV     @#4,-(SP)       ;Save the bus timeout vector
020406  013746  000010                  MOV     @#10,-(SP)      ;Save illegal instruction vector
                                ;
                                ;  Try to lookup handler file on system disk
                                ;
020412  010237  000202'                 MOV     R2,HANNAM+2     ;Set the device name for the lookup
                                ;;; Don't change channel # without changing STDVTB
020416                                  .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
020436  103006                          BCC     1$              ;Br if we found the handler file
                                ;
                                ;  Error -- Cannot find handler file

020440  012700  001510'         2$:     MOV     #CFHMSG,R0      ;Can't find handler
020444  004737  022102'                 CALL    HLERR           ;See if should abort initialization
020450  000137  020760'                 JMP     13$
                                ;
                                ;  We were able to open the handler file.
                                ;  Read in block 0 of handler.
                                ;
020454                          1$:     .READW  #AREA,#1,WRKBUF,#256.,#0 ;Read block 0 into WRKBUF
020512  103006                          BCC     3$              ;Br if read ok
020514  012700  001552'                 MOV     #ERHMSG,R0      ;Error during read
020520  004737  022102'                 CALL    HLERR
020524  000137  020760'                 JMP     13$
                                ;
                                ;  Determine if the handler is supported under the current RT-11 version
                                ;
020530  012700  000322'         3$:     MOV     #HVTBL,R0       ;Point to table with handler version info
020534  013701  000152'                 MOV     WRKBUF,R1       ;Point to buffer with block 0 of handler
020540  116101  000000G                 MOVB    H.DSTS(R1),R1   ;Get device ID code from handler
020544  120160  000000          51$:    CMPB    R1,HV$ID(R0)    ;Compare handler ID code with table entry
020550  001011                          BNE     53$             ;Br if this entry not for this handler
020552  026037  000002  000156'         CMP     HV$VER(R0),RTVPTR ;Is handler valid for this version?
020560  101412                          BLOS    54$             ;Br if OK
020562  012700  001616'                 MOV     #ERHNDV,R0      ;Wrong version of RT for handler
020566  004737  022102'                 CALL    HLERR           ;See if should Report error and abort
020572  000472                          BR      13$
020574  062700  000004          53$:    ADD     #HV$$SZ,R0      ;Point to next handler version table entry
020600  020027  000336'                 CMP     R0,#HVEND       ;Are there more entries?
```

```
020604  103757                          BLO     51$               ;Loop if more to check
                                ;
                                ;  Check handler sysgen options
                                ;
020606  013700  000152'         54$:    MOV     WRKBUF,R0         ;Point to buffer with handler block 0
020612  032760  000000G 000000G         BIT     #SG$MMU,H.GEN(R0);Was handler genned with XM support?
020620  001005                          BNE     4$                ;Br if yes
020622  012700  002027'                 MOV     #HSGER,R0         ;Error if not XM version of handler
020626  004737  022102'                 CALL    HLERR
020632  000452                          BR      13$
020634  016037  000000G 000240' 4$:     MOV     H.GEN(R0),HGENFL;Save handler sysgen flags for later
                                ;
                                ;  Check the CSR address specified in the handler to see if the
                                ;  hardware device for this handler exists.
                                ;
020642  012737  017132' 000004          MOV     #TRCSET,@#4       ;Catch bus timeout traps
020650  012737  017132' 000010          MOV     #TRCSET,@#10      ;Catch illegal instruction traps
020656  013700  000152'                 MOV     WRKBUF,R0         ;Point to start of block 0 of handler
020662  016001  000000G                 MOV     H.CSR(R0),R1      ;Get address of CSR for device
020666  001407                          BEQ     5$                ;Br if no CSR specified
020670  005711                          TST     (R1)              ;Is CSR accessible?
020672  103005                          BCC     5$                ;Br if ok
020674  012700  001670'                 MOV     #NOCSRR,R0        ;Trap occurred while accessing CSR
020700  004737  022102'                 CALL    HLERR             ;See if should report error and abort
020704  000425                          BR      13$
                                ;
                                ;  Execute the device installation code.
                                ;  The installation code will set the C-flag if the handler should
                                ;  not be loaded.
                                ;
020706  062700  000000G         5$:     ADD     #H.INS,R0         ;Offset 200 in block 0
020712  005710                          TST     @R0               ;Does any installation code exist?
020714  001420                          BEQ     11$               ;Br if no driver installation code
020716  013746  000000G                 MOV     @#RMON,-(SP)      ;Save RT-11 RMON pointer
020722  012737  000000G 000000G         MOV     #MONVEC,@#RMON    ;Set TSX-Plus RMON pointer
020730  013703  000000G                 MOV     RPRVEC,R3         ;Get pointer to Pro vec addr routine
020734  004710                          CALL    @R0               ;Call the installation code
020736  012637  000000G                 MOV     (SP)+,@#RMON      ;Restore RT-11 RMON pointer
020742  103006                          BCC     13$               ;C-flag now indicates handler load status
020744  012700  001721'                 MOV     #ERHINS,R0        ;Error occured in handler installation code
020750  004737  022102'                 CALL    HLERR
020754  000401                          BR      13$
                                ;
                                ;  Finished with installation verification.
                                ;
020756  000241                  11$:    CLC                       ;Clear the c-bit for driver installation
020760  012637  000010          13$:    MOV     (SP)+,@#10        ;Restore illegal instruction vector
020764  012637  000004                  MOV     (SP)+,@#4         ;Restore the bus timeout vector
020770  012605                          MOV     (SP)+,R5
020772  012604                          MOV     (SP)+,R4
020774  012603                          MOV     (SP)+,R3
020776  012602                          MOV     (SP)+,R2
021000  012601                          MOV     (SP)+,R1
021002  000207                          RETURN
```

```
                                                .SBTTL  STDVTB -- Set up device table entries for a device
                                        ;-----------------------------------------------------------------------
                                        ;  STDVTB is called to set up device table entries for a device whose
                                        ;  handler is being loaded.
                                        ;
                                        ;  Inputs:
                                        ;   R2 = Rad50 name of device (less unit number).
                                        ;   R4 = DX$xxx device flags for DVFLAG table.
                                        ;   Block 0 of the handler must be in WRKBUF.
                                        ;   Channel 1 must be open to the handler file.
                                        ;
                                        ;  Outputs:
                                        ;   R4 = Device table index number for this device.
                                        ;   NUMDEV = Incremented by 2.
                                        ;   PNAME(i) = Rad50 name of the device.
                                        ;   DVSTAT(i) = Device status flags.
                                        ;   DVFLAG(i) = TSX-Plus control flags.
                                        ;   HANSIZ(i) = Size of handler (bytes).
                                        ;   DEVSIZ(i) = Size of device (blocks).
                                        ;
 021004                                 STDVTB:
                                        ;
                                        ;  Increment device counter
                                        ;
 021004  062737  000002  000000G                ADD     #2,NUMDEV       ;Say another device added to tables
 021012  013700  000000G                        MOV     NUMDEV,R0       ;Get device index number
                                        ;
                                        ;  Set up PNAME and DVFLAG.
                                        ;
 021016  010260  000000G                        MOV     R2,PNAME(R0)    ;Set permanent device name
 021022  010460  000000G                        MOV     R4,DVFLAG(R0)   ;Set up TSX-Plus control flags for the device
                                        ;
                                        ;  Set HANDSK entry to 1.
                                        ;  This entry is supposed to hold the absolute block number on the disk where
                                        ;  block 1 of the handler is located.  We set to 1 because all IO we do
                                        ;  on behalf of the handler is relative to the base of the handler rather than
                                        ;  relative to the start of the disk.  This is critical during handler
                                        ;  load/fetch code.
                                        ;  NOTE:: This table is replaced during KMINIT by the location of handler
                                        ;  block 1 relative to the start of the disk so that utilities can find
                                        ;  the handler files in the same way as under RT-11.
                                        ;
 021026  012760  000001  000000G                MOV     #1,HANDSK(R0)   ;Set block 1 relative offset of handler file
                                        ;
                                        ;  Extract parameters from handler block 0
                                        ;
 021034  010004                                 MOV     R0,R4           ;Carry device index in R4
 021036  013700  000152'                        MOV     WRKBUF,R0       ;Point to block 0 of handler
 021042  016064  000000G 000000G                MOV     H.SIZ(R0),HANSIZ(R4) ;Set handler size
 021050  016064  000000G 000000G                MOV     H.DVSZ(R0),DEVSIZ(R4) ;Number of blocks on device
 021056  016064  000000G 000000G                MOV     H.DSTS(R0),DVSTAT(R4) ;Set device status flags
                                        ;
                                        ;  Disable MOUNTs and data caching for certain devices
                                        ;
 021064  032764  000000G 000000G                BIT     #DS$DIR,DVSTAT(R4) ;Is this a directory structured device?
 021072  001404                                 BEQ     1$              ;Br if not -- No mounts allowed
 021074  032764  000000G 000000G                BIT     #DS$NRD,DVSTAT(R4) ;Non RT-11 directory structure (mag tape)?
```

```
     58 021102  001403                           BEQ     9$              ;Br if not
     59 021104  052764  000000C 000000G 1$:      BIS     #<DX$NMT!DX$NCA>,DVFLAG(R4) ;Disable mounts and data caching
     60                                 ;
     61                                 ;  Finished
     62                                 ;
     63 021112  000207                  9$:      RETURN
```

```
                                                 .SBTTL  LDHNLO -- Load device handler into low memory
                                         ;-----------------------------------------------------------------------
                                         ;  LDHNLO is called to load a device handler into low memory.
                                         ;
                                         ;  Inputs:
                                         ;   R4 = Device index number.
                                         ;   R5 = Address of start of free memory area.
                                         ;
                                         ;  Outputs:
                                         ;   R5 = Address of new start of free memory area.
                                         ;
 021114  010346                          LDHNLO: MOV     R3,-(SP)
                                         ;
                                         ;  Determine if we have enough free memory space to read the handler
                                         ;
 021116  005064  000000G                         CLR     HANPAR(R4)       ;Say this handler is not mapped
 021122  010500                                  MOV     R5,R0            ;Get current top of memory address
 021124  016403  000000G                         MOV     HANSIZ(R4),R3    ;Get size of handler
 021130  060300                                  ADD     R3,R0            ;Get address above top of handler
 021132  004737  027630'                         CALL    CHKMEM           ;See if handler will fit in memory
                                         ;
                                         ;  Handler will fit. Read it into memory.
                                         ;
 021136  006203                                  ASR     R3               ;Get number of words to read
 021140                                          .READW  #AREA,#1,R5,R3,#1
 021174  103005                                  BCC     1$               ;Br if read ok
 021176  012700  001552'                         MOV     #ERHMSG,R0       ;Error reading handler
 021202  004737  022102'                         CALL    HLERR            ;See if should Abort initialization
 021206  000414                                  BR      2$
                                         ;
                                         ;  Set address of handler entry point and compute address beyond
                                         ;  end of the handler.
                                         ;
 021210  010564  000000G                 1$:     MOV     R5,HANENT(R4)    ;Set address of handler entry point
 021214  062764  000006  000000G                 ADD     #6,HANENT(R4)    ;(Point to fourth word of handler)
 021222  006303                                  ASL     R3               ;Convert handler size to bytes
 021224  060305                                  ADD     R3,R5            ;Point beyond end of handler
                                         ;
                                         ;  Set up table of addresses of support routines at end of handler.
                                         ;
 021226  010503                                  MOV     R5,R3            ;Get address past end of handler
 021230  004737  022010'                         CALL    STHNPV           ;Set up pointer vector in handler
                                         ;
                                         ;  If handler has any load-time exectuion code, run it now
                                         ;
 021234  004737  022152'                         CALL    DOHNLC           ;Run any load-time code for handler
                                         ;
                                         ;  Finished
                                         ;
 021240  012603                          2$:     MOV     (SP)+,R3
 021242  000207                                  RETURN
```

```
                                                 .SBTTL  GETHNH -- Load handlers into extended memory
                                         ;-----------------------------------------------------------------------
                                         ;  GETHNH is called to load those handlers that can be placed in extended
                                         ;  memory.  The status tables for these devices have already been set up
                                         ;  by GETHNL.
                                         ;
                                         ;  Inputs:
                                         ;   R5 = 64-byte block number of top of free memory area.
                                         ;
                                         ;  Outputs:
                                         ;   R5 = 64-byte block number of new top of free memory area.
                                         ;
021244  010446                           GETHNH: MOV     R4,-(SP)
                                         ;
                                         ;  Begin looking for handlers that are to be loaded into extended memory.
                                         ;  GETHNL stored a non-zero (but meaningless) value in the HANPAR entry
                                         ;  for each handler that is to be mapped.
                                         ;
021246  012704  000002                           MOV     #2,R4           ;Get index for first device entry
                                         ;
                                         ;  See if this device has a mapped handler
                                         ;
021252  005764  000000G                  1$:     TST     HANPAR(R4)      ;Is this handler mapped?
021256  001402                                   BEQ     2$              ;Br if not
                                         ;
                                         ;  We found an entry for a device with a mapped handler.
                                         ;  Load the handler.
                                         ;
021260  004737  021302'                          CALL    LDHNHI          ;Load a mapped handler
                                         ;
                                         ;  Look for more mapped handlers
                                         ;
021264  062704  000002                   2$:     ADD     #2,R4           ;Increment device index
021270  020437  000000G                          CMP     R4,NUMDEV       ;Checked all devices?
021274  101766                                   BLOS    1$              ;Loop if not
                                         ;
                                         ;  Finished
                                         ;
021276  012604                                   MOV     (SP)+,R4
021300  000207                                   RETURN
```

```
                                        .SBTTL  LDHNHI -- Load device handler into extended memory
                                ;-----------------------------------------------------------------------
                                ;  LDHNHI is called to load a device handler into extended memory.
                                ;
                                ;  Inputs:
                                ;   R4 = Device index number.
                                ;   R5 = 64-byte block number of top of free memory area.
                                ;
                                ;  Outputs:
                                ;   R5 = 64-byte block number of new top of free memory area.
                                ;
021302  010146                  LDHNHI: MOV     R1,-(SP)
021304  010246                          MOV     R2,-(SP)
021306  010346                          MOV     R3,-(SP)
021310  010446                          MOV     R4,-(SP)
                                ;
                                ;  Open channel 1 to the handler file.
                                ;
021312  016437  000000G 000202'         MOV     PNAME(R4),HANNAM+2 ;Set the device name for the lookup
021320  016437  000000G 000146'         MOV     PNAME(R4),CURNAM;Set name in case we have an error
021326                                  .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
021346  103006                          BCC     8$              ;Br if we found the handler file
021350  012700  001510'                 MOV     #CFHMSG,R0      ;Can't find handler
021354  004737  022102'                 CALL    HLERR           ;See if should Abort initialization
021360  000137  021770'                 JMP     9$
                                ;
                                ;  Read block 0 of the handler file and extract some information
                                ;
021364  013702  000152'         8$:     MOV     WRKBUF,R2       ;Get address of work buffer
021370                                  .READW  #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
021424  016237  000000G 000240'         MOV     H.GEN(R2),HGENFL;Save handler sysgen flags
                                ;
                                ;  Set virtual address of handler entry point
                                ;
021432  012764  000006G 000000G         MOV     #VPAR5+6,HANENT(R4) ;Set virtual addr of handler entry point
                                ;
                                ;  Get information about the size of the handler and determine the
                                ;  address in extended memory where the handler is to be loaded.
                                ;
021440  016402  000000G                 MOV     HANSIZ(R4),R2   ;Get size of handler (bytes)
021444  005202                          INC     R2              ;Make sure handler size is even
021446  042702  000001                  BIC     #1,R2
021452  010200                          MOV     R2,R0
021454  062700  000077                  ADD     #63.,R0         ;Round up to # 64-byte blocks
021460  072027  177772                  ASH     #-6,R0          ;Get # 64-byte blocks for handler
021464  060037  000000G                 ADD     R0,MHNSIZ       ;Accumulate total space for mapped handlers
021470  160005                          SUB     R0,R5           ;Reserve room for handler
021472  010564  000000G                 MOV     R5,HANPAR(R4)   ;Set mapping value for handler
021476  010537  000126'                 MOV     R5,HMAP         ;Set initial PAR base for handler
021502  012737  000001  000142'         MOV     #1,FILBLK       ;Set # of block to read from file
                                ;
                                ;  Begin loop to read handler into memory
                                ;
021510  010203                  1$:     MOV     R2,R3           ;Get remaining size of handler
021512  020327  001000                  CMP     R3,#512.        ;Compare with max we can read at one time
021516  101402                          BLOS    2$              ;Br if we can read remainder of handler
021520  012703  001000                  MOV     #512.,R3        ;Read one block
```

```
021524  160302                          2$:     SUB     R3,R2           ;Reduce amt of handler left to read
                                        ;
                                        ;  Read next block of handler
                                        ;
021526  006203                                  ASR     R3              ;Get # words to read
021530  013701  000152'                         MOV     WRKBUF,R1       ;Get address of buffer for read
021534                                          .READW  #AREA,#1,R1,R3,FILBLK ;Read a block
021570  103005                                  BCC     3$              ;Br if read ok
021572  012700  001552'                         MOV     #ERHMSG,R0      ;Get error message
021576  004737  022102'                         CALL    HLERR           ;See if should Abort initialization
021602  000472                                  BR      9$
                                        ;
                                        ;  Move the code we just read into the XM area for the handler
                                        ;
021604  012700  000000G                 3$:     MOV     #VPAR5,R0       ;Get virtual address of mapped region
021610                                          DISABL                  ;** Disable interrupts **
021616  013746  000000G                         MOV     @#KPAR5,-(SP)   ;;;Save current mapping of PAR 5
021622  013737  000126' 000000G                 MOV     HMAP,@#KPAR5    ;;;Set up mapping to get to XM area
021630  052737  000000G 000000G                 BIS     #MMENBL,@#SR0MMR;;;Enable memory management
021636  105737  000000G                         TSTB    MEM256          ;;;Does machine have > 256KB?
021642  001403                                  BEQ     4$              ;;;Br if not
021644  052737  000000G 000000G                 BIS     #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
021652  012120                          4$:     MOV     (R1)+,(R0)+     ;;;Move from WRKBUF to XM region
021654  077302                                  SOB     R3,4$           ;;;Loop till all moved
021656  105737  000000G                         TSTB    MEM256          ;;;Does machine have > 256KB?
021662  001403                                  BEQ     5$              ;;;Br if not
021664  042737  000000G 000000G                 BIC     #EMMAP,@#SR3MMR ;;;Disable extended memory addressing
021672  042737  000000G 000000G 5$:     BIC     #MMENBL,@#SR0MMR;;;Enable memory management
021700  012637  000000G                         MOV     (SP)+,@#KPAR5   ;;;Replace PAR 5 mapping
021704                                          ENABL                   ; ** Enable interrupts **
                                        ;
                                        ;  See if there is more to read
                                        ;
021712  062737  000010  000126'                 ADD     #8.,HMAP        ;Increase XM region base
021720  005237  000142'                         INC     FILBLK          ;Increment file block number
021724  005702                                  TST     R2              ;Is there more to read?
021726  001270                                  BNE     1$              ;Loop if more to read
                                        ;
                                        ;  We have finished moving the handler into its XM region.
                                        ;  Set up addresses of system routines in a vector at the end of the handler
                                        ;
021730  012703  000000G                         MOV     #VPAR5,R3       ;Get virtual address of handler base
021734  066403  000000G                         ADD     HANSIZ(R4),R3   ;Get virtual address beyond end of handler
021740  004737  022652'                         CALL    HANMAP          ;;;Map KPAR5 to the handler
021744  004737  022010'                         CALL    STHNPV          ;;;Set up handler pointer vector
021750  004737  022726'                         CALL    HANUMP          ;Restore mapping
                                        ;
                                        ;  If handler has any load-time code, run it now
                                        ;
021754  010537  000134'                         MOV     R5,FMEMHI       ;Set addr of top of free memory area
021760  004737  022152'                         CALL    DOHNLC          ;Run any load-time code for handler
021764  013705  000134'                         MOV     FMEMHI,R5       ;Get new top of free memory address
                                        ;
                                        ;  Close the handler file
                                        ;
021770                                  9$:     .CLOSE  #1              ;Close the handler file
                                        ;
```

```
                                   ;   Finished
                                   ;
 021776  012604                            MOV     (SP)+,R4
 022000  012603                            MOV     (SP)+,R3
 022002  012602                            MOV     (SP)+,R2
 022004  012601                            MOV     (SP)+,R1
 022006  000207                            RETURN
```

```
                                                  .SBTTL  STHNPV -- Initialize pointer vector in a handler
                                          ;-------------------------------------------------------------------------------
                                          ;  STHNPV is called to initialize the pointer vector at the end of a
                                          ;  handler which provides the addresses of various system routines to the
                                          ;  handler.
                                          ;
                                          ;  Inputs:
                                          ;   R3 = Address beyond the end of the handler.
                                          ;   HGENFL = Sysgen option flags for the handler being loaded.
                                          ;
 022010  010346                           STHNPV: MOV     R3,-(SP)
                                          ;
                                          ;  Set up addresses in the pointer vector
                                          ;
 022012  012743  000000G                          MOV     #FORK,-(R3)       ;Address of fork routine
 022016  012743  000000G                          MOV     #INTEN,-(R3)      ;Address of inten routine
 022022  032737  000000G 000240'                  BIT     #SG$IOT,HGENFL    ;Does handler want timeout support?
 022030  001402                                   BEQ     2$                ;Br if not
 022032  012743  000000G                          MOV     #IOTIMR,-(R3)     ;Set address of timeout support routine
 022036  032737  000000G 000240'          2$:     BIT     #SG$ELG,HGENFL    ;Does handler want error logging support?
 022044  001402                                   BEQ     3$                ;Br if not
 022046  012743  000000G                          MOV     #ERRLOG,-(R3)     ;Set address of error logging routine
 022052  012743  000000G                  3$:     MOV     #PTWRD,-(R3)
 022056  012743  000000G                          MOV     #PTBYT,-(R3)
 022062  012743  000000G                          MOV     #GTBYT,-(R3)
 022066  012743  000000G                          MOV     #MPPHY,-(R3)
 022072  012743  000000G                          MOV     #RELOC,-(R3)
                                          ;
                                          ;  Finished
                                          ;
 022076  012603                                   MOV     (SP)+,R3
 022100  000207                                   RETURN
```

