Table of contents

Table of contents

```
      1          000001                    PROASM  =       1                ;Assemble for Professioal
```

```
                                        .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
                                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
                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$$SZ
                        .GLOBL  LSTPL,LMXNUM,MXCSR,MXVEC,RSR,INVEC,VHIMEM,CXTPAG
                        .GLOBL  LSWPBK,LSTSL,SWDBLK,SWPCHN,NUMDEV,CS$NMX,SCHED
                        .GLOBL  H.DSTS,DVSTAT,HANENT,H.GEN,FORK,INTEN,PNAME
                        .GLOBL  $SXON,LSW10,LHIRBB,LHIRBE,LHIRBP,LHIRBG,LHIRBA
                        .GLOBL  LHIRBS,LHIRBC,VNCSLO,VNCXOF,VNCXON,SDDVU,VMSCHR,MAXSLO
                        .GLOBL  MPAR0,MPAR16,PARENL,MPARFL,TSEMT,VDBFLG,DX$EBA
                        .GLOBL  H.SIZ,HANSIZ,H.DVSZ,DEVSIZ,LOMAP,MMENBL,UPAR7
                        .GLOBL  PSW,HIMAP,FSTDL,LSTDL,LINBUF,LINSIZ,NUMCCB,TK1SEC
                        .GLOBL  FRKINI,FRKGEN,NUMFRK,FQ$$SZ,H.CSR,H.INS,VSWPSL,DMYDEV
                        .GLOBL  LINEND,LOTBUF,LOTSIZ,LOTEND,KMNTOP,KMNHI,NSL,NDL
                        .GLOBL  DX$MPH,DX$NHM,DX$IBH,HANPAR,HANXIT,MAPPAR,LINSPC
                        .GLOBL  KMNPGS,KMNSTK,KMNSTR,KMNCHN,SROMMR,KPAR0,PROFLG
                        .GLOBL  EMMAP,IOMAP,SR3MMR,IOPAGE,MAPSIZ,SR3FLG,NSPLDV
                        .GLOBL  UDDR0,IDSFLG
                        .GLOBL  UPAR0,KPDR0,UPDR0,KPAR7,BASMAP,PTWRD,PTBYT,LOKMEM
                        .GLOBL  GTBYT,MPPHY,RELOC,BRKPT,TSGEN,TSEXEC,VSWPFL
                        .GLOBL  CW$GDH,CW$BTH,CW$LGS,CW$FB,CW$FGJ,MSGBAS,RPRVEC
                        .GLOBL  CW$USR,CW$XM,CONFIG,CW$50H,JMP0,DTLX,USRBAS,WINBAS
                        .GLOBL  DATIML,DATIMH,RMON,CONFG2,SG$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.
```

```
                                        ;
000040                                  STA     =       40              ;PROGRAM START ADDRESS
000042                                  STK     =       42              ;INITIAL STACK POINTER
000052                                  RSZ     =       52              ;ROOT SIZE
000056                                  OSZ     =       56              ;OVERLAY SIZE
000060                                  RID     =       60              ;REL FILE ID
000062                                  RBD     =       62              ;DISPLACEMENT TO 1ST REL BLOCK
001000                                  ODTBAS  =       1000            ;BASE ADDRESS ODT WAS LINKED FOR
                                        ;
                                        ;  Data areas
                                        ;
000000                                  AREA:   .BLKW   8.
000020  000000  000000  000000  NFSBLK: .WORD   0,0,0,0,0,0      ;EXTENDED TO 6 WORDS FOR .CSTAT
000026  000000  000000  000000
000034  000000                  ODTFLG: .WORD   0
000036  000000                  ODTTOP: .WORD   0
000040  000000                  CCAFLG: .WORD   0
000042  000000                  CLK100: .WORD   0
000044  000000                  RTTRP4: .WORD   0
000046  000000                  RTMNVC: .WORD   0
000050                          SAVBLK: .BLKW   5
000062  075250  100020  000000  TSXSAV: .RAD50  /SY TSX    SAV/
000070  073376
000072  075250  100003  051646  KMNNAM: .RAD50  /SY TSKMONSAV/
000100  073376
000102  075250  011504  000000  CCLNAM: .RAD50  /SY CCL    SAV/
000110  073376
000112  000000  000000  000000  DSTBLK: .WORD   0,0,0,0
000120  000000
000122  000000                  XMVBAS: .WORD   0
000124  000000                  NMXHAN: .WORD   0
000126  000000                  HMAP:   .WORD   0
000130  000000                  FETDEV: .WORD   0
000132  000000                  TOPMEM: .WORD   0
000134  000000                  FMEMHI: .WORD   0               ;64-byte block # below high alloc memory
000136  000000                  FMEMLO: .WORD   0               ;64-byte block # above top of low alloc memory
000140  000000                  OVLBAS: .WORD   0               ;Start loading overlays over TSINIT from here
000142  000000                  FILBLK: .WORD   0
000144  000000                  CURDEV: .WORD   0
000146  000000                  CURNAM: .WORD   0
000150  000000                  PROBUF: .WORD   0
000152  025350'                 WRKBUF: .WORD   INITOP
000154  004000                  WRKSIZ: .WORD   2048.