```
                                     ;
                                     ;  Error occured while loading device handler.
                                     ;  R0 = error message address;  CURNAM = device name.
                                     ;
 022102  010001                      HLERR:  MOV     R0,R1           ;SAVE ERROR MESSAGE ADDRESS
 022104  105737  000000G                     TSTB    VINABT          ;ABORT OR JUST PRINT MESSAGE?
 022110  001416                              BEQ     9$              ;BR IF NOT ABORT
 022112                                      .PRINT  #TSXHD          ;PRINT ERROR MESSAGE HEADING
 022120                                      .PRINT  R1              ;PRINT ERROR MESSAGE
 022124  013700  000146'                     MOV     CURNAM,R0       ;GET RAD50 DEVICE NAME
 022130  004737  030020'                     CALL    PRTR50          ;PRINT DEVICE NAME
 022134                                      .PRINT  #CRLF
 022142  000137  004102'                     JMP     INISTP          ;ABORT INITIALIZATION
 022146  000261                      9$:     SEC                     ;MAKE SURE CARRY IS SET
 022150  000207                              RETURN
```

```
                                        .SBTTL  DOHNLC -- Execute and handler load/fetch code
                                ;-----------------------------------------------------------------------
                                ;   If the handler being loaded has any Load-time execution code, read it
                                ;   into our work buffer and execute it now.
                                ;
                                ;   Inputs:
                                ;    R4 = Device index number of handler that is being loaded.
                                ;
                                ;   Outputs:
                                ;    C-flag is set on return if load code signals an error during its
                                ;    execution.
                                ;
022152  010146                  DOHNLC: MOV     R1,-(SP)
022154  010246                          MOV     R2,-(SP)
022156  010346                          MOV     R3,-(SP)
022160  010446                          MOV     R4,-(SP)
022162  010546                          MOV     R5,-(SP)
                                ;
                                ;   Examine 1st word of handler to see if it could have any load-time code.
                                ;
022164  016405  000000G                 MOV     HANENT(R4),R5   ;Get address of handler entry point
022170  004737  022652'                 CALL    HANMAP          ;;;Map Kpar5 to handler if mapped handler
022174  016500  000004                  MOV     4(R5),R0        ;;;Get 1st instruction located at 4 in handler
022200  004737  022726'                 CALL    HANUMP          ;Restore normal mapping
022204  020027  000240                  CMP     R0,#240         ;Is it a NOP?
022210  103516                          BLO     7$              ;Br if can't be any load code
022212  020027  000277                  CMP     R0,#277         ;
022216  101113                          BHI     7$              ;Br if can't be any load code
022220  132700  000004                  BITB    #4,R0           ;Is there load code?
022224  001510                          BEQ     7$              ;Br if not
                                ;
                                ;   Handler may have load code.
                                ;   Read block 0 of handler and get offset to load code.
                                ;
022226  013702  000152'                 MOV     WRKBUF,R2       ;Get addr of our work buffer
022232                                  .READW  #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
022266  022227  031066                  CMP     (R2)+,#^RHAN    ;Is this a new type handler?
022272  001065                          BNE     7$              ;Br if not
022274  016203  000004                  MOV     4(R2),R3        ;Get offset to load code
022300  001462                          BEQ     7$              ;Br if there is none
                                ;
                                ;   There is load-time code.
                                ;   Read into WRKBUF the portion of the handler with the load code.
                                ;
022302  020327  001000                  CMP     R3,#1000        ;Is load code in block 0 of handler?
022306  103424                          BLO     1$              ;Br if yes
022310  010302                          MOV     R3,R2           ;Get offset to start of load code
022312  072227  177767                  ASH     #-9.,R2         ;Convert to a block number
022316  042702  177400                  BIC     #^C377,R2       ;Clear all but block number
022322                                  .READW  #AREA,#1,WRKBUF,#512.,R2 ;Read 2 blocks from handler file
                                ;
                                ;   The load code is now in WRKBUF.  Set up and execute it.
                                ;
022360  010437  000144'         1$:     MOV     R4,CURDEV       ;Save current device index number
022364  042703  177000                  BIC     #^C777,R3       ;Get offset within block of load code entry pt
022370  010300                          MOV     R3,R0           ;Get entry point offset
022372  063700  000152'                 ADD     WRKBUF,R0       ;Add base address
```

```
022376  013701  000000G                 MOV     RPRVEC,R1       ;Get pointer to GETVEC routine for Pro
022402  012702  000000C                 MOV     #MAXDEV*2,R2    ;Get 2** entries in device tables
022406  012703  000004                  MOV     #4,R3           ;Set code saying this is load code
022412  012705  000000G                 MOV     #HANENT,R5      ;Point to handler entry address vector
022416  060405                          ADD     R4,R5           ;Point to entry cell for this handler
022420  004737  022652'                 CALL    HANMAP          ;;;Map Kpar5 to handler if it is a mapped
022424  012704  022502'                 MOV     #LDREAD,R4      ;;;Get pointer to Read routine
022430  004710                          CALL    (R0)            ;;;Execute the load code
022432  103407                          BCS     2$              ;;;Br if handler load code signaled an error
                                ;
                                ;  Fetch/load code ran ok.
                                ;  Turn off handler mapping.
                                ;
022434  012737  001400  000000G         MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
022442  004737  022726'                 CALL    HANUMP          ;Unmap the handler
022446  000241                  7$:     CLC                     ;Clear the carry flag for return
022450  000406                          BR      9$
                                ;
                                ;  Error occurred in fetch/load code
                                ;
022452  012737  001400  000000G 2$:     MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
022460  004737  022726'                 CALL    HANUMP          ;Unmap the handler
022464  000261                          SEC                     ;Set the carry flag for return
                                ;
                                ;  Finished
                                ;
022466  012605                  9$:     MOV     (SP)+,R5
022470  012604                          MOV     (SP)+,R4
022472  012603                          MOV     (SP)+,R3
022474  012602                          MOV     (SP)+,R2
022476  012601                          MOV     (SP)+,R1
022500  000207                          RETURN
```

```
                                       .SBTTL  LDREAD -- Perform I/O for handler load code
                                ;-----------------------------------------------------------------------
                                ;  This routine performs Read operations for handler load code.
                                ;  It simulates the operation of the bootstrap read routine.
                                ;  When called, Channel 1 must be open to the handler file.
                                ;
                                ;  Inputs:
                                ;   R0 = Block number within handler file to be read.
                                ;   R1 = Number of words to read.
                                ;   R2 = Buffer address
                                ;
                                ;  Outputs:
                                ;   C-flag is set if a read error occurs
                                ;
022502  010046                  LDREAD: MOV     R0,-(SP)
022504  010446                          MOV     R4,-(SP)        ;;;
022506  010004                          MOV     R0,R4           ;;;Get starting block number
                                ;
                                ;  Save current mapping information
                                ;
022510  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256?
022514  001402                          BEQ     1$              ;;;Br if not
022516  013746  000000G                 MOV     @#SR3MMR,-(SP)  ;;;Save extended memory address register
022522  013746  000000G         1$.     MOV     @#SR0MMR,-(SP)  ;;;Save memory mapping
022526  013746  000000G                 MOV     @#KPAR5,-(SP)   ;;;Save current KPAR5 mapping
022532  013746  000000G                 MOV     @#KPAR6,-(SP)   ;;;Save current KPAR6 mapping
022536  012737  001400  000000G         MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
                                ;
                                ;  Turn off handler mapping
                                ;
022544  004737  022726'                 CALL    HANUMP          ;Turn off handler mapping
                                ;
                                ;  Read the requested data from the handler
                                ;
022550                                  .READW  #AREA,#1,R2,R1,R4 ;Read the blocks
022602  103420                          BCS     9$              ;Br if read error
                                ;
                                ;  Restore handler mapping
                                ;
022604  013704  000144'                 MOV     CURDEV,R4       ;Get current device index
022610  004737  022652'                 CALL    HANMAP          ;;;Map Kpar5 if necessary
                                ;
                                ;  Restore mapping information
                                ;
022614  012637  000000G                 MOV     (SP)+,@#KPAR6   ;;;Restore KPAR6 mapping
022620  012637  000000G                 MOV     (SP)+,@#KPAR5   ;;;Restore KPAR5 mapping
022624  012637  000000G                 MOV     (SP)+,@#SR0MMR  ;;;Restore memory mapping
022630  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256?
022634  001402                          BEQ     2$              ;;;Br if not
022636  012637  000000G                 MOV     (SP)+,@#SR3MMR  ;;;Restore extended memory address register
                                ;
                                ;  Finished
                                ;
022642  000241                  2$:     CLC                     ;;;Signal success on return
022644  012604                  9$:     MOV     (SP)+,R4
022646  012600                          MOV     (SP)+,R0
022650  000207                          RETURN
```

```
                                                .SBTTL  HANMAP -- Set up KPAR5 to access a mapped handler
                                        ;-----------------------------------------------------------------------
                                        ;  This routine is called to determine if a handler is mapped and if so
                                        ;  to turn on mapping and set up KPAR5 to access the mapped handler.
                                        ;  If the handler is not mapped, mapping is not turned on and KPAR5 is
                                        ;  not altered.
                                        ;  Interrupts are left disabled by this routine.
                                        ;  In addition to setting up mapping, this routine also changes the RMON
                                        ;  pointer to point to the TSX-Plus simulated RMON vector.
                                        ;
                                        ;  Inputs:
                                        ;   R4 = Device index number
                                        ;
 022652                                 HANMAP:
                                        ;
                                        ;  Disable interrupts
                                        ;
 022652                                         DISABL                  ;;;Disable interrupts
                                        ;
                                        ;  Change RMON pointer to point to TSX-Plus vector
                                        ;
 022660  012737  000000G 000000G                MOV     #MONVEC,@#RMON  ;;;Say TSX-Plus is the monitor
                                        ;
                                        ;  See if this handler is mapped
                                        ;
 022666  005764  000000G                        TST     HANPAR(R4)      ;;;Is this handler mapped?
 022672  001403                                 BEQ     9$              ;;;Br if not
                                        ;
                                        ;  This handler is mapped.
                                        ;  Set up mapping to access it.
                                        ;
 022674  016437  000000G 000000G                MOV     HANPAR(R4),@#KPAR5;;;Map KPAR5 to the handler code
 022702  052737  000000G 000000G        9$:     BIS     #MMENBL,@#SR0MMR;;;Enable memory mapping
 022710  105737  000000G                        TSTB    MEM256          ;;;Does machine have > 256KB?
 022714  001403                                 BEQ     10$             ;;;Br if not
 022716  052737  000000G 000000G                BIS     #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
                                        ;
                                        ;  Finished
                                        ;
 022724  000207                         10$:    RETURN

                                                .SBTTL  HANUMP -- Turn off memory mapping to a handler
                                        ;-----------------------------------------------------------------------
                                        ;  This routine is the companion to HANMAP.  It turns off memory mapping
                                        ;  and restores KPAR5 to its normal mapping value.
                                        ;  Enter with interrupts disabled. Interrupts are enabled on return.
                                        ;  This routine also changes the RMON pointer back to RT-11.
                                        ;
 022726                                 HANUMP:
                                        ;
                                        ;  Turn off memory management
                                        ;
 022726  105737  000000G                        TSTB    MEM256          ;;;Does machine have > 256KB?
 022732  001403                                 BEQ     1$              ;;;Br if not
 022734  042737  000000G 000000G                BIC     #EMMAP,@#SR3MMR ;;;Turn off extended memory addressing
 022742  042737  000000G 000000G        1$:     BIC     #MMENBL,@#SR0MMR;;;Turn off memory mapping
 022750  012737  001200  000000G                MOV     #1200,@#KPAR5   ;;;Reset KPAR5 to its normal mapping
```

```
     58                                        ;
     59                                        ;   Restore RMON pointer to RT11
     60                                        ;
     61 022756  013737  000046' 000000G                MOV     RTMNVC,@#RMON   ;;;Reset RMON pointer
     62                                        ;
     63                                        ;   Enable interrupts
     64                                        ;
     65 022764                                         ENABL                   ;Enable interrupts
     66                                        ;
     67                                        ;   Finished
     68                                        ;
     69 022772  000207                                 RETURN
```

```
                                          .IF     EQ,<PROASM-1>   ;No handler XM support if Pro-only
                                          .SBTTL  FNDHRB,HANXMR Inoperative Pro versions
                                  ;-----------------------------------------------------------------
                                  ;  None of the current device handlers on the Pro require handler XM
                                  ;  region support.  Make sure they return intelligent errors if
                                  ;  someone does try to use them.
                                  ;
                                          .ENABL  LSB
                                  FNDHRB: CLR     R1              ;Point to next region control block (none)
                                          BR      1$              ;Go signal error and return
                                  ;
                                  HANXMR: CLR     R2              ;Return largest possible region size (none)
                                  1$:     SEC                     ;Signal error on request
                                          RETURN
                                          .DSABL  LSB
                                  ;
                                          .IFF    ;EQ,<PROASM-1>  ;Include handler XM support for 11 versions
                                          .SBTTL  FNDHRB -- Try to find a handler global region
                                  ;--------------------------------------------------------------------
                                  ;  This routine is called to try to locate an allocated XM region with
                                  ;  a specified name.
                                  ;  If a region control block with the specified name cannot be found,
                                  ;  the address of a free one is returned and the specified name is stored
                                  ;  into the free block.
                                  ;
                                  ;  Inputs:
                                  ;   R5 = Pointer to 2-word cell containing Rad50 name of region to be found.
                                  ;
                                  ;  Outputs:
                                  ;   C-flag cleared ==> Found the specified RCB.
                                  ;       R1 = Address of the RCB
                                  ;   C-flag set ==> Could not find the specified RCB.
                                  ;       R1 = Pointer to a free RCB or 0 if no available RCB's.
                                  ;
022774  010246                    FNDHRB: MOV     R2,-(SP)
                                  ;
                                  ;  Search for specified RCB and also remember if we see a free RCB
                                  ;
022776  005002                            CLR     R2              ;Say no free RCB found
023000  013701  000000G                   MOV     HANRCB,R1       ;Point to start of RCB area
023004  005721                            TST     (R1)+           ;Skip over -1 word at front
023006  005711                    1$:     TST     (R1)            ;What is the status of this RCB
023010  001002                            BNE     2$              ;Br if this is not a free RCB
023012  010102                            MOV     R1,R2           ;Remember address of a free RCB
023014  000412                            BR      3$
023016  021127  177777            2$:     CMP     (R1),#-1        ;Are we at the end of the list?
023022  001412                            BEQ     4$              ;Br if yes
023024  021561  000006                    CMP     (R5),6(R1)      ;Compare the names
023030  001004                            BNE     3$              ;Br if don't match
023032  026561  000002  000010            CMP     2(R5),10(R1)    ;Compare 2nd half of name
023040  001417                            BEQ     6$              ;Br if found the RCB we were searching for
023042  062701  000012            3$:     ADD     #12,R1          ;Point to the next RCB
023046  000757                            BR      1$              ;Continue searching
                                  ;
                                  ;  We could not find the specified RCB.
                                  ;  If there was a free one, initialize the name.
                                  ;
```

```
023050  010201                          4$:     MOV     R2,R1           ;Was there a free RCB?
023052  001410                                  BEQ     5$              ;Br if not
023054  011561  000006                          MOV     (R5),6(R1)      ;Set name in the RCB
023060  016561  000002  000010                  MOV     2(R5),10(R1)    ;(2nd word of name)
023066  063761  000144' 000010                  ADD     CURDEV,10(R1)   ;Make name unique to device
023074  000261                          5$:     SEC                     ;Signal that we did not find the RCB
023076  000401                                  BR      9$
                                        ;
                                        ;  We found the RCB
                                        ;
023100  000241                          6$:     CLC                     ;Signal success on return
                                        ;
                                        ;  Finished
                                        ;
023102  012602                          9$:     MOV     (SP)+,R2
023104  000207                                  RETURN
```

22774
22152
622

```
                                        .SBTTL  HANXMR -- Allocate XM region during handler load
                                ;-----------------------------------------------------------------------------
                                ;  This routine can be called by a handler as its is being loaded to
                                ;  allocate an XM region for the handler.
                                ;
                                ;  Inputs:
                                ;   R2 = Number of 64-byte units needed for XM region
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Successfully allocated a region
                                ;       R1 = 64-byte address of base of allocated region
                                ;       R2 = Requested size
                                ;   C-flag set ==> Could not allocate the region
                                ;       R2 = Largest available region size
                                ;
                                ;  Notes: FMEMLO and FMEMHI are used by this routine to indicate the
                                ;  bottom and top of free memory area that can be allocated.
                                ;  The allocation is done from the top of free memory downward.
                                ;  FMEMHI is updated to have the new top of free memory after the
                                ;  allocation has been done.
                                ;
023106  010046                  HANXMR: MOV     R0,-(SP)
                                ;
                                ;  Get the total amount of free memory space available now
                                ;  and see if the requested region can be allocated.
                                ;
023110  013700  000134'                 MOV     FMEMHI,R0       ;Top of free memory
023114  163700  000136'                 SUB     FMEMLO,R0       ;-Base of free memory
023120  020200                          CMP     R2,R0           ;Do we have room for the requested region?
023122  101006                          BHI     8$              ;Br if not
                                ;
                                ;  There is room for the region so allocate it from the top of memory
                                ;
023124  160237  000134'                 SUB     R2,FMEMHI       ;Allocate the region
023130  013701  000134'                 MOV     FMEMHI,R1       ;Return the address of the base of the region
023134  000241                          CLC                     ;Signal success on return
023136  000402                          BR      9$
                                ;
                                ;  We are not able to allocate the region.
                                ;  Return with C-flag set and the size of the largest possible region in R2.
                                ;
023140  010002                  8$:     MOV     R0,R2           ;Get the size of the largest possible region
023142  000261                          SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
023144  012600                  9$:     MOV     (SP)+,R0
023146  000207                          RETURN

                                        .ENDC   ;EQ,<PROASM-1>
```

```
                                                .IF     NE,<PROASM-1>   ;If assembling for PDP-11
                                                .SBTTL  SETMIO -- Set up information about mapped devices
                                        ;-----------------------------------------------------------------------
                                        ;  SETMIO is called to set up information about which devices have to have
                                        ;  their I/O mapped through system buffers. I/O mapping is done for DMA
                                        ;  devices with 18-bit controllers being used on Q-bus systems with more
                                        ;  than 256Kb of memory.
                                        ;  The DX$MAP flag is set in the DVFLAG word for devices that require mapping.
                                        ;
                                        ;  Inputs:
                                        ;   R5 = Pointer to low-memory free area
                                        ;
                                        ;  Outputs:
                                        ;   MIOFLAG = 0==>I/O mapping not required for any device;
                                        ;             1==>I/O mapping required for some device.
                                        ;   R5 = Pointer to new low-memory free area.
                                        ;
 023150  010146                         SETMIO: MOV     R1,-(SP)
                                        ;
                                        ;  Determine if this machine requires mapping at all
                                        ;
 023152  005727  000000G                        TST     #MIODBG         ;Are we debugging mapped I/O system?
 023156  001017                                 BNE     2$              ;Br if yes
 023160  032737  000001  000242'                BIT     #EXTLSI,ICONFG  ;Is this a Q-bus machine with more than 256Kb?
 023166  001013                                 BNE     2$              ;Br if yes
                                        ;
                                        ;  This is not a Q-bus system with more than 256Kb.
                                        ;  Mapping is not required at all.
                                        ;
 023170  012701  000002                         MOV     #2,R1           ;Get initial device index number
 023174  042761  000000G 000000G 1$:            BIC     #DX$MAP,DVFLAG(R1) ;Clear mapped-I/O flag
 023202  062701  000002                         ADD     #2,R1           ;Get next device index
 023206  020137  000000G                        CMP     R1,NUMDEV       ;More to do?
 023212  101770                                 BLOS    1$              ;Br if yes
 023214  000464                                 BR      9$
                                        ;
                                        ;  This is a Q-bus system with more than 256Kb.
                                        ;  See if any devices have requested mapped I/O.
                                        ;
 023216  005000                         2$:     CLR     R0              ;Clear composite flag word
 023220  012701  000002                         MOV     #2,R1           ;Initialize device index number
 023224  056100  000000G                3$:     BIS     DVFLAG(R1),R0   ;Combine flags from all devices
 023230  062701  000002                         ADD     #2,R1           ;Get next device index number
 023234  020137  000000G                        CMP     R1,NUMDEV       ;Checked all devices?
 023240  101771                                 BLOS    3$              ;Br if not
 023242  032700  000000G                        BIT     #DX$MAP,R0      ;Does any device require mapping?
 023246  001447                                 BEQ     9$              ;Br if not
                                        ;
                                        ;  I/O mapping is required
                                        ;
 023250  105237  000000G                        INCB    MIOFLG          ;Remember that mapping is required
                                        ;
                                        ;  Zero the area where we will build the control structures
                                        ;
 023254  113701  000000G                        MOVB    VMIOBF,R1       ;Get # buffers wanted
 023260  070127  000000G                        MUL     #MI$$SZ,R1      ;Times size for each control block
 023264  062701  000000C                        ADD     #MIONWB*MW$$SZ,R1;Add space for MIO wait blocks
```

```
023270  006201                          ASR     R1                      ;Get # words to zero
023272  010500                          MOV     R5,R0                   ;Get pointer to start of area
023274  005020                  4$:     CLR     (R0)+                   ;Zero the entire area
023276  077102                          SOB     R1,4$
                                ;
                                ;  Allocate and initialize the control structures
                                ;
023300  010537  000000G                 MOV     R5,MIOBHD               ;Start of area for I/O mapping control blks
023304  113700  000000G                 MOVB    VMIOBF,R0               ;Get # buffers wanted
023310  010501                  5$:     MOV     R5,R1                   ;Get pointer to current control block
023312  062701  000000G                 ADD     #MI$$SZ,R1              ;Get pointer to next control block
023316  005300                          DEC     R0                      ;Need to link more together?
023320  003404                          BLE     6$                      ;Br if not
023322  010165  000000G                 MOV     R1,MI$LNK(R5)           ;Set pointer to next control block
023326  010105                          MOV     R1,R5                   ;Advance pointer to next block
023330  000767                          BR      5$                      ;See if more to do
023332  010105                  6$:     MOV     R1,R5                   ;Set pointer past last block
023334  010537  000000G                 MOV     R5,MIOWHD               ;Start of wait blocks
023340  012700  177777G                 MOV     #MIONWB-1,R0            ;Get # wait blocks wanted - 1
023344  010501                  7$:     MOV     R5,R1                   ;Get pointer to current block
023346  062701  000000G                 ADD     #MW$$SZ,R1              ;Get pointer to next block
023352  010165  000000G                 MOV     R1,MW$LNK(R5)           ;Set pointer to next wait block
023356  010105                          MOV     R1,R5                   ;Advance pointer to next block
023360  077007                          SOB     R0,7$                   ;Loop if more to allocate
023362  062705  000000G                 ADD     #MW$$SZ,R5              ;Allocate space for last block (with 0 link)
                                ;
                                ;  Finished
                                ;
023366  012601                  9$:     MOV     (SP)+,R1
023370  000207                          RETURN
                                        .IFF    ;NE,<PROASM-1>
                                ;
                                ;  Define dummy routine for Pro
                                ;
                                SETMIO: RETURN
                                        .ENDC   ;NE,<PROASM-1>
```

```
                                            .SBTTL  OVLPOS -- Determine which overlays go over TSINIT
                                    ;-----------------------------------------------------------------------
                                    ;  OVLPOS is called to determine which system overlays are to be placed
                                    ;  over the TSINIT code (specifically, between @OVLBAS and INITOP).
                                    ;
                                    ;  Inputs:
                                    ;   R5 = Base address in TSINIT where overlays may be loaded.
                                    ;
                                    ;  Outputs:
                                    ;   Overlay segment information is set up in OSTABL.
                                    ;   OS$FLG(seg) = 0==>Load seg into high memory; 1==>Load over TSINIT.
                                    ;   OVLBAS = Address of location within TSINIT where overlays start.
                                    ;   R5 = Pointer past last overlay loaded over TSINIT.
                                    ;
 023372  010146                     OVLPOS: MOV     R1,-(SP)
 023374  010246                             MOV     R2,-(SP)
 023376  010346                             MOV     R3,-(SP)
 023400  010446                             MOV     R4,-(SP)
 023402  010504                             MOV     R5,R4           ;Get address where we may load overlays
                                    ;
                                    ;  Build the table that holds information about the overlays
                                    ;
 023404  004737  023602'                    CALL    OVLBLD          ;Build overlay information table
                                    ;
                                    ;  First determine how much space will be used by those overlays that are
                                    ;  forced to be loaded over TSINIT.
                                    ;
 023410  062704  000077                     ADD     #63.,R4         ;Bound address to 64-byte boundary
 023414  042704  000077                     BIC     #77,R4
 023420  010437  000140'                    MOV     R4,OVLBAS       ;Remember address where we load overlays
 023424  012702  000516'                    MOV     #OSTABL,R2      ;Point to start of table
 023430  005762  000000             1$:     TST     OS$SIZ(R2)      ;Is this overlay to be loaded?
 023434  001423                             BEQ     2$              ;Br if not
 023436  016200  000004                     MOV     OS$OVL(R2),R0   ;Point to linker-built entry
 023442  016000  000000G                    MOV     O.ADR(R0),R0    ;Get Rad50 segment ID
 023446  012701  000746'                    MOV     #LOWOVL,R1      ;Point to table of overlays to go over TSINIT
 023452  020021                     6$:     CMP     R0,(R1)+        ;Must this overlay go over TSINIT?
 023454  001404                             BEQ     4$              ;Br if yes
 023456  020127  000754'                    CMP     R1,#LOWEND      ;End of low-overlay table?
 023462  103773                             BLO     6$              ;Br if not
 023464  000407                             BR      2$              ;This overlay is not forced over TSINIT
 023466  005262  000002             4$:     INC     OS$FLG(R2)      ;Set flag saying load over TSINIT
 023472  016200  000000                     MOV     OS$SIZ(R2),R0   ;Get # 64-byte blocks needed for overlay
 023476  072027  000006                     ASH     #6,R0           ;Get # bytes needed for overlay
 023502  060004                             ADD     R0,R4           ;Advance address within TSINIT
 023504  062702  000006             2$:     ADD     #OS$$SZ,R2      ;Point to entry for next segment
 023510  020237  000744'                    CMP     R2,OSLAST       ;Have we finished?
 023514  103745                             BLO     1$              ;Loop if not
                                    ;
                                    ;  Determine how much memory space is available in TSINIT for other overlays
                                    ;
 023516  020427  030012'                    CMP     R4,#INITOP-50.  ;Any space left for other overlays?
 023522  103021                             BHIS    9$              ;Br if not
 023524  012705  030074'                    MOV     #INITOP,R5      ;Point to top of overlay area
 023530  160405                             SUB     R4,R5           ;Total space available for overlays
 023532  072527  177772                     ASH     #-6,R5          ;Convert to # 64-byte blocks
                                    ;
```

```
                                ;       Now begin loop which determines which other overlays go over TSINIT.
                                ;       We do this in the order of largest to smallest to try to fill
                                ;       the overlay area as completely as possible.
                                ;
023536  004737  025012'         3$:     CALL    OVLTRY          ;Try to find largest overlay that will fit
023542  103411                          BCS     9$              ;Br if no more overlays will fit
023544  005262  000002                  INC     OS$FLG(R2)      ;Remember to load over TSINIT
023550  016200  000000                  MOV     OS$SIZ(R2),R0   ;Get # 64-byte blocks needed for overlay
023554  160005                          SUB     R0,R5           ;Reduce remaining free space in TSINIT
023556  072027  000006                  ASH     #6,R0           ;Get # bytes needed for overlay
023562  060004                          ADD     R0,R4           ;Advance overlay address in TSINIT
023564  000764                          BR      3$              ;See if we can find more segments to load
                                ;
                                ;       Finished
                                ;
023566  010405                  9$:     MOV     R4,R5           ;Return top-of-overlay address in R5
023570  012604                          MOV     (SP)+,R4
023572  012603                          MOV     (SP)+,R3
023574  012602                          MOV     (SP)+,R2
023576  012601                          MOV     (SP)+,R1
023600  000207                          RETURN
```

```
                                                .SBTTL  OVLBLD -- Build overlay information table
                                        ;-----------------------------------------------------------------------
                                        ;  OVLBLD is called to build an overlay information table that is used
                                        ;  by TSINIT while loading TSX overlays into memory.
                                        ;
                                        ;  Outputs:
                                        ;   Overlay segment information is set up in OSTABL.
                                        ;   OSLAST = Pointer past last entry in OSTABL.
                                        ;
023602  010146                          OVLBLD: MOV     R1,-(SP)
023604  010246                                  MOV     R2,-(SP)
023606  010346                                  MOV     R3,-(SP)
                                        ;
                                        ;  Read 1st block of SAV file to get pointer to overlay table
                                        ;
023610  013702  000152'                         MOV     WRKBUF,R2       ;Point to work buffer
023614                                          .READW  #AREA,#17,R2,#256.,#0 ;read first block of the save file
023650  103444                                  BCS     22$             ;Br if error on read
023652  016201  000064                          MOV     64(R2),R1       ;point to the overlay table
023656  001012                                  BNE     15$             ;br if overlays exist
                                        ;
                                        ;  Must be verion 3B overlays structure at absolute location.
                                        ;
023660  012737  000137  001000                  MOV     #137,@#1000     ;position jump instrucion over 3b ovly handler
023666  012737  000000G 001002                  MOV     #$OVRH,@#1002   ;position overlay intercept location
023674  012701  001104                          MOV     #1104,R1        ;point to the overlay table
023700  010137  000000G                         MOV     R1,OVRADD       ;save the address of the overlay table
                                        ;
                                        ;  Initialize the table that holds information about the overlays
                                        ;
023704  012703  000516'                 15$:    MOV     #OSTABL,R3      ;Point to table for overlay info
023710  010163  000004                  11$:    MOV     R1,OS$OVL(R3)   ;Save pointer to overlay control block
023714  005063  000002                          CLR     OS$FLG(R3)      ;Assume seg will be loaded in high memory
023720  004737  024122'                         CALL    ALCOVL          ;Determine if we should load this overlay
023724  010263  000000                          MOV     R2,OS$SIZ(R3)   ;Remember total size of overlay+data
023730  062703  000006                          ADD     #OS$$SZ,R3      ;Point to next overlay table entry
023734  062701  000006                  12$:    ADD     #6,R1           ;find the next region
023740  021127  004537                          CMP     (R1),#4537      ;compare with a <JSR R5,$OVRH> instruction
023744  001361                                  BNE     11$             ;Br if not at end
023746  010337  000744'                         MOV     R3,OSLAST       ;Save pointer past last overlay table entry
                                        ;
                                        ;  Finished
                                        ;
023752  012603                                  MOV     (SP)+,R3
023754  012602                                  MOV     (SP)+,R2
023756  012601                                  MOV     (SP)+,R1
023760  000207                                  RETURN
                                        ;
                                        ;  Error -- Read error occured while reading overlay table
                                        ;
023762                                  22$:    .PRINT  #TSXHD
023770                                          .PRINT  #RDERR
023776  000137  004102'                         JMP     INISTP
```

```
                                     .SBTTL  GETMAP -- Load any mapped system code regions
                           ;-----------------------------------------------------------------------
                           ;  GETMAP is called to load those system overlays that are placed
                           ;  in high memory.
                           ;
                           ;  Inputs:
                           ;    R5 = 64-byte block number of top of free memory.
                           ;
                           ;  Outputs:
                           ;   R5 = New 64-byte block number of top of free memory.
                           ;
024002  010146             GETMAP: MOV     R1,-(SP)
024004  010246                     MOV     R2,-(SP)
024006  010346                     MOV     R3,-(SP)
024010  010537  000000G            MOV     R5,SMRSIZ         ;Save memory pointer at start of allocation
                           ;
                           ;  Now that most of the system initialization is completed, we must check
                           ;  again to see which overlays need to be loaded.
                           ;
024014  012703  000516'            MOV     #OSTABL,R3        ;Point to 1st overlay table entry
024020  004737  024266'    1$:     CALL    OPTOVL            ;See if this segment should be loaded
024024  010263  000000             MOV     R2,OS$SIZ(R3)     ;Save # 64-byte blocks needed for overlay
024030  062703  000006             ADD     #OS$$SZ,R3        ;Point to next overlay table entry
024034  020337  000744'            CMP     R3,OSLAST         ;Checked all entries in overlay table?
024040  103767                     BLO     1$                ;Br if not
                           ;
                           ;  Load those overlays that go into high memory
                           ;
024042  012702  000516'            MOV     #OSTABL,R2        ;Point to 1st overlay entry
024046  005762  000000     3$:     TST     OS$SIZ(R2)        ;Is this overlay segment wanted?
024052  001405                     BEQ     4$                ;Br if not
024054  005762  000002             TST     OS$FLG(R2)        ;Load over TSINIT or into high memory?
024060  001002                     BNE     4$                ;Br if load over TSINIT
024062  004737  025110'            CALL    GETOVL            ;Load overlay into high memory
024066  062702  000006     4$:     ADD     #OS$$SZ,R2        ;Point to next overlay table entry
024072  020237  000744'            CMP     R2,OSLAST         ;Have we done all overlays?
024076  103763                     BLO     3$                ;Loop if not
                           ;
                           ;  Finished
                           ;
024100  013700  000000G    19$:    MOV     SMRSIZ,R0         ;Get memory pointer at start of allocation
024104  160500                     SUB     R5,R0             ;Calc amt of space allocated
024106  010037  000000G            MOV     R0,SMRSIZ         ;Save total space used for mapped regions
024112  012603                     MOV     (SP)+,R3
024114  012602                     MOV     (SP)+,R2
024116  012601                     MOV     (SP)+,R1
024120  000207                     RETURN
```

```
                                               .SBTTL  ALCOVL -- Allocate space for a system overlay region
                                       ;---------------------------------------------------------------------------
                                       ;  ALCOVL is called to determine if a system overlay region is wanted
                                       ;  (based on sysgen options), and if it is wanted to determine how
                                       ;  much space is needed for the code and data.
                                       ;
                                       ;  Inputs:
                                       ;   R3 = Pointer to overlay table entry (OS$xxx)
                                       ;
                                       ;  Outputs:
                                       ;   C-flag cleared ==> This segment is to be loaded.
                                       ;   C-flag set     ==> Do not load this overlay segment.
                                       ;   R2 = # 64-Byte blocks needed for segment including data areas within it.
                                       ;
024122  010146                         ALCOVL: MOV     R1,-(SP)
                                       ;
                                       ;  Get pointer to linker-build overlay entry for segment
                                       ;
024124  016301  000004                         MOV     OS$OVL(R3),R1   ;Get pointer to linker-built entry for seg
                                       ;
                                       ;  Read in the first block of the overlay segment
                                       ;
024130  013702  000152'                        MOV     WRKBUF,R2       ;Point to work buffer
024134                                         .READW  #AREA,#17,R2,#256.,O.BLK(R1) ;read the first block
024172  103415                                 BCS     3$              ;Br if read error
                                       ;
                                       ;  Save the 3 character Rad50 segment ID in the O.ADR cell of the
                                       ;  linker-built overlay table entry for this segment.
                                       ;
024174  016261  000002  000000G                MOV     2(R2),O.ADR(R1) ;save the rad50 overlay identifier
                                       ;
                                       ;  Make sure the segment is not larger than 8Kb
                                       ;
024202  016102  000000G                        MOV     O.SIZ(R1),R2    ;get the word count of the code region
024206  006302                                 ASL     R2              ;convert to byte count
024210  020227  020000                         CMP     R2,#20000       ;check for 8kb overflow
024214  101014                                 BHI     21$             ;Br if region is too big
                                       ;
                                       ;  Don't load some optional segments if features were not selected
                                       ;  in TSGEN.
                                       ;
024216  004737  024266'                        CALL    OPTOVL          ;See if we want to load this segment
                                       ;
                                       ;  Finished
                                       ;  The C-flag is set or reset by OPTOVL.
                                       ;
024222  012601                                 MOV     (SP)+,R1
024224  000207                                 RETURN
                                       ;
                                       ;  Error -- Error on reading from SAV file
                                       ;
024226                                 3$:     .PRINT  #TSXHD          ;Print heading
024234                                         .PRINT  #RDERR          ;Read error
024242  000137  004102'                        JMP     INISTP          ;Abort initialization
                                       ;
                                       ;  Error -- Insufficient memory space to load run-time systems
                                       ;
```

```
     58 024246                      21$:     .PRINT   #TSXHD          ;PRINT HEADING
     59 024254                               .PRINT   #TSXSIZ         ;PRINT ERROR MESSAGE
     60 024262  000137  004102'              JMP      INISTP          ;ABORT INITIALIZATION
```

```
                                        .SBTTL  OPTOVL -- Check for optional system overlay regions
                                ;---------------------------------------------------------------------
                                ;  OPTOVL is called to determine if a specific system overlay is or is
                                ;  not to be loaded based on sysgen options.
                                ;  This routine may also add space for buffers to the overlay regions size.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to overlay table entry for segment (OS$xxx)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Load this overlay.
                                ;   C-flag set     ==> Do not load this overlay.
                                ;   R2 = # 64-byte blocks needed for code + data for the segment.
                                ;
024266  010346                  OPTOVL: MOV     R3,-(SP)
024270  010446                          MOV     R4,-(SP)
024272  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get the name of the overlay segment
                                ;
024274  016305  000004                  MOV     OS$OVL(R3),R5   ;Get pointer to linker-built entry
024300  016504  000000G                 MOV     O.ADR(R5),R4    ;Get name of the segment
                                ;
                                ;  Get size of code portion of overlay segment
                                ;
024304  016502  000000G                 MOV     O.SIZ(R5),R2    ;Get # words needed by code portion of seg
024310  006302                          ASL     R2              ;Convert to # bytes
                                ;
                                ;  See if this is an optional segment that we need to deal with specially
                                ;
024312  012700  024340'                 MOV     #OVLLST,R0      ;Point to overlay name list
024316  020420                  1$:     CMP     R4,(R0)+        ;Found name of overlay?
024320  001406                          BEQ     2$              ;Br if yes
024322  005720                          TST     (R0)+           ;No -- Skip over address word
024324  020027  024430'                 CMP     R0,#OVLEND      ;Checked all names in the list?
024330  103772                          BLO     1$              ;Loop if not
024332  000137  024750'                 JMP     OOXYES          ;Load this overlay
                                ;
                                ;  Branch off to processing routine
                                ;
024336  000130                  2$:     JMP     @(R0)+          ;Enter processing routine for the overlay
                                ;
                                ;  Table of overlay names and processing routines
                                ;
                                        .MACRO  OVLTBL  NAME
                                        .RAD50  /'NAME'/
                                        .WORD   OOR'NAME
                                        .ENDM   OVLTBL
                                ;
024340                          OVLLST:
024340                                  OVLTBL  USR             ;TSUSR  -- File management
024344                                  OVLTBL  SPL             ;TSSPOL -- Spooling system
024350                                  OVLTBL  SP2             ;TSSPL2 -- Spooler flag pages
024354                                  OVLTBL  LOK             ;TSLOCK -- Shared file record locking
024360                                  OVLTBL  MSG             ;TSMSG  -- Inter-job message communication
024364                                  OVLTBL  SWP             ;TSSWAP -- Job swapper
024370                                  OVLTBL  PLS             ;TSPLAS -- PLAS support
```

```
024374                             OVLTBL  SLE             ;TSSLE  -- Single line editor
024400                             OVLTBL  WIN             ;TSWIN  -- Display window management
024404                             OVLTBL  MIO             ;TSMIO  -- Mapped I/O
024410                             OVLTBL  CLO             ;TSCLO  -- CL handler
024414                             OVLTBL  DBG             ;TSDBUG -- Program debugger
024420                             OVLTBL  CSH             ;TSCASH -- Data caching
024424                             OVLTBL  DMP             ;TSDUMP -- Crash dump generator
024430                     OVLEND:
                           ;
                           ;    File management
                           ;
024430  013703  000000G    OORUSR: MOV     VNFCSH,R3       ;Get # file cache entries
024434  070327  000000G            MUL     #FC$$SZ,R3      ;Multiply by size of each entry
024440  060302                     ADD     R3,R2           ;Allocate space for directory cache
024442  000542                     BR      OOXYES          ;Load the segment
                           ;
                           ;    Spooling system
                           ;
024444  005727  000000G    OORSPL: TST     #SPLND          ;Are there any spooled devices?
024450  001534                     BEQ     OOXNO           ;Br if not
024452  062702  000000C            ADD     #<SPLNB*512.>,R2;Reserve room for spool buffers
024456  013703  000000G            MOV     NSPLBL,R3       ;Get # blocks for spool file
024462  062703  000007             ADD     #7,R3           ;Bound up to byte boundary
024466  072327  177775             ASH     #-3,R3          ;Divide by 8 to get # bytes for table
024472  005203                     INC     R3              ;Round up to word boundary
024474  042703  000001             BIC     #1,R3
024500  060302                     ADD     R3,R2           ;Add space for spool file allocation table
024502  000522                     BR      OOXYES          ;Load the segment
                           ;
                           ;    Spooler flag pages?
                           ;
024504  005727  000000G    OORSP2: TST     #SPLND          ;Are there any spooled devices?
024510  001514                     BEQ     OOXNO           ;If not, don't load overlay
024512  000516                     BR      OOXYES          ;Load the segment
                           ;
                           ;    Record locking system
                           ;
024514  005737  000000G    OORLOK: TST     VMXSF           ;Any shared files?
024520  001510                     BEQ     OOXNO           ;Br if not
024522  005737  000000G            TST     VNUMDC          ;Shared file data caching wanted?
024526  001110                     BNE     OOXYES          ;Br if yes
024530  162702  000000G            SUB     #DCCSIZ,R2      ;Reduce size of segment - Leave out cache code
024534  000505                     BR      OOXYES          ;Load the segment
                           ;
                           ;    Message communication system
                           ;
024536  013703  000000G    OORMSG: MOV     VMAXMC,R3       ;Is message communication facility wanted?
024542  001477                     BEQ     OOXNO           ;Br if not
024544  070327  000000G            MUL     #MB$$SZ,R3      ;Space for message channel blocks
024550  060302                     ADD     R3,R2
024552  013703  000000G            MOV     VMXMRB,R3       ;Number of message request blocks
024556  070327  000000G            MUL     #MR$$SZ,R3      ;Times size of request block
024562  060302                     ADD     R3,R2
024564  013703  000000G            MOV     VMSCHR,R3       ;Max # chars in a message
024570  005203                     INC     R3              ;Bound up to word
024572  042703  000001             BIC     #1,R3           ;Reserve whole number of words
024576  062703  000000G            ADD     #MU$TXT,R3      ;Plus space for message header
```

```
024602  070337  000000G                  MUL     VMXMSG,R3       ;Times maximum number of messages
024606  060302                           ADD     R3,R2           ;Space for message buffers
024610  000457                           BR      OOXYES
                                ;
                                ;  PLAS support
                                ;
024612  013703  000000G         OORPLS:  MOV     VPLAS,R3        ;PLAS support wanted?
024616  001451                           BEQ     OOXNO           ;Br if not
024620  062703  000021                   ADD     #17.,R3         ;Bound up # blocks
024624  072327  177775                   ASH     #-3,R3          ;Get # bytes needed for swap file bit map
024630  060302                           ADD     R3,R2           ;Reserve room for swap file bit map
024632  000446                           BR      OOXYES          ;Load the segment
                                ;
                                ;  Job swapper
                                ;
024634  105737  000000G         OORSWP:  TSTB    VSWPFL          ;Is this a swapping system?
024640  001440                           BEQ     OOXNO           ;Br if not
024642  000442                           BR      OOXYES          ;Br if yes -- Load the segment
                                ;
                                ;  Single line editor
                                ;
024644  105737  000000G         OORSLE:  TSTB    VSLEDT          ;Is SL editor wanted?
024650  001434                           BEQ     OOXNO           ;Br if not
024652  000436                           BR      OOXYES          ;Load the segment
                                ;
                                ;  Display windows
                                ;
024654  013703  000000G         OORWIN:  MOV     VMXWIN,R3       ;Are any display windows wanted?
024660  001430                           BEQ     OOXNO           ;Br if not
024662  070327  000000G                  MUL     #DW$$SZ,R3      ;Amt of space needed for window control blks
024666  060302                           ADD     R3,R2           ;Add to size of overlay
024670  000427                           BR      OOXYES          ;Load the segment
                                ;
                                ;  Mapped I/O
                                ;
024672  105737  000000G         OORMIO:  TSTB    MIOFLG          ;Is I/O mapping needed?
024676  001421                           BEQ     OOXNO           ;Br if not
024700  000423                           BR      OOXYES          ;Load the segment
                                ;
                                ;  CL handler
                                ;
024702  005727  000000G         OORCLO:  TST     #CLTOTL         ;Any I/O lines?
024706  001415                           BEQ     OOXNO           ;Br if not
024710  000417                           BR      OOXYES          ;Yes, load the segment
                                ;
                                ;  Program debugger
                                ;
024712  105737  000000G         OORDBG:  TSTB    VDBFLG          ;Is the program debugger wanted?
024716  001411                           BEQ     OOXNO           ;Br if not
024720  000413                           BR      OOXYES          ;Load this segment
                                ;
                                ;  Data caching
                                ;
024722  005737  000000G         OORCSH:  TST     CSHALC          ;Is data caching wanted?
024726  001405                           BEQ     OOXNO           ;Br if not
024730  000407                           BR      OOXYES          ;Load this segment
                                ;
```