000156  000322'                  RTVPTR: .WORD   RTVEND          ;Initially no limit on system version number
                                        ;  The following RAD50 definitions were converted from word cells to
                                        ;  definitions to conserve space at V6.40.  All subsequent references to
                                        ;  one of these cells are converted from:
                                        ;  relative (cmp R50xxx,r1) to immediate (cmp #R50xxx,r1).
052077                                  R50MSG  = ^RMSG
110466                                  R50WIN  = ^RWIN
046543                                  R50LOK  = ^RLOK
103112                                  R50USR  = ^RUSR
012700                                  R50CSH  = ^RCSH
077167                                  R50TIO  = ^RTIO
100040                                  R50TT   = ^RTT          ;"TT "
075250                                  R50SY   = ^RSY          ;"SY "
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  004312'                 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
003444  105737  000000G                 TSTB    PROFLG          ;Are we running on a PRO?
003450  001403                          BEQ     7$              ;Br if not on a PRO
003452  004737  000000G                 CALL    PRONOP          ;Disable its interrupts
003456  000407                          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).
                                ;
003460  012702  000000G         7$:     MOV     #UEXINT,R2      ;SEND UNEXPECTED INTERRUPTS TO THIS ROUTINE
003464  012700  000044                  MOV     #44,R0          ;120 ENCODED IN PS FIELDS
003470  105737  000000G                 TSTB    VUXIFL          ;ARE WE TO IGNORE UNEXPECTED INTERRUPTS?
003474  001004                          BNE     10$             ;BR IF NOT
003476  012702  000000G         5$:     MOV     #UEXRTN,R2      ;ROUTINE TO GO TO TO IGNORE INTERRUPT
003502  012700  000340                  MOV     #340,R0         ;SET PRIO=7 IN PS
003506  012701  000120         10$:    MOV     #120,R1         ;INIT ALL VECTORS STARTING AT 120
003512  010221                  1$:     MOV     R2,(R1)+        ;SET PC FOR INTERRUPT
003514  010021                          MOV     R0,(R1)+        ;SET PS FOR INTERRUPT (ENCODED ADDRESS VALUE)
003516  105737  000000G         6$:     TSTB    VUXIFL          ;ARE WE TO IGNORE UNEXPECTED INTS?
003522  001411                          BEQ     2$              ;BR IF YES
003524  105737  000000G                 TSTB    PROFLG          ;IS THIS A PRO?
003530  001006                          BNE     2$              ;BR IF YES
003532  005200                          INC     R0              ;ADVANCE ENCODED ADDRESS
003534  032700  000020                  BIT     #20,R0          ;DID WE CARRY INTO "T"-FIELD?
003540  001402                          BEQ     2$              ;BR IF NOT
```

```
* * * Initialization taking over control from RT-11 * * *

     58 003542  062700  000020                ADD     #20,R0          ;FORCE CARRY OUT OF T-FIELD AND INTO PRIO FIELD
     59 003546  020127  000420          2$:   CMP     R1,#420         ;DONE ALL INTERRUPT VECTORS OF INTEREST?
     60 003552  103757                        BLO     1$              ;BR IF NOT
     61 003554  010237  000060                MOV     R2,@#60         ;CATCH CONSOLE TERMINAL VECTOR TOO
     62 003560  012737  000014  000062        MOV     #14,@#62        ;ENCODED 60
     63 003566  010237  000064                MOV     R2,@#64
     64 003572  012737  000015  000066        MOV     #15,@#66        ;ENCODED 64
     65                                ;
     66                                ;  Direct clock interrupt to a dummy RTI instruction to avoid it causing
     67                                ;  trouble during the initialization process when things aren't set up
     68                                ;  and ready to go.