```
                                        ;  Crash dump generator
                                        ;
 024732  105737  000000G        OORDMP: TSTB    VSYDMP          ;Is dump facility wanted?
 024736  001401                         BEQ     OOXNO           ;Br if not
 024740  000403                         BR      OOXYES          ;Br if yes
                                        ;
                                        ;  Don't load this segment
                                        ;
 024742  005002                 OOXNO:  CLR     R2              ;Say no space needed for overlay
 024744  000261                         SEC                     ;Signal don't load the segment
 024746  000415                         BR      OOXFIN
                                        ;
                                        ;  Load this segment
                                        ;
 024750  005202                 OOXYES: INC     R2              ;Make sure size is even
 024752  042702  000001                 BIC     #1,R2
 024756  020227  020000                 CMP     R2,#8192.       ;Don't allow code + data to exceed 8Kb
 024762  101402                         BLOS    1$              ;Br if ok
 024764  012702  020000                 MOV     #8192.,R2       ;Note, init code in segment will truncate dat
 024770  062702  000077         1$:     ADD     #63.,R2         ;Convert to # 64-byte blocks
 024774  072227  177772                 ASH     #-6,R2
 025000  000241                         CLC                     ;Signal to load the segment
                                        ;
                                        ;  Finished
                                        ;
 025002  012605                 OOXFIN: MOV     (SP)+,R5
 025004  012604                         MOV     (SP)+,R4
 025006  012603                         MOV     (SP)+,R3
 025010  000207                         RETURN
```

```
                                        .SBTTL  OVLTRY -- Find an overlay to place over TSINIT
                                ;-----------------------------------------------------------------------
                                ;  OVLTRY is called to identify the largest overlay segment which
                                ;  will fit in the TSINIT area and which is not already marked to go
                                ;  over TSINIT.
                                ;
                                ;  Inputs:
                                ;   R5 = # 64-byte blocks available for segment in TSINIT.
                                ;
                                ;  Outputs:
                                ;   R2 = Pointer to OSTABL entry for segment
                                ;   C-flag set ==> No more segments will fit.
                                ;
025012  010346                  OVLTRY: MOV     R3,-(SP)
                                ;
                                ;  Begin loop to examine all segments
                                ;
025014  005002                          CLR     R2              ;Say we haven't found any segment yet
025016  012703  000516'                 MOV     #OSTABL,R3      ;Point to entry for 1st segment
025022  005763  000000          1$:     TST     OS$SIZ(R3)      ;Is this segment to be loaded?
025026  001415                          BEQ     2$              ;Br if not
025030  005763  000002                  TST     OS$FLG(R3)      ;Is this segment already over TSINIT?
025034  001012                          BNE     2$              ;Br if yes
025036  026305  000000                  CMP     OS$SIZ(R3),R5   ;Will this segment fit?
025042  101007                          BHI     2$              ;Br if not
025044  005702                          TST     R2              ;Have we found any other seg yet?
025046  001404                          BEQ     3$              ;Br if not
025050  026362  000000  000000          CMP     OS$SIZ(R3),OS$SIZ(R2) ;Is new seg larger than old?
025056  101401                          BLOS    2$              ;Br if not
025060  010302                  3$:     MOV     R3,R2           ;Remember largest segment
025062  062703  000006          2$:     ADD     #OS$$SZ,R3      ;Point to entry for next segment
025066  020337  000744'                 CMP     R3,OSLAST       ;Have we checked all segments?
025072  103753                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
025074  000241                          CLC                     ;Assume we found a segment
025076  005702                          TST     R2              ;Did we find a segment that will fit?
025100  001001                          BNE     9$              ;Br if yes
025102  000261                          SEC                     ;Signal failure on return
025104  012603                  9$:     MOV     (SP)+,R3
025106  000207                          RETURN
```

```
                                        .SBTTL  GETOVL -- Load system overlay into high memory
                                ;-----------------------------------------------------------------------
                                ;  GETOVL is called to load a system overlay into high memory.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to overlay table entry for segment in OSTABL.
                                ;   R5 = 64-byte physical memory block number where seg is to be loaded.
                                ;
                                ;  Outputs:
                                ;   R5 = Update 64-byte physical memory block pointer for next segment.
                                ;
025110                          GETOVL:
                                ;
                                ;  Allocate space for the overlay segment
                                ;
025110  166205  000000                  SUB     OS$SIZ(R2),R5   ;Allocate space for overlay
025114  020527  001600                  CMP     R5,#1600        ;Are we about to run over RT-11?
025120  103405                          BLO     10$             ;Br if yes -- Insufficient memory
                                ;
                                ;  Remember the base address of some key segments
                                ;
025122  004737  004470'                 CALL    KEYSEG          ;Remember address of some segments
                                ;
                                ;  Load the segment
                                ;
025126  004737  025154'                 CALL    LODOVL          ;Load the segment
                                ;
                                ;  Finished
                                ;
025132  000207                          RETURN
                                ;
                                ;  Error: Memory overflow
                                ;
025134                          10$:    .PRINT  #TSXHD
025142                                  .PRINT  #TSXSIZ
025150  000137  004102'                 JMP     INISTP
```

```
                                                   .SBTTL  LODOVL -- Read and relocate system overlay
                                         ;-----------------------------------------------------------------------
                                         ;  LODOVL is called to load a system overlay region into memory.
                                         ;
                                         ;  Inputs:
                                         ;   R2 = Pointer to OSTABL entry for segment being loaded.
                                         ;   R5 = 64-byte physical memory block number where segment is to be loaded.
                                         ;
 025154  010146                          LODOVL: MOV     R1,-(SP)
 025156  010246                                  MOV     R2,-(SP)
 025160  010346                                  MOV     R3,-(SP)
 025162  010446                                  MOV     R4,-(SP)
 025164  010546                                  MOV     R5,-(SP)

                                         ;
                                         ;  Get info about size of the overlay and position within SAV file
                                         ;
 025166  016201  000004                          MOV     OS$OVL(R2),R1    ;Get pointer to linker-built segment entry
 025172  016103  000000G                         MOV     O.SIZ(R1),R3     ;Get size of overlay segment (# words)
 025176  016137  000000G 000142'                 MOV     O.BLK(R1),FILBLK;Get block in SAV file where segment starts
 025204  010302                                  MOV     R3,R2            ;Get total number of words in segment
 025206  062702  000377                          ADD     #255.,R2         ;round to the nearest number of blocks
 025212  000302                                  SWAB    R2               ;Divide by 256. words per segment
 025214  042702  177400                          BIC     #177400,R2       ;kill sign extension bits
 025220  010561  000000G                         MOV     R5,O.PAR(R1)     ;Remember where segment is being loaded
                                         ;
                                         ;  Read next block of overlay segment into low-memory buffer
                                         ;
 025224  013704  000152'                 10$:    MOV     WRKBUF,R4        ;Point to work buffer
 025230                                          .READW  #AREA,#17,R4,#256.,FILBLK       ;read a block
 025266  103466                                  BCS     22$              ;read error occured
                                         ;
                                         ;  Move from low buffer to high position in memory
                                         ;
 025270  012701  000000G                         MOV     #VPAR5,R1        ;get the virtual address of the mapped region
 025274  012700  000400                          MOV     #256.,R0         ;obtain the number of words to move
 025300  020300                                  CMP     R3,R0            ;Do we need to move as many as 256 words?
 025302  103001                                  BHIS    2$               ;Br if yes
 025304  010300                                  MOV     R3,R0            ;Get number of words to move for last block
 025306  160003                          2$:     SUB     R0,R3            ;Get number of words left after this move
 025310                                          DISABL                   ;** Disable interrupts **
 025316  013746  000000G                         MOV     @#KPAR5,-(SP)    ;save the contents of the mapping register
 025322  010537  000000G                         MOV     R5,@#KPAR5       ;change the mapping register
 025326  052737  000000G 000000G                 BIS     #MMENBL,@#SROMMR;enable memory management
 025334  105737  000000G                         TSTB    MEM256           ;Does machine have at least 256Kb of memory?
 025340  001403                                  BEQ     11$              ;Br if not
 025342  052737  000000G 000000G                 BIS     #EMMAP,@#SR3MMR  ;enable extended memory addressing
 025350  012421                          11$:    MOV     (R4)+,(R1)+      ;move into high memory
 025352  077002                                  SOB     R0,11$
 025354  105737  000000G                         TSTB    MEM256           ;Does this machine have at least 256Kb?
 025360  001403                                  BEQ     12$              ;Br if not
 025362  042737  000000G 000000G                 BIC     #EMMAP,@#SR3MMR  ;disable extended memory management
 025370  042737  000000G 000000G 12$:            BIC     #MMENBL,@#SROMMR;disable memory management
 025376  012637  000000G                         MOV     (SP)+,@#KPAR5    ;restore the mapping register
 025402                                          ENABL                    ;** Enable interrupts **
 025410  062705  000010                          ADD     #10,R5           ;advance 64-byte block # by 512-bytes
 025414  005237  000142'                         INC     FILBLK           ;increment file block #
```

```
025420  005302                          DEC     R2              ;More to be copied?
025422  001402                          BEQ     5$              ;Br if not
025424  000137  025224'                 JMP     10$             ;Read and copy rest of mapped segment
                                ;
                                ;  Finished loading the segment
                                ;
025430  012605                  5$:     MOV     (SP)+,R5
025432  012604                          MOV     (SP)+,R4
025434  012603                          MOV     (SP)+,R3
025436  012602                          MOV     (SP)+,R2
025440  012601                          MOV     (SP)+,R1
025442  000207                          RETURN
                                ;
                                ;  Error occurred on read
                                ;
025444                          22$:    .PRINT  #TSXHD          ;Print heading
025452                                  .PRINT  #RDERR          ;Read error
025460  000137  004102'                 JMP     INISTP          ;Abort initialization
```

```
                                        .SBTTL  GETSRT -- Load any shared run-time systems
                                ;-------------------------------------------------------------------------
                                ;       GETSRT is called to load into memory a shared run-time system.
                                ;       Shared run-time systems are loaded into the top of memory.
                                ;
                                ;       Inputs:
                                ;        R1 = Pointer to shared run-time descriptor block.
                                ;        R5 = 64-byte block number of top of free memory.
                                ;
                                ;       Outputs:
                                ;        R5 = New top of memory block number
                                ;
025464  010146                  GETSRT: MOV     R1,-(SP)
025466  010246                          MOV     R2,-(SP)
025470  010346                          MOV     R3,-(SP)
025472  010446                          MOV     R4,-(SP)
                                ;
                                ;       See if this is a dummy run-time entry to allow for patching
                                ;
025474  021127  000000G                 CMP     (R1),#DMYDEV    ;Dummy run-time entry?
025500  001540                          BEQ     7$              ;Br if yes
                                ;
                                ;       Try to open a channel to run-time file
                                ;
025502                                  .LOOKUP #AREA,#1,R1     ;OPEN CHANNEL TO RUN-TIME FILE
025520  103010                          BCC     8$              ;BR IF OPEN WAS SUCCESSFUL
                                ;
                                ;       Cannot open shared run-time file.
                                ;       See if he wants to abort or continue.
                                ;
025522  105737  000000G                 TSTB    VINABT          ;ABORT OR CONTINUE
025526  001132                          BNE     9$              ;BR IF ABORT WANTED
025530  005061  000000G                 CLR     RT$NAM(R1)      ;Mark run-time as not-available
025534  005061  000002G                 CLR     RT$NAM+2(R1)
025540  000520                          BR      7$              ;GO LOAD NEXT RUN-TIME SYSTEM
                                ;
                                ;       Set up information about position of run-time in physical memory
                                ;
025542  116102  000000G         8$:     MOVB    RT$SKP(R1),R2   ;GET # BLOCKS TO SKIP AT FRONT OF RUN-TIME
025546  042702  177400                  BIC     #^C377,R2       ;CLEAR SIGN EXTENSION
025552  160200                          SUB     R2,R0           ;GET # BLOCKS TO READ (LOOKUP SET R0 W  SIZE)
025554  010561  000000G                 MOV     R5,RT$TOP(R1)   ;SET 64-BYTE BLOCK # ABOVE TOP OF RUN-TIME
025560  010003                          MOV     R0,R3           ;GET # 512-BYTE BLOCKS IN RUN-TIME
025562  072027  000003                  ASH     #3,R0           ;CONVERT TO # 64-BYTE BLOCKS
025566  160005                          SUB     R0,R5           ;CALCULATE BASE 64-BYTE BLOCK # OF RUN-TIME
025570  020527  001600                  CMP     R5,#1600        ;ARE WE ABOUT TO RUN OVER RT-11?
025574  103530                          BLO     11$             ;BR IF YES
025576  010561  000000G                 MOV     R5,RT$BAS(R1)   ;SET BASE 64-BYTE BLOCK # OF RUN-TIME
                                ;
                                ;       Read run-time system into memory and position in high-memory
                                ;
025602  010546                          MOV     R5,-(SP)        ;Save address of bottom of run-time
025604  013704  000152'         4$:     MOV     WRKBUF,R4       ;Point to work buffer
025610                                  .READW  #AREA,#1,R4,#256.,R2 ;READ A BLOCK OF RUN-TIME FILE
                                ;       Use memory management to access high-memory area.
025644  012701  000000G                 MOV     #VPAR6,R1       ;GET VIRTUAL ADDRESS OF PAR6 ADDRESS REGION
025650  010537  000000G                 MOV     R5,@#UPAR6      ;SET USER-MODE PAR6 MAP OFFSET VALUE
```

```
025654  012737  077406  000000G                 MOV     #077406,@#UPDR6 ;SET PDR TO ALLOW FULL ACCESS TO PAGE
025662  052737  000000G 000000G                 BIS     #UPMODE,@#PSW   ;SET PREVIOUS-MODE = USER FOR MTPD ACCESS
025670  012700  000400                          MOV     #256.,R0        ;GET # WORDS TO MOVE
025674                                          DISABL                  ;** Disable interrupts **
025702  052737  000000G 000000G                 BIS     #MMENBL,@#SR0MMR;enable memory management
025710  105737  000000G                         TSTB    MEM256          ;DOES THIS MACHINE HAVE AT LEAST 256KB?
025714  001403                                  BEQ     3$              ;BR IF NOT
025716  052737  000000G 000000G                 BIS     #EMMAP,@#SR3MMR ;enable extended memory addressing
025724  012446                          3$:     MOV     (R4)+,-(SP)     ;TRANSFER DATA FROM BUFFER TO HIGH MEMORY
025726  106621                                  MTPD    (R1)+
025730  077003                                  SOB     R0,3$
025732  105737  000000G                         TSTB    MEM256          ;DOES THIS MACHINE HAVE AT LEAST 256KB?
025736  001403                                  BEQ     31$             ;BR IF NOT
025740  042737  000000G 000000G                 BIC     #EMMAP,@#SR3MMR ;DISABLE EXTENDED MEMORY MANAGEMENT
025746  042737  000000G 000000G 31$:    BIC     #MMENBL,@#SR0MMR;DISABLE MEMORY MANAGEMENT
025754                                          ENABL                   ;** Enable interrupts **
025762  062705  000010                          ADD     #10,R5          ;ADVANCE 64-BYTE BLOCK # BY 512-BYTES
025766  005202                                  INC     R2              ;INC FILE BLOCK #
025770  077373                                  SOB     R3,4$           ;READ AND COPY REST OF FILE
                                        ;
                                        ;  Finished loading the run-time system.
                                        ;
025772  012605                                  MOV     (SP)+,R5
025774                                          .CLOSE  #1
                                        ;
                                        ;  Finished
                                        ;
026002  012604                          7$:     MOV     (SP)+,R4
026004  012603                                  MOV     (SP)+,R3
026006  012602                                  MOV     (SP)+,R2
026010  012601                                  MOV     (SP)+,R1
026012  000207                                  RETURN
                                        ;
                                        ;  Error -- Cannot find run-time system file
                                        ;
026014                                  9$:     .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
026022                                          .PRINT  #COSRT          ;PRINT ERROR MESSAGE
026030  012702  000004                          MOV     #4,R2           ;PRINT 4 RAD50 VALUES
026034  012100                          10$:    MOV     (R1)+,R0        ;GET PART OF NAME
026036  004737  030020'                         CALL    PRTR50          ;PRINT RAD50 VALUE
026042  077204                                  SOB     R2,10$
026044                                          .PRINT  #CRLF           ;END LINE
026052  000137  004102'                         JMP     INISTP          ;ABORT INITIALIZATION
                                        ;
                                        ;  Error -- Insufficient memory space to load run-time systems
                                        ;
026056                                  11$:    .PRINT  #TSXHD          ;PRINT HEADING
026064                                          .PRINT  #SRTOVF         ;PRINT ERROR MESSAGE
026072  000137  004102'                         JMP     INISTP          ;ABORT INITIALIZATION
```

```
                                          .SBTTL  CSHBUF -- Allocate space for data cache tables
                                  ;-------------------------------------------------------------------------
                                  ;  Allocate space for data cache blocks and control tables.
                                  ;
                                  ;  Inputs:
                                  ;   R4 = 64-byte block number of base of free memory.
                                  ;   R5 = 64-byte block number of top of free memory.
                                  ;
                                  ;  Outputs:
                                  ;   R5 = Updated 64-byte block number of top of free memory.
                                  ;
 026076  010246                   CSHBUF: MOV     R2,-(SP)
 026100  010346                           MOV     R3,-(SP)
                                  ;
                                  ;  See if data caching is wanted
                                  ;
 026102  013737  000000G 000000G          MOV     CSHALC,VCSHNB    ;Set # blocks in use = # blocks allocated
 026110  013702  000000G                  MOV     CSHALC,R2        ;Did used request data caching?
 026114  001464                           BEQ     9$               ;Br if not
                                  ;
                                  ;  Calculate number of 64-byte blocks needed for each cache control table
                                  ;
 026116  062702  000037                   ADD     #31.,R2          ;Bound up to 32 word block
 026122  072227  177773                   ASH     #-5.,R2          ;Convert to # 64-byte blocks
                                  ;
                                  ;  Compute total space that will be used by all cache data
                                  ;
 026126  013703  000000G                  MOV     CSHALC,R3        ;Get # blocks in cache
 026132  072327  000003                   ASH     #3,R3            ;Get # 64-byte blks used by cache data buffers
 026136  012700  000010                   MOV     #8.,R0           ;Get number of cache control tables
 026142  060203                   1$:     ADD     R2,R3            ;Accumulate total space needed
 026144  077002                           SOB     R0,1$
 026146  010337  000000G                  MOV     R3,CSHSIZ        ;Save total space used by cache data
                                  ;
                                  ;  See if there is enough memory space available for the specified cache
                                  ;
 026152  010500                           MOV     R5,R0            ;Get top of memory address
 026154  160400                           SUB     R4,R0            ;Compute # free 64-byte blocks
 026156  020300                           CMP     R3,R0            ;Is there enough total space?
 026160  103045                           BHIS    10$              ;Br if not
                                  ;
                                  ;  Allocate space for cache data buffers
                                  ;
 026162  013700  000000G                  MOV     CSHALC,R0        ;Get # blocks in data cache
 026166  072027  000003                   ASH     #3,R0            ;Get # 64-byte blocks needed for allocation
 026172  160005                           SUB     R0,R5            ;Allocate space for cache data buffers
 026174  010537  000000G                  MOV     R5,CSHBFP        ;Save pointer to base of buffer area
                                  ;
                                  ;  Allocate space for each control table
                                  ;
 026200  160205                           SUB     R2,R5            ;Allocate space for table
 026202  010537  000000G                  MOV     R5,CA$BLK        ;Block number associated with entry
 026206  160205                           SUB     R2,R5            ;Allocate space for table
 026210  010537  000000G                  MOV     R5,CA$DVU        ;Device and unit number
 026214  160205                           SUB     R2,R5            ;Allocate space for table
 026216  010537  000000G                  MOV     R5,CA$WCT        ;Number of words
 026222  160205                           SUB     R2,R5            ;Allocate space for table
 026224  010537  000000G                MOV     R5,CA$UFL       ;LRU chain forward link
 026230  160205                         SUB     R2,R5           ;Allocate space for table
 026232  010537  000000G                MOV     R5,CA$UBL       ;LRU chain backward link
 026236  160205                         SUB     R2,R5           ;Allocate space for table
 026240  010537  000000G                MOV     R5,CA$HFL       ;Hash chain forward link
 026244  160205                         SUB     R2,R5           ;Allocate space for table
 026246  010537  000000G                MOV     R5,CA$HBL       ;Hash chain backward link
 026252  160205                         SUB     R2,R5           ;Allocate space for table
 026254  010537  000000G                MOV     R5,CA$HSH       ;Hash chain list heads
 026260  020527  001600                 CMP     R5,#1600        ;Did we run over RT-11?
 026264  101403                         BLOS    10$             ;Br if yes
                                ;
                                ;  Finished
                                ;
 026266  012603                 9$:     MOV     (SP)+,R3
 026270  012602                         MOV     (SP)+,R2
 026272  000207                         RETURN
                                ;
                                ;  Insufficent memory space available for cache data
                                ;
 026274                         10$:    .PRINT  #TSXHD          ;Print heading
 026302                                 .PRINT  #CSHOVF         ;Overlow message
 026310  000137  004102'                JMP     INISTP          ;Abort the initialization
```

```
                                        .IF     EQ,PROCID       ;Don't allow ODT for production PRO version
                                        .SBTTL  GETODT -- Load ODT
                                ;-----------------------------------------------------------------------
                                ;  GETODT is called to load ODT into memory above TSX and transfer control
                                ;  to it.  On return, ODT has been started.
                                ;
                                ;  Inputs:
                                ;   R5 = Address where ODT is to be loaded.
                                ;
                                ;  Outputs:
                                ;   R5 = Address above top of ODT.
                                ;
 026314  010146                 GETODT: MOV     R1,-(SP)
 026316  010246                         MOV     R2,-(SP)
 026320  010346                         MOV     R3,-(SP)
 026322  010446                         MOV     R4,-(SP)
                                ;
                                ;  Try to lookup ODT rel file.
                                ;
 026324                                 .LOOKUP #AREA,#1,#ODTBLK ;LOOKUP ODT REL FILE
 026344  103010                         BCC     1$              ;BR IF FOUND IT
 026346                                 .PRINT  #TSXHD          ;CAN'T FIND ODT
 026354                                 .PRINT  #NOODT
 026362  000137  004102'                JMP     INISTP          ;ABORT INITIALIZATION
                                ;
                                ;  Read first block of ODT file and determine size of ODT.
                                ;
 026366  062705  000310         1$:     ADD     #200.,R5        ;RESERVE SPACE FOR ODT STACK
 026372  010500                         MOV     R5,R0           ;CHECK MEMORY ADDRESS FOR OVERFLOW
 026374  062700  001000                 ADD     #512.,R0
 026400  004737  027630'                CALL    CHKMEM
 026404                                 .READW  #AREA,#1,R5,#256.,#0
 026440  103002                         BCC     2$              ;Br if no read error
 026442  000137  027024'                JMP     ODTRDX          ;Read error
 026446  016502  000052         2$:     MOV     RSZ(R5),R2      ;GET SIZE OF ODT
 026452  010203                         MOV     R2,R3
 026454  060503                         ADD     R5,R3           ;GET ADDRESS ABOVE TOP OF ODT
 026456  010300                         MOV     R3,R0           ;CHECK MEMORY ADDRESS FOR OVERFLOW
 026460  004737  027630'                CALL    CHKMEM
 026464  000241                         CLC
 026466  006002                         ROR     R2              ;GET # WORDS IN ODT
                                ; Get starting address of ODT
 026470  016500  000040                 MOV     STA(R5),R0      ;GET OFFSET TO START ADDRESS
 026474  162700  001000                 SUB     #ODTBAS,R0      ;CALCULATE ABSOLUTE STARTING ADDRESS
 026500  060500                         ADD     R5,R0
 026502  010037  000234'                MOV     R0,ODTSTA       ;THIS IS REAL STARTING ADDRESS
 026506  016501  000062                 MOV     RBD(R5),R1      ;GET # OF BLOCK WITH RELOCATION INFO
                                ;
                                ;  Read in ODT rel file image.
                                ;
 026512                                 .READW  #AREA,#1,R5,R2,#1
 026546  103526                         BCS     ODTRDX          ;BR IF READ ERROR
                                ;
                                ;  Relocate addresses in ODT.
                                ;  R5 = Address of base of ODT;  R3 = Address above top of ODT.
                                ;  R1 = Block number in rel file of start of relocation info.
                                ;
 026550  010337  000036'           RELFIL: MOV     R3,ODTTOP         ;SAVE ADDRESS ABOVE TOP OF ODT
 026554  010337  000230'                   MOV     R3,RLBF
                                   .IF     NE,PROCID                 ;Only if PRO protection code is included
                                           TSTB    PROFLG            ;Are we running on a Pro?
                                           BNE     1$                ;Br if yes
                                   .ENDC   ;NE,PROCID
 026560  013737  000152' 000230'           MOV     WRKBUF,RLBF       ;READ RELOCATION INFO HERE
 026566  013737  000230' 000232'   1$:     MOV     RLBF,RLBFND
 026574  062737  002000  000232'           ADD     #1024.,RLBFND     ;GET ADDRESS OF END OF BUFFER AREA
 026602  010504                            MOV     R5,R4             ;GET BASE ADDRESS OF ODT
 026604  162704  001000                    SUB     #ODTBAS,R4        ;SUBTRACT LINK BASE ADDRESS
                                   ;  Read in relocation address list.
 026610                            4$:     .READW  #AREA,#1,RLBF,#512.,R1
 026646  103003                            BCC     7$                ;BR IF NO READ ERROR
 026650  105737  000052                    TSTB    @#52              ;END OF FILE IS OK
 026654  001063                            BNE     ODTRDX            ;BR IF READ ERROR
                                   ;  Relocate some addresses in ODT.
 026656  013702  000230'           7$:     MOV     RLBF,R2           ;POINT TO RELOCATION INFO
 026662  012203                    3$:     MOV     (R2)+,R3          ;GET ADDRESS OF LOCATION TO RELOCATE
 026664  020327  177776                    CMP     R3,#-2            ;TIME TO STOP?
 026670  001416                            BEQ     9$                ;BR IF FINISHED
 026672  012200                            MOV     (R2)+,R0          ;GET VALUE TO RELOCATE
 026674  006303                            ASL     R3                ;CVT TO BYTE ADDRESS
 026676  103002                            BCC     5$                ;BR IF ADDITIVE RELOCATION
 026700  160400                            SUB     R4,R0             ;RELOCATE THE ADDRESS
 026702  000401                            BR      6$
 026704  060400                    5$:     ADD     R4,R0             ;RELOCATE THE ADDRESS
 026706  060503                    6$:     ADD     R5,R3             ;GET LOCATION WHERE WORD GOES
 026710  010013                            MOV     R0,@R3            ;STORE RELOCATED ADDRESS
 026712  020237  000232'                   CMP     R2,RLBFND         ;TIME TO READ NEXT BUFFER FULL?
 026716  103761                            BLO     3$                ;BR IF NOT
 026720  062701  000002                    ADD     #2,R1             ;ADVANCE BLOCK #
 026724  000731                            BR      4$                ;GO READ NEXT BUFFER FULL
                                   ;
                                   ;  Finished relocation.
                                   ;  Close ODT rel file.
                                   ;
 026726                            9$:     .CLOSE  #1
                                   ;
                                   ;  Direct interrupts to 60 and 64 to an RTI instruction
                                   ;
                                   ;       MOV     #DORTI,@#60       ;Catch interrupt 60
                                   ;       MOV     #DORTI,@#64       ;Catch interrupt 64
                                   ;
                                   ;  Load registers with the following values for initial entry to ODT:
                                   ;   R0 = Base of TSINIT
                                   ;   R1 = Important breakpoint (^R) in TSX
                                   ;   R2 = Base of TSGEN
                                   ;   R3 = Base of TSEXEC
                                   ;   R4 = Base of TSEMT
                                   ;   R5 = Return address to start execution
                                   ;   0(SP) = Address of mapsys routine
                                   ;   2(SP) = Address of  sysmap cell
                                   ;
 026734  012700  000000'                   MOV     #TSINIT,R0
 026740  012701  000000G                   MOV     #BRKPT,R1
 026744  012702  000000G                   MOV     #TSGEN,R2
```

```
026750  012703  000000G                 MOV     #TSEXEC,R3
026754  012704  000000G                 MOV     #TSEMT,R4
026760  012746  000000G                 MOV     #SYSMAP,-(SP)   ;PASS ADDRESS OF SYSMAP CELL TO ODT
026764  012746  000000G                 MOV     #MAPSYS,-(SP)   ;PASS ADDRESS OF MAPSYS ROUTINE
026770  012705  027000'                 MOV     #10$,R5         ;ADDRESS FOR ODT TO RETURN TO
                                ;
                                ;   Enter ODT
                                ;
026774  000177  151234                  JMP     @ODTSTA         ;JUMP TO START OF ODT
                                ;
                                ;   Return from ODT.
                                ;   Continue initialization of TSX.
                                ;
027000  013737  000014  000000G 10$:    MOV     @#14,ODTTRP     ;SAVE ODT BREAKPOINT ENTRY ADDRESS
027006  013705  000036'                 MOV     ODTTOP,R5       ;ADDRESS ABOVE TOP OF ODT
027012  012604                          MOV     (SP)+,R4
027014  012603                          MOV     (SP)+,R3
027016  012602                          MOV     (SP)+,R2
027020  012601                          MOV     (SP)+,R1
027022  000207                          RETURN
                                ;
                                ;   Error while reading ODT rel file.
                                ;
027024                          ODTRDX: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
027032                                  .PRINT  #ODTRDM
027040  000137  004102'                 JMP     INISTP          ;ABORT INITIALIZATION
                                ;
                                ;   RTI instruction to disable interrupts
                                ;
027044  000002                  DORTI:  RTI
                                        .ENDC   ;EQ,PROCID
```

```
                                             .SBTTL  OPNCHN -- Open a TSX-Plus channel
                                     ;--------------------------------------------------------------------------------
                                     ;  OPNCHN is called to set up information in a TSX-Plus channel block
                                     ;  to make it look as if the channel has been opened to a specified
                                     ;  device with a .ENTER.
                                     ;
                                     ;  Inputs:
                                     ;   R0 = Address of channel block to be opened.
                                     ;   R2 = Rad50 device name.
                                     ;
                                     ;  Outputs:
                                     ;   C-flag set ==> Cannot open the device.
                                     ;
027046  010146                       OPNCHN: MOV     R1,-(SP)
027050  010346                               MOV     R3,-(SP)
027052  010003                               MOV     R0,R3             ;Carry channel block address in R3
                                     ;
                                     ;  Initialize the channel block
                                     ;
027054  010301                               MOV     R3,R1             ;Point to the channel block
027056  012700  000000C                      MOV     #<CHNSIZ/2>,R0    ;Get # words to zero
027062  005021                       2$:     CLR     (R1)+             ;Zero the channel block
027064  077002                               SOB     R0,2$
027066  012763  000000C 000000G              MOV     #<CS$OPN!CS$ENT>,C.CSW(R3) ;Initialize CSW to say chan open
                                     ;
                                     ;  Convert the device name into device # and unit #
                                     ;
027074  010200                               MOV     R2,R0             ;Get the full device name
027076  004737  012704'                      CALL    CVTDVU            ;Convert to dev # and unit #
027102  103411                               BCS     9$                ;Br if we don't recognize the device name
027104  010001                               MOV     R0,R1             ;Get index # and unit #
027106  000301                               SWAB    R1                ;Get unit # to low byte
027110  110163  000000G                      MOVB    R1,C.DEVQ(R3)     ;Set unit # in channel block
027114  042700  000000C                      BIC     #^C<CS$NMX>,R0    ;Clear all but device index number in R0
027120  050063  000000G                      BIS     R0,C.CSW(R3)      ;Store device index # into CSW
                                     ;
                                     ;  Success
                                     ;
027124  000241                               CLC                       ;Signal success on return
                                     ;
                                     ;  Finished
                                     ;
027126  012603                       9$:     MOV     (SP)+,R3
027130  012601                               MOV     (SP)+,R1
027132  000207                               RETURN
```

```
                                        .SBTTL  SETCHN -- Copy RT-11 channel information into TSX system chan
                                ;-------------------------------------------------------------------------------
                                ;  SETCHN is called to set up a TSX system channel block to access a file
                                ;  that has been opened using RT-11.  The device index number is converted
                                ;  from the RT-11 device number to the corresponding TSX device number.
                                ;  Note: the channel must have been opened with a .lookup (not .enter)
                                ;  to use this routine.
                                ;
                                ;  Inputs:
                                ;   Channel # 1 = Open to file of interest.
                                ;   R0 = Address of TSX channel block which is to be set up.
                                ;   R2 = Rad-50 device name.
                                ;
                                ;  Outputs:
                                ;   Channel block pointed to by R0 is set up for future TSX I/O.
                                ;   Channel # 1 is closed.
                                ;
027134  010146                  SETCHN: MOV     R1,-(SP)
027136  010246                          MOV     R2,-(SP)
027140  010346                          MOV     R3,-(SP)
027142  010001                          MOV     R0,R1           ;GET ADDRESS OF TSX CHANNEL BLOCK
                                ;
                                ;  Do .SAVESTATUS to store channel information into TSX channel block.
                                ;
027144                                  .SAVEST #AREA,#1,R1     ;STORE CHANNEL STATUS INTO TSX CHANNEL BLOCK
                                ;
                                ;  Now convert RT-11 device table index number into corresponding TSX
                                ;  device table index number.
                                ;
027162  011103                          MOV     (R1),R3         ;GET CSW FOR CHANNEL
027164  042703  177701                  BIC     #^C76,R3        ;GET RT-11 DEVICE INDEX NUMBER
027170                                  .GVAL   #AREA,#404      ;GET RT-11 OFFSET TO PNAME TABLE
027210  060003                          ADD     R0,R3           ;GET ADDRESS OF NAME OF DEVICE IN PNAME TABLE
027212                                  .GVAL   #AREA,R3        ;GET NAME OF DEVICE FROM RT-11
027230  013703  000000G                 MOV     NUMDEV,R3       ;GET INDEX # FOR LAST TSX DEVICE
027234  020063  000000G         1$:     CMP     R0,PNAME(R3)    ;LOOK FOR DEVICE IN OUR TABLES
027240  001404                          BEQ     2$              ;BR IF FOUND
027242  162703  000002                  SUB     #2,R3           ;CHECK NEXT ENTRY
027246  002372                          BGE     1$              ;BR IF MORE TO CHECK
027250  000407                          BR      MTSXDV          ;VERY STRANGE THAT WE DIDN'T FIND IT
027252  042711  000076          2$:     BIC     #76,(R1)        ;CLEAR OUT RT-11 DEVICE #
027256  050311                          BIS     R3,(R1)         ;STORE TSX DEVICE #
                                ;
                                ;  Finished
                                ;
027260  012603                          MOV     (SP)+,R3
027262  012602                          MOV     (SP)+,R2
027264  012601                          MOV     (SP)+,R1
027266  000207                          RETURN
                                ;
                                ;  Error: Could not locate Rt-11 device number in TSX device table.
                                ;
027270                          MTSXDV: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
027276                                  .PRINT  #REQMIS         ;Missing a required device
027304  010200                          MOV     R2,R0           ;GET RAD50 DEVICE NAME
027306  004737  030020'                 CALL    PRTR50          ;DISPLAY DEVICE NAME
027312                                  .PRINT  #CRLF
```

```
     58 027320  000137  004102'                  JMP     INISTP          ;ABORT INITIALIZATION
```

```
                                        .SBTTL  SETSY  -- Set up information about SY device
                                ;-----------------------------------------------------------------------
                                ;  SETSY is called to set up information about the SY device.
                                ;  It does this by determining what device RT-11 recognizes as SY.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of base of free memory area
                                ;
                                ;  Outputs:
                                ;   SYNAME = RAD50 spec for physical system disk
                                ;   SYINDX = TSX device table index for SY device
                                ;   SYUNIT = SY device unit number
                                ;
027324  010146                  SETSY:  MOV     R1,-(SP)
027326  010246                          MOV     R2,-(SP)
                                ;
                                ;  Set up system device unit number
                                ;
027330                                  .GVAL   #AREA,#274      ;Get system unit # from RT-11 (high byte)
027350  010037  000000G                 MOV     R0,SYUNIT       ;Set system unit number
                                ;
                                ;  Set up system device index number
                                ;
027354                                  .GVAL   #AREA,#364      ;Get RT-11 system device index number
027374  010002                          MOV     R0,R2           ;Save device index number
027376                                  .GVAL   #AREA,#404      ;Get offset within RMON of PNAME table
027416  060002                          ADD     R0,R2           ;Get offset to name of SY device
027420                                  .GVAL   #AREA,R2        ;Get name of RT-11 system device
027436  013701  000000G                 MOV     NUMDEV,R1       ;Get index to last TSX-Plus device entry
027442  020061  000000G         1$:     CMP     R0,PNAME(R1)    ;Search for device in TSX tables
027446  001405                          BEQ     2$              ;Br if found it
027450  162701  000002                  SUB     #2,R1           ;Keep looking if more
027454  002372                          BGE     1$
027456  010002                          MOV     R0,R2           ;Save name of system device
027460  000703                          BR      MTSXDV          ;Missing device error
027462  010137  000000G         2$:     MOV     R1,SYINDX       ;Store index # of TSX-Plus system device
                                ;
                                ;  Set up RAD50 name of SY disk
                                ;
027466  113702  000001G                 MOVB    SYUNIT+1,R2     ;GET SYSTEM UNIT NUMBER
027472  062702  000036                  ADD     #36,R2          ;PUT IN "0" AS 3'RD CHARACTER OF NAME
027476  066102  000000G                 ADD     PNAME(R1),R2    ;ADD DEVICE NAME
027502  010237  000000G                 MOV     R2,SYNAME       ;THIS IS THE FULL SY DISK NAME
                                ;
                                ;  Finished
                                ;
027506  012602                          MOV     (SP)+,R2
027510  012601                          MOV     (SP)+,R1
027512  000207                          RETURN
```

```
                                        .SBTTL  RTFTCH -- Fetch a RT-11 device handler
                                ;------------------------------------------------------------------------------
                                ;  RTFTCH is called to fetch an RT-11 device handler.
                                ;  If the handler is already resident, nothing is done.
                                ;  If the handler will fit in WRKBUF, it is fetched into there.
                                ;  If the handler will not fit in WRKBUF, it is fetched into the top
                                ;  of memory.
                                ;
                                ;  Inputs:
                                ;   R0 = RAD50 device name.
                                ;   R5 = Address of start of free memory.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Fetch was successful.
                                ;   C-flag set     ==> Error on fetch.
                                ;
027514  010046                  RTFTCH: MOV     R0,-(SP)
027516  010246                          MOV     R2,-(SP)
027520  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set the name of the device being fetched
                                ;
027522  010037  000130'                 MOV     R0,FETDEV       ;Set name of device whose handler to fetch
                                ;
                                ;  Do a .DSTAT to get information about the handler
                                ;
027526                                  .DSTAT  #DSTBLK,#FETDEV ;Get information about the device handler
027540  103425                          BCS     9$              ;Br if device not recognized
                                ;
                                ;  Determine if the handler is currently resident
                                ;
027542  005737  000116'                 TST     DSTBLK+4        ;Is the handler resident now?
027546  001021                          BNE     8$              ;Br if yes
                                ;
                                ;  The handler is not currently resident.
                                ;  See if it will fit in WRKBUF.
                                ;
027550  013702  000152'                 MOV     WRKBUF,R2       ;Set address where handler will be loaded
027554  013700  000114'                 MOV     DSTBLK+2,R0     ;Get the size of the handler
027560  020037  000154'                 CMP     R0,WRKSIZ       ;Will handler fit in WRKBUF?
027564  101405                          BLOS    1$              ;Br if handler will fit in WRKBUF
                                ;
                                ;  Handler will not fit in WRKBUF.
                                ;  See if there is room to load it into the top of memory.
                                ;
027566  060500                          ADD     R5,R0           ;Get address above top of area needed
027570  020037  000132'                 CMP     R0,TOPMEM       ;Is there room for handler?
027574  101013                          BHI     10$             ;Br if not
027576  010502                          MOV     R5,R2           ;Set address where handler is to be loaded
                                ;
                                ;  Fetch the handler
                                ;
027600                          1$:     .FETCH  R2,#FETDEV      ;Try to fetch the handler
027610  103401                          BCS     9$              ;Br if error on fetch
                                ;
                                ;  We successfully fetched the handler
                                ;
```

```
027612  000241                  8$:     CLC                             ;Signal success on return
                                ;
                                ;  Finished
                                ;
027614  012605                  9$:     MOV     (SP)+,R5
027616  012602                          MOV     (SP)+,R2
027620  012600                          MOV     (SP)+,R0
027622  000207                          RETURN
                                ;
                                ;  Insufficient memory available to load the handler
                                ;
027624  004737  027650'         10$:    CALL    SIZERR                  ;Generated system is too big -- abort
```

```
                                         .SBTTL  CHKMEM -- Check for memory space overflow
                                 ;-------------------------------------------------------------------------
                                 ;  CHKMEM is called to make sure we have not overflowed the available memory
                                 ;  space while allocating space for TSX.
                                 ;  If a memory overflow occurs, an error message is printed and
                                 ;  the initialization is aborted.
                                 ;
                                 ;  Inputs:
                                 ;   R0 = Address to be tested for validity.
                                 ;
027630  020037  000236'          CHKMEM: CMP     R0,MEMLIM       ;IS THE ADDRESS OK?
027634  103402                           BLO     1$              ;BR IF OK
027636  004737  027650'                  CALL    SIZERR          ;Generated system is too big -- abort
027642  004737  027670'          1$:     CALL    CCATST          ;CHECK FOR ^C ABORT REQUEST
027646  000207                           RETURN

                                 ;-------------------------------------------------------------------------
                                 ;  Generated system is too big.  Abort the initialization.
                                 ;
027650                           SIZERR: .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
027656                                   .PRINT  #TOOBIG         ;PRINT ERROR MESSAGE
027664  000137  004102'                  JMP     INISTP          ;ABORT INITIALIZATION

                                 ;-------------------------------------------------------------------------
                                 ;  Check for control-C and abort initialization if requested.
                                 ;
027670  005737  000040'          CCATST: TST     CCAFLG          ;DID USER REQUEST ^C ABORT?
027674  001402                           BEQ     1$              ;BRANCH IF NOT
027676  000137  004102'                  JMP     INISTP          ;ELSE ABORT INITIALIZATION
027702  000207                   1$:     RETURN
```

```
                                        .SBTTL  PRTOCT -- Print octal value
                                ;-----------------------------------------------------------------------
                                ;  PRTOCT is called to print an octal value without trailing Cr-Lf.
                                ;
                                ;  Inputs:
                                ;   R0 = value to be printed.
                                ;
 027704  010146                 PRTOCT: MOV     R1,-(SP)
 027706  010246                         MOV     R2,-(SP)
 027710  010001                         MOV     R0,R1                   ;GET VALUE TO PRINT
 027712  012702  000006                  MOV     #6,R2                   ;PRINT 6 DIGITS
 027716  005000                          CLR     R0
 027720  073027  000001                  ASHC    #1,R0                   ;GET 1ST OCTAL DIGIT (1 BIT)
 027724  000403                          BR      2$
 027726  005000                 1$:     CLR     R0                      ;INITIALIZE FOR SHIFT
 027730  073027  000003                  ASHC    #3,R0                   ;SHIFT AN OCTAL DIGIT INTO R0
 027734  062700  000060         2$:     ADD     #'0,R0                  ;CONVERT TO ASCII CHARACTER
 027740                                 .TTYOUT                         ;PRINT THE CHARACTER
 027744  077210                          SOB     R2,1$                   ;LOOP AND PRINT MORE DIGITS
 027746  012602                          MOV     (SP)+,R2
 027750  012601                          MOV     (SP)+,R1
 027752  000207                          RETURN
```

```
                                        .SBTTL  PRTDEC -- Print decimal value
                                ;-------------------------------------------------------------------------------
                                ;  PRTDEC is called to print a decimal value with leading zeroes suppressed
                                ;  and with no trailing Cr-Lf.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to be printed
                                ;
027754  010146                  PRTDEC: MOV     R1,-(SP)
027756  005046                          CLR     -(SP)            ;NULL ON STACK TO STOP US
                                ;
                                ;  Convert value to ascii digit string and stack the digits.
                                ;
027760  010001                          MOV     R0,R1            ;GET VALUE TO BE CONVERTED
027762  005000                  1$:     CLR     R0               ;SET HIGH-ORDER PART OF VALUE TO 0
027764  071027  000012                  DIV     #10.,R0          ;DIVIDE R0-R1 BY 10.
027770  062701  000060                  ADD     #'0,R1           ;CONVERT REMAINDER TO ASCII DIGIT
027774  010146                          MOV     R1,-(SP)         ;AND STACK THE DIGIT
027776  010001                          MOV     R0,R1            ;GET QUOTIENT
030000  001370                          BNE     1$               ;BR IF MORE DIGITS TO CONVERT
                                ;
                                ;  Finished conversion.  Print result.
                                ;
030002  012600                  2$:     MOV     (SP)+,R0         ;GET A DIGIT FROM THE STACK
030004  001403                          BEQ     3$               ;BR IF REACHED END
030006                                  .TTYOUT                  ;PRINT THE DIGIT
030012  000773                          BR      2$               ;PRINT MORE
                                ;
                                ;  Finished
                                ;
030014  012601                  3$:     MOV     (SP)+,R1
030016  000207                          RETURN
```