     69                                ;
     70 003600  012700  000340          11$:  MOV     #340,R0         ;PRIORITY 7 PS
     71 003604  012737  000000G 000000G       MOV     #CLKRTI,@#CLKVEC;Send clock interrupt to RTI instruct for now
     72 003612  010037  000002G               MOV     R0,@#CLKVEC+2
     73                                ;
     74                                ;  Take over traps, EMT, BPT, etc.
     75                                ;
     76 003616  005001                        CLR     R1              ;Start at location 0
     77 003620  012721  000137                MOV     #137,(R1)+      ;[JMP @#JMP0] ==> 0
     78 003624  012721  000000G               MOV     #JMP0,(R1)+     ;CATCH JUMPS TO LOCATION 0
     79 003630  012721  000000G               MOV     #TRP4,(R1)+     ;TRAP 4
     80 003634  005021                        CLR     (R1)+
     81 003636  012721  000000G               MOV     #TRP10,(R1)+    ;TRAP 10
     82 003642  005021                        CLR     (R1)+
     83 003644  012721  000000G               MOV     #TRP14,(R1)+    ;TRAP 14 (BREAKPOINTS)
     84 003650  010021                        MOV     R0,(R1)+
     85 003652  012721  000000G               MOV     #TRP20,(R1)+    ;IOT TRAP
     86 003656  005021                        CLR     (R1)+
     87 003660  012721  000000G               MOV     #TRP24,(R1)+    ;POWER FAIL
     88 003664  010021                        MOV     R0,(R1)+
     89 003666  012721  000000G               MOV     #EMTENT,(R1)+   ;EMT
     90 003672  005021                        CLR     (R1)+
     91 003674  012721  000000G               MOV     #TRP34,(R1)+    ;TRAP
     92 003700  005021                        CLR     (R1)+
     93 003702  012737  000000G 000114        MOV     #MEMPAR,@#114   ;MEMORY PARITY TRAP
     94 003710  010037  000116                MOV     R0,@#116
     95 003714  012737  000000G 000244        MOV     #FPTRAP,@#244   ;TRAP 244 -- FLOATING POINT TRAP
     96 003722  010037  000246                MOV     R0,@#246        ;Enter FPU trap at priority 7
     97 003726  012737  000000G 000250        MOV     #TRP250,@#250   ;TRAP 250 -- MEMORY MANAGEMENT TRAP
     98 003734  005037  000252                CLR     @#252
     99                                ;
    100                                ;  Initialize the system mapped region.
    101                                ;
    102 003740  010546                        MOV     R5,-(SP)        ;SAVE THE CURRENT CONTENTS OF R5
    103 003742  012705  000006                MOV     #6,R5           ;INITIALIZE TO THE FIRST REGION
    104 003746  004737  000000G               CALL    MAPSYS          ;CALL THE SYSTEM MAPPING ROUTINE
    105 003752  012605                        MOV     (SP)+,R5        ;RESTORE THE PREVIOUS CONTENTS OF R5
    106                               ;
    107                               ;  Set up Unibus mapping if needed
    108                               ;
    109 003754  004737  004650'               CALL    SETUMP          ;SET UP UNIBUS MAPPING
    110                               ;
    111                               ;  Initialize time-sharing lines.
    112                               ;
    113 003760  004737  005262'               CALL    LININI          ;INIT LINES & SET UP VECTORS
    114                               ;
```

```
                                        ;  Enable memory management
                                        ;  (The kernel-mode mapping registers are already set up)
                                        ;
003764  052737  000000G 000000G         BIS     #MMENBL,@#SROMMR;Turn on memory management
003772  105737  000000G                 TSTB    SR3FLG          ;Does machine have memory management reg 3?
003776  001415                         BEQ     4$              ;Br if register does not exist (no ext. mem.)
004000  023727  000000G 010000         CMP     PHYMEM,#4096.   ;Does machine have at least 256Kb phys memory?
004006  103411                         BLO     4$              ;Br if not
004010  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;Set extended memory on
004016  105737  000000G                 TSTB    MEM256          ;Will TSX-Plus use at least 256Kb?