```
                                    .SBTTL  PRTR50 -- Print Rad-50 value
                            ;------------------------------------------------------------------------
                            ;  PRTR50 is called to print a Rad-50 value.
                            ;
                            ;  Inputs:
                            ;   R0 = value to be printed.
                            ;
030020  010146              PRTR50: MOV     R1,-(SP)
030022  010246                      MOV     R2,-(SP)
                            ;
                            ;  Convert value to ascii string and stack the characters.
                            ;
030024  012702  000003              MOV     #3,R2               ;GET # CHARS TO CVT
030030  010001                      MOV     R0,R1               ;GET VALUE TO BE CONVERTED
030032  005000              1$:     CLR     R0                  ;CLEAR HIGH-ORDER VALUE
030034  071027  000050              DIV     #50,R0              ;DIVIDE R0-R1 BY 50
030040  116101  003347'             MOVB    R50CHR(R1),R1       ;CONVERT REMAINDER TO ASCII CHARACTER
030044  010146                      MOV     R1,-(SP)            ;STACK THE CHARACTER
030046  010001                      MOV     R0,R1               ;GET QUOTIENT
030050  077210                      SOB     R2,1$               ;BR IF MORE CHARS TO CONVERT
                            ;
                            ;  Finished conversion.  Print the result.
                            ;
030052  012702  000003              MOV     #3,R2               ;GET # CHARS TO PRINT
030056  012600              2$:     MOV     (SP)+,R0            ;GET NEXT CHARACTER
030060                              .TTYOUT                     ;PRINT THE CHARACTER
030064  077204                      SOB     R2,2$               ;LOOP IF MORE CHARS TO PRINT
                            ;
                            ;  Finished
                            ;
030066  012602                      MOV     (SP)+,R2
030070  012601                      MOV     (SP)+,R1
030072  000207                      RETURN
                            ;-----------------------------------------------------------------
                            ;  Define top of TSINIT
                            ;
030074                      INITOP:
                            ;
```

```
.IF     NE,PROCID       ;Only assemble for protected Pro 350 version
;
;  The following startup code is only included for the Pro version.
;  It is loaded here and executed very early during initialization
;  and subsequently overwritten by I/O buffers.
;
        .SBTTL  INSCHK -- Installation validation subroutines for Pro-350
        .MCALL  .PRINT
;
;  Reserve an arg block area for encryption calls
;
EDARGB: .WORD   -32.                    ;# OF BYTES TO BE DECRYPTED (EDMTH3)
EDADDR: .WORD   DSKBUF                  ;POINTER TO BUFFER TO BE DECRYPTED
;
;  Recover license number and decrypt disk image of Pro ID to intermed. state
;
INSCHK: MOV     R1,-(SP)                ;SAVE REGISTERS
        MOV     R2,-(SP)
        MOV     R3,-(SP)
        MOV     R4,-(SP)
        MOV     R5,-(SP)
        MOV     (PC)+,R0                ;DECRYPT LICENSE NUMBER
        .RAD50  /SCB/                   ;WITH THIS CODE
        XOR     R0,LICNUM               ;BY XORING IT
        MOV     LICNUM,TSXSIT           ;MOVE LICENCE NUMBER TO TSGEN CELL
        MOV     #EDARGB,R0              ;POINT TO ENC/DEC ARG BLOCK (PRESET)
        CALL    EDMTH3                  ;DECRYPT TO INTERMED STATE
;
;  Copy Pro ID ROM low bytes into memory, and encrypt 1 step
;
IDADDR  = 173600        ;ADDRESS OF START OF PRO 350 ID ROM
        MOV     #IDADDR,R1              ;GET POINTER TO PRO ID ROM
        MOV     #ROMBUF,R2              ;POINTER TO COPY OF HARDWARE ID
        MOV     R2,EDADDR               ;SAVE ADDRESS FOR ENCRYPTION
        NEG     EDARGB                  ;MAKE +32. FOR ENCRYPTION
        MOV     EDARGB,R0               ;ALSO USE AS LOOP COUNTER
3$:     MOVB    (R1)+,(R2)+             ;GET NEXT LOW BYTE
        INC     R1                      ;SKIP ID ROM HIGH BYTES
        SOB     R0,3$                   ;REPEAT THROUGH 32 BYTE ROM
        MOV     #EDARGB,R0              ;POINT TO ENCRYPTION ARG BLOCK
        CALL    EDMTH2                  ;PERFORM METHOD 2 ENCRYPTION
;
;  Have intermediate state of both hardware and disk copies of Pro ID
;  in memory. Verify them against each other and correct memory image
;  of SCHED at the same time.
;
        MOV     #ROMBUF,R1              ;POINT TO HARDWARE COPY OF ID
        MOV     #DSKBUF,R2              ;POINT TO DISK COPY OF ID
        MOV     #SCHED,R4               ;POINT TO CODE TO BE CORRECTED
        MOV     EDARGB,R3               ;INIT LOOP COUNTER
        CALL    GETLIC                  ;USE LIC # AS SEED FOR EDPRNW, IN R0
4$:     CMPB    (R1)+,(R2)+             ;VERIFY ID'S ARE THE SAME
        BNE     5$                      ;ABORT IF NO MATCH ON ANY BYTE
        CALL    EDPRNW                  ;RANDOMIZE R0 FOR XOR (LIC# INIT SEED)
        MOV     @R4,R5                  ;GET ENCRYPTED CODE
        XOR     R0,R5                   ;RESTORE FUNCTIONAL CODE
        MOV     R5,(R4)+                ;PUT DECRYPTED CODE BACK IN MEMORY
        SOB     R3,4$                   ;REPEAT THROUGH ID TESTS
        MOV     (SP)+,R5                ;RESTORE REGISTERS
        MOV     (SP)+,R4
        MOV     (SP)+,R3
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        RETURN                          ;ID CHECKS AND CODE DECRYPTED
;
5$:     .PRINT  #TSXHD                  ;?TSX-F
        .PRINT  #NOTLIC                 ;NOT LICENSED FOR THIS MACHINE
        JMP     INISTP                  ;ID'S DON'T MATCH, ABORT INIT.
;
        .NLIST  BEX
NOTLIC: .ASCIZ  /This copy of TSX-Plus not licensed for use on this machine./
        .LIST   BEX
        .EVEN
;
;   Subroutine to recover incremental license number. Assume it has been
;   decrypted already by XORing with .RAD50 /SCB/.
;
GETLIC: MOV     LICNUM,R0               ;RETRIEVE DECRYPTED LIC # INTO R0
        RETURN
;
;   Reserve room for both disk and hardware copies of the Pro ID number
;   and for the incremental license number
;
DSKBUF: .BLKB   32.                     ;DISK IMAGE OF PRO ID
LICNUM: .WORD   0                       ;INCREMENTAL LICENSE NUMBER
ROMBUF: .BLKB   32.                     ;COPY OF ROM ID LOW BYTES
```

```
        .SBTTL  EDEXPL -- Comments on encryption methods
;
;  Encryption and decryption methods used here depend heavily on
;  pseudo-random numbers generated by the linear congrutential method.
;  See Hull and Dobell, SIAM Review, 4, 230, 1962.
;
;  For the linear congruence relation:
;
;    X(I) == ( A * X(I-1) + C ) MOD M
;
;    X(I) is in the range 0 to M-1
;
;  The sequence has full period M, provided that:
;    1) C is relatively prime to M
;    2) If p is a prime factor of M, A MOD p == 1
;    3) If 4 is a factor of M, A MOD 4 == 1
;
;  In the special case where M is a power of 2, these rules simplify to
;    1) C must be odd
;    2) A MOD 4 == 1
;
        .SBTTL  EDMTH2 -- Encryption method 2 (XOR with PRN high bytes)
;
;  Using the license number as the initial seed, mask out the low 3 bits,
;  add 1 and call the PRN generator this many times to form the seed.
;  XOR each byte in the input buffer with the high byte of the next PRN
;  and replace the result in the input buffer. Decryption is accomplished
;  by a second application of the same process.
;
;  Inputs:
;       R0      Points to an arg block of the form:
;               R0 ---> buff_siz              ;word holding byte length of buffer
;                       buff_addr             ;address of buffer to be encrypted
;  Outputs:
;       R0      Randomized
;       input buffer encrypted
;
EDMTH2:
        MOV     R1,-(SP)                ;Save registers
        MOV     R2,-(SP)
        MOV     R3,-(SP)
;
;  Fetch byte count, buffer pointer and initialize PRN seed
;
        MOV     (R0)+,R3                ;Fetch byte count of input buffer
        MOV     (R0),R1                 ;Fetch pointer to input buffer
        CALL    GETLIC                  ;Use license number as initial PRN seed
        MOV     R0,R2                   ;Copy license number to form repeat count
        BIC     #^C7,R2                 ;No more than 8 repeats
        INC     R2                      ;Make sure there is at least one
2$:     CALL    EDPRNW                  ;Get a new PRN
        SOB     R2,2$                   ;Advance the seed between 1 and 8 times
;
;  Now sweep the buffer, XORing each byte with the high PRN byte
;
1$:     CALL    EDPRNW                  ;With seed in R0, get next random number
        MOVB    (R1),R2                 ;Get next input byte
        SWAB    R0              ;Reverse PRN high and low bytes
        XOR     R0,R2           ;Encrypt the byte
        SWAB    R0              ;Restore PRN high and low bytes for next seed
        MOVB    R2,(R1)+        ;Save encrypted bytes back into input buffer
        SOB     R3,1$           ;Repeat for entire input buffer
;
        MOV     (SP)+,R3        ;Restore registers
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        RETURN
        .SBTTL  EDMTH3 -- Encryption/decryption meth 3 (swap bytes&shift bits)
;
;  Using a prn of repeat length same as input string length, select prn
;  numbered bytes from the input string, combine them into a word,
;  shift the combined bytes a random number of bits, recombine the shifted
;  bits and set the confused bytes back into the prn selected string bytes.
;  Sign of the byte count indicates: + = encryption; - = decryption.
;  If the byte count is 0 or 1, no encryption occurs. If the byte count is
;  odd, then one random selected byte will not be encrypted.
;
;  Inputs:
;       R0      Points to an arg block of the form:
;               R0 --->  buff_siz       ;Word holding byte length of buffer.
                                        ;Note that buffer must be 512 or less
                                        ;in length. Flag encryption by using
                                        ;positive byte count ( 2 to 512. ).
                                        ;Flag decryption by using negative
                                        ;byte count (-2 to -512. ).
                                        ;
;                        buff_addr      ;Address of buffer to be encrypted
;  Outputs:
;       R0      Randomized
;       Input buffer encrypted
;
EDMTH3: MOV     R1,-(SP)                ;Save registers
        MOV     R2,-(SP)
        MOV     R3,-(SP)
        MOV     R4,-(SP)
        MOV     R5,-(SP)
        MOV     (R0)+,R3                ;Save the string length
        MOV     @R0,-(SP)               ;And save the input buffer pointer
;  Initialize prn generator of desired length
        MOV     R3,R0                   ;Recover string length
        BGE     1$                      ;Branch if encryption
        NEG     R0                      ;If decryption, get real repeat
        INC     R3                      ;If neg, correct for ASR round down
1$:     CALL    INPRNM                  ;Set up for desired repeat length
;  Start encryption loop through string
        ASR     R3                      ;Repeat for 1/2 the string length
        CALL    GETLIC                  ;Get lic. num. for initial seed in R0
        CALL    EDPRNM                  ;Seed PRN generator (-adjacent pairs)
2$:     TST     R3                      ;Less than 2 bytes left?
        BEQ     9$                      ;Quit if so (odd len -> 1 byte unch.)
        CLR     R4                      ;Clean out shifting registers
        CLR     R5
        MOV     @SP,R1                  ;Retrieve buffer pointer
        MOV     R1,R2                   ;And second copy
;  Select first random byte
        CALL    EDPRNM                  ;Randomize in range 0 - <strlen-1>
        ADD     R0,R1                   ;Point to first random byte of pair
;  Select second random byte
        CALL    EDPRNM                  ;Randomize again
        ADD     R0,R2                   ;Point to next random byte
;  Use part of PRNM as semi-random shift amount
        MOV     R0,-(SP)                ;Save EDPRNM seed for later
        BIC     #^C6,R0                 ;Get a semi-random shift amount
;  Select encryption or decryption
```

```
                TST     R3                      ;Positive for encryption
                BMI     3$                      ;Branch if decrypting
        ;  Do this part for encryption
                BISB    @R1,R4                  ;Get first byte without sign extend
                SWAB    R4                      ;And put it in the high byte
                BISB    @R2,R4                  ;Combine it with first byte
                CLC                             ;Always do at least one shift
                ROR     R4                      ;Shift once
                ROR     R5                      ;Get low bit into r5
                NEG     R0                      ;Right shifts for encryption
                DEC     R3                      ;Reduce count of pairs remaining
                BR      4$                      ;Skip decryption stuff
        ;  Do this part for decryption
        3$:     BISB    @R1,R5                  ;Get first byte without sign extend
                SWAB    R5                      ;And put it in the high byte
                BISB    @R2,R5                  ;Combine it with the first byte
                CLC                             ;Always do at least one shift
                ROL     R5                      ;Shift once
                ROL     R4                      ;Get high bit into R4
                INC     R3                      ;Reduce count of pairs remaining
        ;  Shift and recombine the (en|de)crypted bytes
        4$:     ASHC    R0,R4                   ;Shift combined bytes 0,2,4 or 6 more
                BIS     R5,R4                   ;Recombine bytes
                MOV     (SP)+,R0                ;Recover EDPRNM seed
        ;  Now put encrypted bytes back into input string
                MOVB    R4,@R2                  ;Store low byte at second byte place
                SWAB    R4                      ;Get high byte
                MOVB    R4,@R1                  ;Store high byte at first byte place
                BR      2$                      ;Repeat through string
        ;  Done, restore registers and return
        9$:     MOV     (SP)+,R0                ;Just pop saved buffer address
                MOV     (SP)+,R5                ;Restore registers
                MOV     (SP)+,R4
                MOV     (SP)+,R3
                MOV     (SP)+,R2
                MOV     (SP)+,R1
                RETURN
        .SBTTL  EDPRNW -- Pseudo random number generator with MOD 2^16
;
;  Linear congruential pseudo-random number generator with maximum repeat
;  length of 65536 (2^16). cf. Hull and Dobell and Knuth, vol 2.
;
;  Inputs:
;       R0      Seed value
;
;  Outputs:
;       R0      New PRN, should be used for next seed
;
EDPRNW:
        MOV     R4,-(SP)                ;Save registers
        MOV     R5,-(SP)
        MOV     R0,R4                   ;Get seed to be multiplied
        MOV     (PC)+,R0                ;Fetch multiplier
EDPRNA: .WORD   104375                  ;Multiplier, can be replaced
        MUL     R0,R4                   ;Multiply by A
        ADD     (PC)+,R5                ;Add C
EDPRNC: .WORD   012705                  ;Additive, can be replaced
        MOV     R5,R0                   ;Return result mod 65536. as PRN
        MOV     (SP)+,R5                ;Restore registers
        MOV     (SP)+,R4
        RETURN
        .SBTTL  INPRNM -- Initialize PRN generator with repeat range M
;
;  Using the Hull and Dobell rules, determine acceptable values for
;  A and C to get a repeat range of M.
;
;  Outputs:
;       EDMULA  Set with first acceptable multiplier
;       EDADDC  Set with first acceptable additive factor
;       EDMODM  Set with desired repeat length
;
INPRNM:
        MOV     #32.,EDMODM     ;Get repeat length to cover Pro ID
        MOV     #5,EDMULA       ;Use first valid A
        MOV     #3,EDADDC       ;And first valid C
        RETURN
```

```
                                        .SBTTL  EDPRNM -- Generate pseudo-random number in specified range M
                                ;
                                ;   *********************************************************************
                                ;   * INPRNM MUST BE CALLED BEFORE FIRST SEED IS PASSED TO THIS ROUTINE!!!! *
                                ;   *********************************************************************
                                ;
                                ;   Using linear congruential method (cf. Hull and Dobell), generate
                                ;   pseudo-random number using seed passed in R0. Return new PRN in R0.
                                ;
                                ;   Inputs:
                                ;       R0      Seed value, must be in range 0 to M (EDMODM)
                                ;
                                ;   Outputs:
                                ;       R0      New pseudo-random number, should be used for next seed
                                ;
                                EDPRNM:
                                        MOV     R4,-(SP)        ;Save R4 and R5
                                        MOV     R5,-(SP)
                                        CMP     R0,EDMODM       ;Is seed in range 0 to EDMODM?
                                        BLO     1$              ;Branch and proceed if so
                                        MOV     R0,R5           ;Set up to divide it by EDMODM
                                        CLR     R4              ;Set up for divide
                                        DIV     EDMODM,R4       ;Divide it
                                        MOV     R5,R0           ;And use remainder as seed
                                1$:     MOV     R0,R4           ;Get current seed ready to be multiplied
                                        MUL     (PC)+,R4        ;Multiply by chosen A
                                EDMULA: .WORD   25173.          ;Replace at run-time with 5
                                        ADD     (PC)+,R5        ;Add in C
                                EDADDC: .WORD   13849.          ;Replace at run-time with 3
                                        CLR     R4              ;Clear high word for division
                                        DIV     (PC)+,R4        ;Perform mod M
                                EDMODM: .WORD   256.            ;Replace at run-time with 32.
                                        MOV     R5,R0           ;Return remainder
                                        MOV     (SP)+,R5        ;Restore R4 and R5
                                        MOV     (SP)+,R4
                                        RETURN
                                ;
                                .IFF    ;NE,PROCID              ;Assemble if protection code not included
 030074                         DSKBUF:                         ;Define dummy DSKBUF global symbol
                                .ENDC   ;NE,PROCID
                                ;
                                ;   Address of real top of TSINIT, including PRO init code
                                ;
 030074                         PROITP:
 000000                                 .CSECT  TSXEND
                000001                  .END
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 11518 Words  ( 45 Pages)
Size of core pool: 17920 Words  ( 70 Pages)
Operating  system: RT-11

Elapsed time: 00:02:16.32
```

```
DK:TSINIT,LP:TSINIT=DK:TSINIT/C/N:SYM
```

```
$8BIT     1-130     41-72
$DEAD     1-123     16-18     16-65     29-69     30-27     39-20
$FORM     1-126     41-69
$HARD     1-155     16-20     16-67     24-30     29-27
$MEMSZ    1-142     45-111*
$NOIN     1-113     16-71
$OVRH     1-137     67-25
$PHONE    1-110     16-69
$SXON     1-90      41-25
$TAB      1-126     41-66
...V1     8-29      8-29      8-45      8-46      25-19     25-23     25-31     25-32     25-33     25-36     25-54     25-55
          25-61     25-63     25-67     25-78     25-182    26-220    26-225    26-226    26-242    26-242    29-84     29-85
          29-91     29-92     29-96     29-97     29-101    30-50     30-50     30-50     30-57     30-57     30-58     30-58
          30-58     30-62     30-62     30-62     30-70     30-70     30-72     30-72     30-84     30-95     30-96     30-102
          30-103    30-108    30-111    30-116    30-119    31-30     31-30     31-30     31-42     31-42     31-43     31-43
          31-43     31-47     31-47     31-47     31-55     31-55     31-56     31-56     31-68     31-77     31-78     31-84
          31-85     32-60     32-61     32-85     32-85     32-85     32-89     32-89     32-90     32-90     32-90     32-94
          32-94     32-94     32-99     32-99     32-101    32-101    32-110    32-131    32-132    32-138    32-139    40-21
          40-21     40-21     40-27     40-27     40-62     40-62     40-62     40-75     40-76     40-81     40-82     42-22
          42-22     42-22     42-29     42-29     42-31     42-31     42-68     42-68     42-68     42-72     42-73     42-73
          42-73     42-78     42-78     42-78     42-82     42-82     42-83     42-83     42-84     42-84     42-84     42-91
          42-101    42-102    42-108    42-109    43-43     43-43     43-43     43-47     43-48     43-48     43-48     43-52
          43-52     43-52     43-55     43-55     43-60     43-65     43-65     43-66     43-76     43-77     43-83     43-84
          45-160    45-161    47-76     47-77     51-77     51-77     53-26     53-26     53-26     53-38     53-38     55-25
          55-25     57-21     57-21     57-21     57-30     57-30     57-64     57-64     57-113    57-113    59-8      59-9
          59-12     60-36     60-36     60-50     60-50     61-35     61-35     67-17     67-17     67-51     67-52     69-24
          69-24     69-52     69-53     69-58     69-59     72-34     72-35     73-30     73-30     73-73     73-74     74-25
          74-25     74-25     74-54     74-54     74-81     74-81     74-93     74-94     74-99     74-104    74-105    75-78
          75-79     76-20     76-20     76-20     76-22     76-23     76-32     76-32     76-51     76-51     76-70     76-70
          76-95     76-95     76-138    76-139    78-25     78-25     78-32     78-34     78-53     78-54     78-57     79-19
          79-24     79-26     79-28     80-27     80-53     81-20     81-21     82-18     83-26     84-26
...V2     8-29      8-29      8-29#     8-29#     26-242    26-242    26-242#   26-242#   30-50     30-50     30-50#    30-50#
          30-57     30-57#    30-58     30-58     30-58#    30-58#    30-62     30-62     30-62#    30-62#    30-70     30-70
          30-70#    30-70#    30-72     30-72#    31-30     31-30     31-30#    31-30#    31-42     31-42#    31-43     31-43
          31-43#    31-43#    31-47     31-47     31-47#    31-47#    31-55     31-55     31-55#    31-55#    31-56     31-56#
          32-85     32-85     32-85#    32-85#    32-89     32-89#    32-90     32-90     32-90#    32-90#    32-94     32-94
          32-94#    32-94#    32-99     32-99     32-99#    32-99#    32-101    32-101#   40-21     40-21     40-21#    40-21#
          40-27     40-27     40-27#    40-27#    40-62     40-62     40-62#    40-62#    42-22     42-22     42-22#    42-22#
          42-29     42-29     42-29#    42-29#    42-31     42-31     42-31#    42-31#    42-68     42-68     42-68#    42-68#
          42-72     42-72#    42-73     42-73     42-73#    42-73#    42-78     42-78     42-78#    42-78#    42-82     42-82
          42-82#    42-82#    42-83     42-83#    42-84     42-84     42-84#    42-84#    43-43     43-43     43-43#    43-43#
          43-47     43-47#    43-48     43-48     43-48#    43-48#    43-52     43-52     43-52#    43-52#    43-55     43-55
          43-55#    43-55#    43-60     43-60     43-60#    43-60#    43-65     43-65#    51-77     51-77#    53-26     53-26
          53-26#    53-26#    53-38     53-38     53-38#    53-38#    55-25     55-25     55-25#    55-25#    57-21     57-21
          57-21#    57-21#    57-30     57-30     57-30#    57-30#    57-64     57-64     57-64#    57-64#    57-113    57-113#
          60-36     60-36     60-36#    60-36#    60-50     60-50     60-50#    60-50#    61-35     61-35     61-35#    61-35#
          67-17     67-17     67-17#    67-17#    69-24     69-24     69-24#    69-24#    73-30     73-30     73-30#    73-30#
          74-25     74-25     74-25#    74-25#    74-54     74-54     74-54#    74-54#    74-81     74-81#    76-20     76-20
          76-20#    76-20#    76-32     76-32     76-32#    76-32#    76-51     76-51     76-51#    76-51#    76-70     76-70
          76-70#    76-70#    76-95     76-95#    78-25     78-25     78-25#    78-25#
AHEND     1-107     50-21
ALBFX     26-160    39-12#
ALCHRB    25-142    28-14#
ALCOVL    67-34     69-15#
ALCSLO    25-103    38-12#
ALCWRK    25-138    27-12#
```

```
ALOCBF  25-102    37-14#
AREA     1-183#    8-29     25-19     25-23     25-61     25-63     25-78     25-182    26-220    26-242    30-50     30-58
        30-62     30-70     31-30     31-43     31-47     31-55     32-85     32-90     32-94     32-99     40-21     40-27
        40-62     42-22     42-29     42-31     42-68     42-73     42-78     42-82     42-84     43-43     43-48     43-52
        43-55     43-60     53-26     53-38     55-25     57-21     57-30     57-64     60-36     60-50     61-35     67-17
        69-24     73-30     74-25     74-54     76-20     76-32     76-51     76-70     78-25     78-32     78-34     79-19
        79-24     79-26     79-28
AUTHAN   1-107    50-21     50-23
BADDEV  30-104    30-116#   31-86     32-140    42-110    43-85
BADLIN   6-6#     29-85
BADOPN   6-11#    30-96     30-103
BADVEC  29-47     29-49     29-51     29-91#
BASMAP   1-101    47-52*
BDLMSG   6-8#     29-97
BDSPOP   6-14#    32-61
BDVMSG   6-7#     29-92
BOSF     6-15#    32-132    32-139
BRKPT    1-102    76-113
BYTES    6-34#    25-36
C.CSW    1-143    33-30*    77-24*    77-35*
C.DEVQ   1-143    33-31*    77-33*
C.SBLK   1-143    33-23*
C1DEVX   1-143    33-27     41-112*
CA$BLK   1-148    75-52*
CA$DVU   1-148    75-54*
CA$HBL   1-149    75-64*
CA$HFL   1-149    75-62*
CA$HSH   1-149    75-66*
CA$UBL   1-149    75-60*
CA$UFL   1-149    75-58*
CA$WCT   1-148    75-56*
CC$$SZ   1-150    37-74
CCAFLG   1-187#   25-23     81-27
CCATST  26-246    81-14     81-27#
CCBHD    1-150    37-73*
CCLNAM   1-194#   40-62     40-65
CCLSAV   1-128    40-64
CD$$SZ   1-86     37-64
CDX$DH   1-142    37-104
CDX$DL   1-151    16-22
CDX$DZ   1-150    16-50     16-75     20-37
CDX$VH   1-153    16-52     16-77     20-40     37-106
CFHMSG   6-18#    30-116    53-31     57-23
CHAIN    1-117    25-85
CHKCLD  32-40     34-13#
CHKMEM  26-148    37-140    55-20     76-31     76-39     81-11#
CHNSIZ   1-122    77-21
CKLIN   26-34#
CL$EPN   1-127    41-58*
CL$EPP   1-127    41-52*
CL$EPS   1-126    41-31*    41-51*
CL$LEN   1-157    41-79*
CL$LIX   1-156    24-19*    41-23
CL$OPT   1-127    41-75*
CL$ORA   1-125    41-43*
CL$ORB   1-125    41-36*
```

```
Cross reference table (CREF V05.05)


CL$ORE    1-125    41-46*
CL$ORG    1-125    41-38*
CL$ORP    1-125    41-37*
CL$ORS    1-127    41-44*
CL$STA    1-127    41-83*
CL$VER    2-13#    25-227
CLDEVX    1-143    33-29     41-94*
CLEOFS    1-126    41-53     41-54
CLHEAD    1-143    41-99     41-117
CLINCP    1-155    24-21
CLINIT   26-72     41-11#
CLK100    1-188#    7-165    25-14*
CLKRTI    1-107     7-71
CLKVEC    1-158     7-71*     7-72*     7-165*   25-14     26-77*
CLORSZ    1-112    41-39
CLOTIR    1-155    24-20
CLSIZE    1-143    41-100
CLSTS     1-118    41-96     41-114
CLTOTL    1-113    26-70     33-18     41-18     41-101    41-108    70-156
CLVERS    1-111    25-224    25-227*
CO$8BT    1-130    41-74
CO$DEF    1-126    41-62
CO$FF     1-126    41-71
CO$TAB    1-126    41-68
CONFG2    1-105    25-165*   25-166*   25-169*   26-78
CONFIG    1-104    25-163*   26-5
CONSPC    6-13#    30-108
COSRT     6-36#    74-94
CRLF      6-9#     29-96     29-101    30-111    30-119    59-12     74-99     78-57
CS$ENT    1-145    77-24
CS$NMX    1-88     77-34
CS$OPN    1-145    77-24
CSHALC    1-122    37-71     70-168    75-17     75-18     75-28     75-44
CSHBAS    1-105     9-39     11-37*
CSHBFP    1-148    75-47*
CSHBUF   26-205    75-12#
CSHDEV    1-86     37-63*
CSHDVN    1-86     37-67*
CSHOVF    6-41#    75-79
CSHSIZ    1-148    75-33*
CSHVEC    1-106     9-42
CURDEV    1-205#   60-54*    61-40     63-62
CURNAM    1-206#   52-25*    57-20*    59-10
CVTDVU   32-37     35-14#    77-29
CW$50H    1-104    26-5
CW$BTH    1-103    25-161
CW$ESP    1-113    25-166
CW$FB     1-103    25-162
CW$FGJ    1-103    25-162
CW$GDH    1-103    25-161
CW$LGS    1-103    25-161
CW$PRO    1-107    25-116    26-78
CW$QBS    1-115    25-169
CW$USR    1-104    25-162
CW$XM     1-104    25-162
CXTALC   26-92     46-12#
```

```
CXTBAS   1-112    46-24
CXTBUF   1-119    37-134*
CXTPAG   1-87     30-40     40-41     46-45*    48-31
CXTPDR   1-111    46-38*
CXTRMN   1-112    46-23*    46-24*
CXTSIZ   1-111    46-16
CXTWDS   1-111    46-30*    46-42
DATIMH   1-105    26-29*
DATIML   1-105    26-30*
DC$$SZ   1-133    39-64
DCCSIZ   1-85     70-99
DEVSIZ   1-93     26-50*    26-65*    41-98*    41-116*   54-50*
DEVVEC   7-139    13-6#
DFJMEM   1-134    48-50*
DHBFSZ   1-109    37-110    37-112
DHINIT  16-56     21-8#
DHLPRM  16-82     17-8#
DHOINT   1-136    20-42
DI$CL    1-118
DI$DU    1-156     3-24
DI$LD    1-118    26-63
DI$MU    1-156     3-26
DI$PI    1-157
DI$TT    1-119    26-48
DI$XL    1-156     3-25
DM$CSR   1-152    17-13     17-15*    17-16*    21-27
DM$LSR   1-152    17-17*
DMYDEV   1-95     32-34     32-57     50-25     74-20
DOHNLC  55-46     57-108    60-13#
DORTI   76-144#
DS$ABT   1-125
DS$DIR   1-139    26-63     54-55
DS$NRD   1-130    54-57
DS$SFN   1-140    26-63
DS$VSZ   1-139    26-63
DSKBUF   1-58     90-39#
DSTBLK   1-195#   80-27     80-32     80-39
DTLX     1-104    40-52*
DTYPE    1-129    50-27
DV$$SZ   4-6#     52-58
DV$FLG   4-5#     52-56
DV$NAM   4-4#     52-54
DVFLAG   1-139    26-49*    26-64*    41-97*    41-115*   51-43     54-31*    54-59*    65-31*    65-42
DVFLBS   4-13#    52-53
DVFLND   4-42#    52-59
DVSTAT   1-89     26-48*    26-63*    41-96*    41-114*   54-51*    54-55     54-57
DW$$SZ   1-115    70-144
DX$DMA   1-139     4-16      4-18      4-19      4-20      4-21      4-22      4-23      4-26      4-28      4-31      4-32
         4-33      4-34      4-37      4-38     52-65
DX$EBA   1-92      4-39     26-64     52-67
DX$IBH   1-97      4-18      4-31      4-32      4-33      4-34     51-48
DX$MAP   1-146    51-52     65-31     65-46
DX$MPH   1-97      4-14      4-15      4-16      4-18      4-24      4-25      4-27      4-29      4-30      4-31      4-32
         4-33      4-35      4-36      4-38      4-40      4-41     51-46
DX$NCA   1-158     4-39     41-97     41-115    54-59
DX$NHM   1-97      4-17      4-19      4-23      4-26      4-34      4-39     51-44
```

```
DX$NMT    1-130     41-97     41-115    54-59
DX$NRD    1-131     41-97     41-115
DX$NST    1-138      4-23      4-34
DZINIT   16-58      19-8#
DZOINT    1-136     20-36
EMMAP     1-99       7-123    45-74     45-118    57-79     57-84     62-36     62-55     73-47     73-52     74-65     74-71
EMTENT    1-108      7-89
ENTVEC    9-35       9-43      9-51     10-11#
ERHINS    6-22#     53-97
ERHMSG    6-19#     53-40     55-27     57-66
ERHNDV    6-20#     53-53
ERP      29-86      29-94#
ERRLOG    1-135     58-22
EXCBUF    1-84      27-17
EXTLSI    1-251#    45-128    51-54     65-24
FC$$SZ    1-135     70-70
FC$LBN    1-116     39-61
FETDEV    1-199#    80-23*    80-27     80-53
FF$$SZ    1-116     39-56
FILBLK    1-204#    57-50*    57-64     57-92*    73-20*    73-30     73-57*
FMEMHI    1-201#    26-167    45-112*   45-123    47-65*    57-107*   57-109    64-27     64-34*    64-35
FMEMLO    1-202#    26-166*   47-37     47-62     64-28
FNDHRB    1-58      63-35#
FORCEO   32-26      36-8#
FORK      1-89      58-15
FPTRAP    1-118      7-95
FQ$$SZ    1-95      37-37
FREIOQ    1-118     37-18*
FREPGS    1-119     48-22*
FRKGEN    1-95      37-34     37-37
FRKINI    1-95      37-36*
FSTDL     1-94      39-22
FSTIOL    1-155     24-11     37-102    38-24
FW$$SZ    1-116     39-58
GENTOP    1-157     25-27
GETHNH   26-201     56-13#
GETHNL   26-96      50-14#
GETMAP   26-171     68-12#
GETODT   25-149     76-13#
GETOVL   68-34      72-12#
GETSRT   26-179     74-13#
GTBYT     1-102     58-25
GTLIN     1-235#    25-51
H.CSR     1-95      53-76
H.DSTS    1-89      53-48     54-51
H.DVSZ    1-93      54-50
H.GEN     1-89      53-63     53-68     57-31
H.INS     1-95      53-88
H.SIZ     1-93      54-49
HANDSK    1-83      54-43*
HANENT    1-89      13-14     26-51*    26-66*    41-99*    41-117*   55-34*    55-35*    57-35*    60-21     60-61
HANMAP   57-101     60-22     60-63     61-41     62-14#
HANNAM    1-236#    53-24*    53-26     57-19*    57-21
HANPAR    1-97      13-20     14-25     14-54     51-60*    51-72*    55-16*    56-23     57-48*    62-26     62-32
HANRCB    1-83      28-22*    63-40
HANRCO    1-83      28-23*    28-24*
```

```
HANSIZ   1-93     41-102*   41-118*   51-64     54-49*    55-18     57-40     57-100
HANUMP  57-103    60-24     60-72     60-79     61-31     62-49#
HANXIT   1-97     14-59
HANXMR   1-58     64-22#
HF$LIN   1-151
HF$MC    1-152    21-21     21-22
HF$RIE   1-151
HF$TIE   1-151
HF$TSB   1-151
HGENFL   1-243#   53-68*    57-31*    58-17     58-20
HIMAP    1-94      7-133    47-56*    48-20
HLERR   51-67     53-32     53-41     53-54     53-66     53-81     53-98     55-28     57-24     57-67     59-5#
HMAP     1-198#   57-49*    57-75     57-91*
HN2BIG   6-29#    51-66
HSGER    6-24#    53-65
HV$$SZ   3-19#    53-56
HV$DMY   3-17#
HV$ID    3-16#    53-49
HV$VER   3-18#    53-51
HVEND    3-27#    53-57
HVTBL    3-23#    53-46
ICONFG   1-247#   45-128*   51-54     65-24
IDSFLG   1-100    45-48*
II$$SZ   1-85     37-57
ILSW2    1-113    16-69     41-65
INDDBL   1-140    42-40*
INDDBS   1-140    42-45*    42-56
INDFIL   1-84     42-62     42-68     42-73     42-78     42-84     42-89     42-91
INDINI  26-117    42-12#
INDNAM   1-238#   42-21*    42-22     42-51
INDOPN   6-42#    42-102    42-109
INDSAV   1-140    42-17*    42-50
INDTSV   1-140    42-88
INICLK  26-5#
INIDEV  26-47#
INIJMP   1-108     7-161
INIOVL   7-13      9-9#
INISTP   7-165#   25-37     25-56     26-227    29-102    30-112    30-120    32-64     32-133    40-77     40-83     45-162
        47-78     59-13     67-53     69-54     69-60     72-36     73-75     74-100    74-106    75-80     76-24     76-140
        78-58     81-22     81-29
INITGO   1-57     25-8#     25-104    25-106
INITOP   1-57      1-208    25-132    25-134    66-52     66-54     84-37#
INMXV    1-109    20-16
INRECV   1-109    16-28
INSCK1  51-25     52-16#
INSCK2  51-31     53-14#
INSTBL   1-85     37-54*
INSTBN   1-85     37-59*
INTEN    1-89     58-16
INTMX1   1-110
INTSND   1-114     7-24*     7-152    25-113*
INTSSZ   1-84     25-122
INTSTK   1-114     7-155    25-123*
INVEC    1-87     16-27     29-38
IOMAP    1-99      1-138     7-126    12-37
IOPAGE   1-99     44-29
```

```
IOQSIZ   1-119    37-19
IOTIMR   1-135    58-19
JCXPGS   1-134    48-32*
JM$$SZ   1-112    37-44
JMP0     1-104     7-78
JSWLOC   1-117    25-85
KEYSEG   9-23     11-10#    72-22
KMNBAS   1-120    40-34
KMNCHN   1-98     40-56
KMNHI    1-96     40-35*
KMNNAM   1-193#   40-21     40-57
KMNPGS   1-98     40-42*    48-26
KMNSTK   1-98     40-44*
KMNSTR   1-98     40-46*
KMNTOP   1-96     40-32*
KPAR0    1-98     44-14
KPAR5    1-136     7-130*   13-9      13-22*    13-62*    14-52     14-55     57-74     57-75*    57-86*    61-25     61-46*
        62-32*    62-57*    73-42     73-43*    73-54*
KPAR6    1-158    60-71*    60-78*    61-26     61-27*    61-45*
KPAR7    1-101    44-29*
KPDR0    1-101    44-16
LCDTYP   1-110    16-22     16-72     37-104    37-106
LCLUNT   1-137    24-13*    24-18     24-26*    41-26*
LDDEVX   1-139    26-56*    26-61*
LDHAND  50-31     50-46     51-21#
LDHB1B   1-111    37-108*
LDHB1P   1-111    37-109*
LDHB2B   1-111    37-111*
LDHNHI  56-29     57-12#
LDHNLO  51-73     55-12#
LDREAD  60-64     61-15#
LDVERS   1-115    25-228    25-230*   25-233*
LHIRBA   1-90     38-30     38-39*
LHIRBB   1-90     38-43*
LHIRBC   1-91     38-54     38-60*    38-61     38-67*
LHIRBE   1-90     38-48*
LHIRBG   1-90     38-45*
LHIRBP   1-90     38-44*
LHIRBS   1-91     38-46*
LINBUF   1-94     39-28*
LINCHK  26-34     29-9#
LINEND   1-96     39-32*
LININI   7-113    16-8#
LINIR    1-155    24-21*    24-29*
LINSIZ   1-94     39-29
LINSPC   1-97     39-30*
LINTRP  29-41     29-43     29-68     29-84#
LINTYP  24-9#     29-26
LMXLN    1-151    16-92     17-16     18-14
LMXNUM   1-87     16-73     17-12     18-12     29-29     41-29
LODINI   8-9#      9-24
LODOVL  72-26     73-9#
LOKBAS   1-106     9-47     11-33*
LOKCSH   1-133    39-71*
LOKMEM   1-101    39-55*
LOKVEC   1-106     9-50
```

```
LOMAP     1-93      7-131    47-50*    48-21
LOTBUF    1-96     39-38*
LOTEND    1-96     39-41*
LOTSIZ    1-96     39-39     39-40     41-42
LOUTIR    1-155    24-20*    24-28*
LOWEND    5-23#    66-39
LOWOVL    5-20#    66-36
LSTDL     1-94     39-24
LSTHL     1-155    16-43     16-64     37-114    38-17
LSTLIN    1-127    29-21
LSTMX     1-117    16-48     37-90
LSTPL     1-87     24-9      30-26     37-100    38-22
LSTSL     1-88     39-46     42-57     43-32
LSW10     1-90     41-25*
LSW3      1-113    16-18     16-20     16-65     16-67     16-71*    24-30*    29-27     29-69*    30-27     39-20
LSW5      1-158
LSW6      1-123
LSWPBK    1-88
LTTPAR    1-106    39-26*
LXCL      1-84     24-27*
MA$SYS    1-113     7-135
MAPALC   26-211    47-27#
MAPPAR    1-97      7-130    47-64*
MAPSIZ    1-99     45-108    45-110
MAPSYS    1-137     7-104    76-118
MAXDEV    1-83     60-59
MAXOVL    5-12#     5-14
MAXSLO    1-91     38-33     38-35
MB$$SZ    1-145    70-106
MEM256    1-133     7-124    45-125*   45-134    57-77     57-82     61-21     61-48     62-34     62-53     73-45     73-50
         74-63     74-69
MEMINI   25-155    44-6#
MEMLIM    1-242#   25-72*    81-11
MEMPAR    1-131     7-93
MEMTST   26-88     45-24#
MF$CM     1-152    21-30
MF$CS     1-152    21-29
MF$LE     1-152    17-17
MF$LIN    1-150    17-15
MH$LPR    1-152
MH$PBR    1-153
MH$SCR    1-151    21-12     21-21*    21-22
MHNSIZ    1-108    57-46*
MI$$SZ    1-147    65-56     65-68
MI$LNK    1-146    39-86     65-71*
MI$SBP    1-146    39-82*
MINCTR    1-120
MIOBHD    1-146    39-80     65-65*
MIODBG    1-128    65-22
MIOFLG    1-146    39-78     65-51*    70-150
MIONWB    1-147    65-57     65-76
MIOWHD    1-147    65-75*
MMENBL    1-93      7-118    45-71     45-119    57-76     57-85     62-33     62-56     73-44     73-53     74-62     74-72
MONFQH    1-112    37-43*
MONVEC    1-120     7-17     28-24     53-92     62-22
MPARO     1-92     49-20
```

```
MPAR16    1-92     49-25
MPARFL    1-92     49-7
MPIVSZ   14-19#    26-104
MPPHY     1-102    58-26
MR$$SZ    1-145    70-109
MSGBAS    1-103    11-21*
MTSXDV   78-40     78-53#    79-35
MU$TXT    1-117    70-114
MUXVEC   19-17     20-9#     21-17     22-17
MVSIZ     1-131    46-25     46-49
MW$$SZ    1-147    65-57     65-78     65-82
MW$LNK    1-147    65-79*
MXCSR     1-87     19-12     19-21*    19-22     29-31     29-72*
MXDTR     1-110    19-26*
MXJADR    1-142    48-43*
MXJMEM    1-134    48-38*
MXLNT     1-118    37-92*
MXLPR     1-113    16-94*
MXRBUF    1-110    19-24     22-27
MXTYPE    1-109    16-50     16-52     20-37     20-40
MXVEC     1-87     20-15     29-33
NDL       1-96     30-25
NDVRCB    1-83     28-18
NEDCHR    1-155    24-28
NEXMSG    6-27#
NFRESB    1-115    32-118*
NFSBLK    1-184#   43-60     43-61     43-62
NIOL      1-122    41-101
NLINES    1-121    39-58
NMSNMB    1-132    37-50
NMXHAN    1-197#   26-101    51-59*
NOCCL     6-17#    40-76
NOCLOK    6-25#    26-226
NOCSRR    6-21#    53-80
NOKMON    6-16#    40-82
NOODT     6-28#    76-23
NOSYDV    6-31#
NOXM     25-61     45-160#
NSCP      1-86     37-130
NSIP      1-85     37-56
NSL       1-96     30-25
NSPLBL    1-114    32-80     32-117    70-79
NSPLDV    1-99     32-22*    32-44*    32-50*    32-75     32-114
NSPLFL    1-114    37-78
NUMCCB    1-94     37-74
NUMCDB    1-85     39-54*
NUMDCD    1-133    39-53*
NUMDEV    1-88     13-57     26-52*    26-59*    26-60     35-31     41-92*    41-93     41-110*   41-111    52-43     54-25*
         54-26     56-34     65-33     65-44     78-35     79-29
NUMFRK    1-95     37-34     37-37
NUMIOQ    1-118    37-19
NUMRDB    1-121     1-149    26-175
NXIVMH    1-84     26-103
NXMMSG    6-26#    45-161
O.ADR     1-136    11-15     66-35     69-30*    70-22
O.BLK     1-136     8-29     69-24     73-20
```

```
Cross reference table (CREF V05.05)


O.PAR    1-136    8-34*   73-25*
O.SIZ    1-136    8-21     8-23    69-34    70-26    73-19
ODTBAS   1-179#  76-44    76-68
ODTBLK   1-237#  76-20
ODTFLG   1-185#  25-89*   25-147
ODTRDM   6-30#   76-139
ODTRDX  76-34    76-52    76-73    76-138#
ODTSTA   1-241#  76-46*   76-123
ODTTOP   1-186#  76-58*   76-129
ODTTRP   1-124   76-128*
OORCLO  70-61    70-156#
OORCSH  70-63    70-168#
OORDBG  70-62    70-162#
OORDMP  70-64    70-174#
OORLOK  70-54    70-95#
OORMIO  70-60    70-150#
OORMSG  70-55    70-104#
OORPLS  70-57    70-121#
OORSLE  70-58    70-136#
OORSP2  70-53    70-89#
OORSPL  70-52    70-76#
OORSWP  70-56    70-130#
OORUSR  70-51    70-69#
OORWIN  70-59    70-142#
OOXFIN  70-182   70-197#
OOXNO   70-77    70-90    70-96    70-105   70-122   70-131   70-137   70-143   70-151   70-157   70-163   70-169
        70-175   70-180#
OOXYES  70-37    70-72    70-85    70-91    70-98    70-100   70-117   70-126   70-132   70-138   70-146   70-152
        70-158   70-164   70-170   70-176   70-186#
OPNCHN  32-48    77-14#
OPNKMN  26-113   40-17#
OPNRSF  26-139   31-11#
OPNSWP  26-135   30-13#
OPTOVL  68-21    69-42    70-15#
OS$$SZ   5-10#    5-14     9-26    66-46    67-36    68-23    68-35    71-31
OS$FLG   5-8#     9-21    66-42*   66-64*   67-33*   68-32    71-22
OS$OVL   5-9#     8-15    11-14    66-34    67-32*   69-19    70-21    73-18
OS$SIZ   5-7#     8-19     9-25    66-32    66-43    66-65    67-35*   68-22*   68-30    71-20    71-24    71-28
        71-28    72-16
OSEND    5-15#
OSLAST   5-16#    9-27    66-47    67-40*   68-24    68-36    71-32
OSTABL   5-14#    5-16     9-20    66-31    67-31    68-20    68-29    71-19
OSZ      1-176#
OTMXV    1-109   20-17
OTRECV   1-109   16-29
OVLBAS   1-203#   9-14    66-30*
OVLBLD  66-23    67-10#
OVLEND  70-35    70-65#
OVLLST  70-31    70-50#
OVLPOS  25-127   66-15#
OVLTRY  66-62    71-14#
OVRADD   1-137   67-27*
PARENL   1-92    49-21
PARSET  26-84    49-7#
PHSOVF   6-35#   47-77
PHYMEM   1-142    7-121   45-107*
```

```
PIINSZ  25-112#   25-120
PISRT    1-57      1-255#
PMCELS   1-132    39-94*
PMPAR    1-132    39-95*
PNAME    1-89     26-47*    26-62*    35-32     41-95*    41-113*   52-45     54-30*    57-19     57-20     78-36     79-30
        79-42
PPTERM   1-57      1-260#
PROASM   1-20      1-32      1-57      1-63      7-29      7-143    12-1      15-1      16-1      26-38     26-186    28-6
        29-1      45-54     45-64     49-1      63-1      65-1
PROBUF   1-58      1-207#   25-119*
PROCID   1-24     25-95     25-146    76-1      76-60     85-1
PROFLG   1-98      7-53     25-118*
PROITP   1-57     90-44#
PRTDEC  25-35     29-100    30-110    83-9#
PRTOCT  29-95     82-8#
PRTR50  30-118    32-63     59-11     74-97     78-56     84-8#
PSW      1-94     45-80*    45-97*    57-73*    57-87*    62-18*    62-65*    73-41*    73-55*    74-59*    74-61*    74-73*
PTBYT    1-101    58-24
PTWRD    1-101    58-23
PVSPBL   1-115    32-115
QBUS     1-138    25-167    45-126
R50C1    1-228#   34-30     41-113
R50C10   1-229#   34-32     34-36
R50C17   1-230#   34-34
R50CHR   6-44#    84-17
R50CL    1-225#   34-17     41-95
R50CLO   1-226#   34-19     34-23
R50CL7   1-227#   34-21
R50CSH   1-219#   11-35
R50LD    1-223#   26-62     52-30
R50LOK   1-217#   11-31
R50LS    1-232#
R50MSG   1-215#   11-19
R50ODT   1-233#   25-87
R50PI    1-224#
R50SY    1-222#
R50TIO   1-220#   11-39
R50TT    1-221#   26-47
R50USR   1-218#   11-27
R50VM    1-231#
R50WIN   1-216#   11-23
RBD      1-178#   76-47
RBR      1-117    16-102
RC$$SZ   1-122    37-28
RDB      1-121    26-177
RDBEND   1-121    26-181
RDERR    6-39#     8-46     67-52     69-53     73-74
RDINT    1-117    16-106
REDUCE   6-33#    25-33
RELFIL  76-58#
RELOC    1-102    58-27
REQMIS   6-10#    78-54
RID      1-177#
RLBF     1-239#   76-59*    76-64*    76-65     76-70     76-75
RLBFND   1-240#   76-65*    76-66*    76-87
RMNPDR   1-113    46-56*
```

```
RMON     1-105     7-17*     7-167*   25-13     25-115    25-159    53-91     53-92*    53-95*    62-22*    62-61*
RPRVEC   1-103    13-48     53-93     60-58
RSFBLK   1-141    31-23     31-30     31-43     31-47     31-63     31-68
RSFERR   6-12#    31-78     31-85
RSR      1-87     16-104    29-37     41-27
RSZ      1-175#   76-35
RT$$SZ   1-123    26-180
RT$BAS   1-122    74-48*
RT$NAM   1-139    74-33*    74-34*
RT$SKP   1-121    74-39
RT$TOP   1-121    74-42*
RT$UPD   2-12#    25-192    25-212
RT$VER   2-11#    25-189    25-211
RT40     2-17#
RT50     2-18#     3-24
RT51     2-19#
RT51B    2-21#     3-25
RT51C    2-22#
RT51X    2-20#
RT52     2-23#
RT53     2-24#
RT54     2-26#     3-26     25-231
RT55     2-27#
RTFTCH  30-20     31-24     32-78     42-63     43-37     80-17#
RTMNVC   1-190#    7-167    25-13*    62-61
RTNKM   45-69     45-155#
RTTRP4   1-189#    7-166    25-12*
RTV$SZ   2-14#    25-195    25-207
RTVDEF   2-25#    25-181    25-211    25-212
RTVEND   1-210     2-28#    25-196    25-203    25-207
RTVER    2-16#    25-188
RTVPTR   1-210#   25-181*   25-203    25-207*   25-218    25-220*   25-226    25-231    53-51
SAVBLK   1-191#
SB$$SZ   1-132    37-50
SCHED    1-88
SCPFHD   1-86     37-129*
SDANAM   1-120    32-30
SDCB     1-128    32-23
SDCBSZ   1-129    32-68
SDCHAN   1-129    32-47     33-23*    33-30*    33-31*
SDDVU    1-91     32-38*
SDNAME   1-129    32-28*    32-57*
SEGCHN   1-141    31-62
SETCHN  30-80     31-64     32-109    40-58     40-66     42-52     42-90     78-18#
SETJSZ  26-215    48-16#
SETLIN  16-49     16-64#
SETMIO  26-143    65-18#
SETMUX  16-48#
SETSY   26-109    79-14#
SETUMP   7-109    12-14#
SETVEC  13-36     13-50     14-21#
SFCB     1-114    37-81*
SFCBFH   1-114
SFCBND   1-114    37-83*
SFCBSZ   1-114    37-80
SG$ELG   1-105    25-172    58-20
```

```
SG$IOT   1-105    25-173    58-17
SG$MMU   1-106    25-173    53-63
SG$MTM   1-106    25-173
SG$MTS   1-106    25-172
SG$PAR   1-106    25-172
SG$TSX   1-142    25-173
SHRRCB   1-121    37-23*
SHRRCN   1-121    37-30*
SIZERR  48-59     80-69     81-13     81-20#
SKPDEV   1-234#   52-35
SLTSIZ   1-117    30-41*
SMRSIZ   1-150    68-15*    68-41     68-43*
SNMSHD   1-132    37-49*
SP$$SZ   1-86     37-130
SPLANM   1-128    32-25
SPLBLK   1-131    32-77     32-85     32-90     32-94     32-108    32-110
SPLCHN   1-131    32-107
SPLCLD  32-42     33-13#
SPLDEV   1-128    32-24     50-42
SPLDVN   1-129    32-70
SPLINI  26-129    32-7#
SPLNB    1-122    70-78
SPLND    1-128    32-7      50-40     70-76     70-89
SPNEED  30-97     30-108#   31-79     42-103    43-78
SROMMR   1-98      7-118*   25-62     45-71*    45-119*   57-76*    57-85*    61-24     61-47*    62-33*    62-56*    73-44*
        73-53*    74-62*    74-72*
SR3FLG   1-99      7-119    45-39*    45-72     45-116
SR3MMR   1-99      7-123*    7-126*   12-37*    45-37     45-74*    45-118*   57-79*    57-84*    61-23     61-50*    62-36*
        62-55*    73-47*    73-52*    74-65*    74-71*
SRTOVF   6-40#    74-105
SRTSIZ   1-150    26-184*
SS       1-117    25-44
SSEND    1-119    25-41
STA      1-173#   76-43
STDVTB  51-38     54-21#
STHNPV  55-42     57-102    58-11#
STK      1-174#
STKLVL   1-84      7-25*     7-157*
SVERR    6-37#
SWDBLK   1-88     30-19     30-50     30-58     30-62     30-79     30-84
SWPCHN   1-88     30-78
SWPJOB   1-116    37-127*
SWPPOS   1-116    37-125*
SYINDX   1-120    79-36*
SYNAME   1-141    26-125    42-21     79-43*
SYSDAT   1-124    26-222*
SYSGEN   1-107    25-174*
SYSMAP   1-137    76-117
SYSUPD   1-156    25-212*
SYSVER   1-156    25-179    25-183*   25-211*
SYTIMH   1-124    25-19     26-220
SYTIML   1-124    26-219    26-223
SYUNIT   1-119    79-20*    79-40
TAKOVR   7-9#     26-250
TIOBAS   1-110     9-32     11-41*
TIOVEC   1-107     9-34
```

```
TK1SEC   1-94     26-10*   26-19*
TK1VAL   1-109    26-13*   26-22*
TK3SVL   1-134    26-12*   26-21*
TK5VAL   1-134    26-11*   26-20*
TOOBIG   6-32#    25-32    81-21
TOPMEM   1-200#   25-68*   80-47
TRCSET  45-34     45-150#  53-73    53-74
TRP10    1-108     7-81
TRP14    1-124     7-83
TRP20    1-108     7-85
TRP24    1-108     7-87
TRP250   1-124     7-97
TRP34    1-108     7-91
TRP4     1-107     7-79
TSEMT    1-92     76-116
TSEXEC   1-102    76-115
TSGEN    1-102    76-114
TSINIT   1-6#      1-57    25-77    25-101   30-70    31-55    32-99    76-112
TSR      1-117    16-99
TSTWRD  45-22#    45-95*   45-98*   45-100
TSXHD    6-5#      8-45    25-31    25-54    26-225   29-84    29-91    30-95    30-102   31-77    31-84    32-60
        32-131    32-138   40-75    40-81    42-101   42-108   43-76    43-83    45-160   47-76    59-8     67-51
        69-52     69-58    72-34    73-73    74-93    74-104   75-78    76-22    76-138   78-53    81-20
TSXPRO   1-33      1-34#    1-73     1-74#
TSXRUN   6-23#    25-55
TSXSAV   1-192#
TSXSIT   1-112
TSXSIZ   6-38#    69-59    72-35
TTINCP   1-154    24-29
UBUSMP   1-138    12-21    45-136*  51-50
UC$$SZ   1-144    43-25
UCLBLK   1-144    43-28*
UCLDAT   1-144    43-36    43-43    43-48    43-52    43-64*   43-66
UCLINI  26-121    43-10#
UCLNAM   1-141    26-125*
UCLOPN   6-43#    43-77    43-84
UDDR0    1-100    45-46
UEXINT   1-131     7-42
UEXRTN   1-130     7-46
UK$$SZ   1-144    43-22
UMODE    1-118    45-80    45-97    45-155
UMRADR   1-138    12-27
UMSYTP   1-119    26-153*
UNIBUS   1-138    45-132
UPAR0    1-101    44-15
UPAR6    1-123    74-57*
UPAR7    1-93     45-79*   45-96*
UPDR0    1-101    44-17
UPDR6    1-123    74-58*
UPMODE   1-122    74-59
US$$SZ   1-144    43-23
USRBAS   1-104    11-29*
VBUSTP   1-120    25-167   45-126   45-132
VCSHNB   1-127    75-17*
VDBFLG   1-92     70-162
VDFMEM   1-134    48-47    48-49
```

```
VF$LIN   1-154    18-13
VF$MR    1-153    22-21     22-22
VF$RE    1-154    18-15
VF$RIE   1-153
VF$SC    1-154
VF$TIE   1-153
VH$CSR   1-153    18-13*    18-14*    22-12     22-21*    22-22
VH$LCR   1-154    18-15*
VH$LPR   1-153
VHIMEM   1-87     30-38     48-35     48-37
VHINIT  16-54     22-8#
VHLPRM  16-87     18-8#
VHOINT   1-154    20-39
VINABT   1-123    29-67     32-58     59-6      74-31
VLDSYS   1-145    26-57     52-32
VMAXMC   1-145    70-104
VMIOBF   1-147    65-55     65-66
VMIOSZ   1-146    39-83
VMLBLK   1-116    39-59
VMSCHR   1-91     70-111
VMXCSH   1-86     37-65
VMXMON   1-112    37-41
VMXMRB   1-148    70-108
VMXMSG   1-145    70-115
VMXSF    1-116    39-51     39-57     70-95
VMXSFC   1-116    39-54     39-62
VMXWIN   1-115    37-25     37-27     70-142
VNCSLO   1-91     38-32
VNCXOF   1-91     38-56
VNCXON   1-91     38-63
VNFCSH   1-135    70-69
VNGR     1-122    37-24
VNUIP    1-85     37-55
VNUMDC   1-85     39-53     39-65     39-72     70-97
VPAR5    1-136    25-69     25-71     47-49     47-53     57-35     57-72     57-99     73-35
VPAR6    1-135    39-27     74-56
VPLAS    1-141    31-18     31-26     31-36     31-47     70-121
VPMSIZ   1-132    39-91
VSLEDT   1-137    70-136
VSWPFL   1-102    26-133    31-16     37-121    70-130
VSWPSL   1-95     30-32     30-34*    30-45     37-123
VSYDMP   1-84     70-174
VU$CL    1-144    43-15
VUCLMC   1-144    43-17
VUXIFL   1-130     7-44      7-51
WINBAS   1-104    11-25*
WRKBUF   1-208#   26-238    26-242    27-20*    40-26     42-28     42-82     43-55     53-38     53-47     53-62     53-75
        54-48     57-29     57-63     60-35     60-50     60-57     67-16     69-23     73-29     74-53     76-64     80-38
WRKSIZ   1-209#   26-236    27-21*    80-40
XMVBAS   1-196#   14-47     14-72*    26-100*
ZCLR     1-110    19-21     19-22
```

```
...CM0   30-84     31-68     32-110    42-91     43-66     80-27     80-53
...CM1    8-29     26-242    30-50     30-62     30-70     31-30     31-47     31-55     32-85     32-94     32-99     40-21
         40-27     40-62     42-22     42-29     42-31     42-68     42-78     42-82     42-84     43-43     43-52     43-55
         43-60     53-26     53-38     55-25     57-21     57-30     57-64     60-36     60-50     61-35     67-17     69-24
         73-30     74-25     74-54     76-20     76-32     76-51     76-70     78-25
...CM2    8-29      8-29      8-29      8-29     25-19     25-23     25-61     25-63     25-78     25-182    26-220    26-242
         26-242    26-242    26-242    30-50     30-50     30-58     30-58     30-62     30-62     30-62     30-70     30-70
         30-70     30-70     31-30     31-30     31-43     31-43     31-47     31-47     31-47     31-55     31-55     31-55
         31-55     32-85     32-85     32-90     32-90     32-94     32-94     32-94     32-99     32-99     32-99     32-99
         40-21     40-21     40-27     40-27     40-27     40-27     40-62     40-62     42-22     42-22     42-29     42-29
         42-29     42-29     42-31     42-31     42-31     42-31     42-68     42-68     42-73     42-73     42-78     42-78
         42-78     42-82     42-82     42-82     42-82     42-84     42-84     43-43     43-43     43-48     43-48     43-52
         43-52     43-52     43-55     43-55     43-55     43-55     43-60     53-26     53-26     53-38     53-38     53-38
         53-38     55-25     55-25     55-25     55-25     57-21     57-21     57-30     57-30     57-30     57-30     57-64
         57-64     57-64     57-64     60-36     60-36     60-36     60-36     60-50     60-50     60-50     60-50     61-35
         61-35     61-35     61-35     67-17     67-17     67-17     67-17     69-24     69-24     69-24     69-24     73-30
         73-30     73-30     73-30     74-25     74-25     74-54     74-54     74-54     74-54     76-20     76-20     76-32
         76-32     76-32     76-32     76-51     76-51     76-51     76-51     76-70     76-70     76-70     76-70     78-25
         78-32     78-34     79-19     79-24     79-26     79-28
...CM3   30-57     30-72     31-42     31-56     32-89     32-101    42-72     42-83     43-47     43-65     51-77     57-113
         74-81     76-95
...CM5    8-29      8-45      8-46     25-19     25-23     25-31     25-32     25-33     25-36     25-54     25-55     25-61
         25-63     25-67     25-78     25-182    26-220    26-225    26-226    26-242    29-84     29-85     29-91     29-92
         29-96     29-97     29-101    30-50     30-58     30-62     30-70     30-84     30-95     30-96     30-102    30-103
         30-108    30-111    30-116    30-119    31-30     31-43     31-47     31-55     31-68     31-77     31-78     31-84
         31-85     32-60     32-61     32-85     32-90     32-94     32-99     32-110    32-131    32-132    32-138    32-139
         40-21     40-27     40-62     40-75     40-76     40-81     40-82     42-22     42-29     42-31     42-68     42-73
         42-78     42-82     42-84     42-91     42-101    42-102    42-108    42-109    43-43     43-48     43-52     43-55
         43-60     43-66     43-76     43-77     43-83     43-84     45-160    45-161    47-76     47-77     53-26     53-38
         55-25     57-21     57-30     57-64     59-8      59-9      59-12     60-36     60-50     61-35     67-17     67-51
         67-52     69-24     69-52     69-53     69-58     69-59     72-34     72-35     73-30     73-73     73-74     74-25
         74-54     74-93     74-94     74-99     74-104    74-105    75-78     75-79     76-20     76-22     76-23     76-32
         76-51     76-70     76-138    76-139    78-25     78-32     78-34     78-53     78-54     78-57     79-19     79-24
         79-26     79-28     80-27     80-53     81-20     81-21     82-18     83-26     84-26
...CM6   25-19     25-23     25-61     25-63     25-78     25-182    26-220    78-32     78-34     79-19     79-24     79-26
         79-28
...CM7    8-29     26-242    30-70     31-55     32-99     40-27     42-29     42-31     42-82     43-55     53-38     55-25
         57-30     57-64     60-36     60-50     61-35     67-17     69-24     73-30     74-54     76-32     76-51     76-70
.CLOSE    1-50#    30-57     30-72     31-42     31-56     32-89     32-101    42-83     43-65     51-77     57-113    74-81
         76-95
.CSTAT    1-53#    43-60
.DATE     1-52#    26-221
.DELET    1-51#    30-58     31-43     32-90     42-73     43-48
.DSTAT    1-52#    80-27
.ENTER    1-49#    30-62     31-47     32-94     42-78     43-52
.EXIT     1-51#     7-168     8-47
.FETCH    1-52#    80-53
.GTIM     1-52#    25-19     26-220
.GVAL     1-49#    25-78     25-182    78-32     78-34     79-19     79-24     79-26     79-28
.HERR     1-51#    25-57
.LOCK     1-52#    25-81
.LOOKU    1-49#    30-50     31-30     32-85     40-21     40-62     42-22     42-68     42-84     43-43     53-26     57-21
         74-25     76-20
.PRINT    1-50#     8-45      8-46     25-31     25-32     25-33     25-36     25-54     25-55     26-225    26-226    29-84
         29-85     29-91     29-92     29-96     29-97     29-101    30-95     30-96     30-102    30-103    30-108    30-111
         30-116    30-119    31-77     31-78     31-84     31-85     32-60     32-61     32-131    32-132    32-138    32-139
```

```
        40-75     40-76     40-81     40-82     42-101    42-102    42-108    42-109    43-76     43-77     43-83     43-84
        45-160    45-161    47-76     47-77     59-8      59-9      59-12     67-51     67-52     69-52     69-53     69-58
        69-59     72-34     72-35     73-73     73-74     74-93     74-94     74-99     74-104    74-105    75-78     75-79
        76-22     76-23     76-138    76-139    78-53     78-54     78-57     81-20     81-21
.PURGE    1-50#   42-72     43-47
.READW    1-49#    8-29     26-242    40-27     42-29     42-31     53-38     55-25     57-30     57-64     60-36     60-50
        61-35     67-17     69-24     73-30     74-54     76-32     76-51     76-70
.RELEA    1-52#   30-84     31-68     32-110    42-91     43-66
.SAVES    1-49#   78-25
.SCCA     1-53#   25-23
.SERR     1-51#   25-50
.SETTO    1-50#   25-67
.TRPSE    1-50#   25-61     25-63
.TTYOU    1-50#   82-18     83-26     84-26
.UNLOC    1-51#   26-231
.WRITW    1-51#   30-70     31-55     32-99     42-82     43-55
DEFFLG    4-8#     4-14      4-15      4-16      4-17      4-18      4-19      4-20      4-21      4-22      4-23      4-24
          4-25     4-26      4-27      4-28      4-29      4-30      4-31      4-32      4-33      4-34      4-35      4-36
          4-37     4-38      4-39      4-40      4-41
DISABL    1-162#  57-73     62-18     73-41     74-61
ENABL     1-166#  57-87     62-65     73-55     74-73
HANVER    3-8#     3-24      3-25      3-26
OVLTBL  70-45#    70-51     70-52     70-53     70-54     70-55     70-56     70-57     70-58     70-59     70-60     70-61
        70-62     70-63     70-64
```