004022  001403                         BEQ     4$              ;Br if not
004024  052737  000000G 000000G         BIS     #IOMAP,@#SR3MMR ;Turn on 22-bit memory management for I/O
                                        ;
                                        ;  Initialize the memory allocation table
                                        ;
004032  013737  000000G 000000G 4$:     MOV     MAPPAR,@#KPAR5  ;Map to memory allocation table
004040  013702  000000G                 MOV     LOMAP,R2        ;Point to 1st user-page entry
004044  105022                 8$:     CLRB    (R2)+           ;Say page is free
004046  020237  000000G                 CMP     R2,HIMAP        ;Done all user pages?
004052  103774                         BLO     8$              ;Loop if not
004054  112712  000000G                 MOVB    #MA$SYS,(R2)    ;Set flag marking start of system pages
                                        ;
                                        ;  Set up I/O device interrupt vectors.
                                        ;
004060  004737  004652'                 CALL    DEVVEC          ;SET UP DEVICE INTERRUPT VECTORS
                                        ;
                                        ;  If we are running on a Professional, initiailize the PI handler
                                        ;
                                .IF     NE,PROASM
004064  105737  000000G                 TSTB    PROFLG          ;Are we running on a Professional?
004070  001404                         BEQ     3$              ;Br if not
004072  004737  000000G                 CALL    PROHAN          ;Initialize the PI handler
004076  004737  005200'                 CALL    PIDVEN          ;Make device table entry for PI
                                .ENDC   ;NE,PROASM
                                        ;
                                        ;  Initialize interrupt stack area
                                        ;
004102  013702  000000G         3$:     MOV     INTSND,R2       ;Point to base of stack area
004106  012700  123456                 MOV     #123456,R0      ;Get initialization value
004112  010022                 12$:    MOV     R0,(R2)+        ;Initialize the interrupt stack area
004114  020237  000000G                 CMP     R2,INTSTK       ;Finished?
004120  103774                         BLO     12$             ;Loop if not
004122  112737  177777  000000G         MOVB    #-1,STKLVL      ;Say we are not running on interrupt stack
                                        ;
                                        ; Enter TSEXEC to complete initialization
                                        ;
004130  000137  000000G                 JMP     INIJMP          ;ENTER INITIALIZATION ROUTINE IN TSEXEC
                                        ;
                                        ;  Abort the initialization
                                        ;
004134  013737  000042' 000000G INISTP: MOV     CLK100,@#CLKVEC ;Restore RT-11 clock vector
004142  013737  000044' 000004          MOV     RTTRP4,@#4      ;Restore trap 4 vector
004150  013737  000046' 000000G         MOV     RTMNVC,@#RMON   ;Restore RT-11 monitor pointer
004156                          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.
                                ;
004160  010146          LODINI: MOV     R1,-(SP)
004162  010346                  MOV     R3,-(SP)
004164  010446                  MOV     R4,-(SP)
                                ;
                                ;  Get pointer to linker-built overlay entry
                                ;
004166  016201  000004          MOV     OS$OVL(R2),R1   ;Get pointer to linker-built table
                                ;
                                ;  Determine how much code to read from the segment
                                ;
004172  016204  000000          MOV     OS$SIZ(R2),R4   ;Get # 64-byte blks allocated for segment
004176  072427  000005          ASH     #5,R4           ;Convert to # words
004202  020461  000000G         CMP     R4,O.SIZ(R1)    ;Compare with original segment code size
004206  101402                  BLOS    1$              ;Br if segment was truncated by init
004210  016104  000000G         MOV     O.SIZ(R1),R4    ;Get code size
                                ;
                                ;  Read the segment into memory
                                ;
004214  010503          1$:     MOV     R5,R3           ;Get 64-byte block #
004216  072327  000006          ASH     #6,R3           ;Convert to byte address
004222                          .READW  #AREA,#17,R3,R4,O.BLK(R1)
004256  103406                  BCS     10$             ;Br if error on read
                                ;
                                ;  Store the physical address of the segment into the overlay descriptor
                                ;
004260  010561  000000G         MOV     R5,O.PAR(R1)    ;Remember physical address of segment
                                ;
                                ;  Finished
                                ;
004264  012604                  MOV     (SP)+,R4
004266  012603                  MOV     (SP)+,R3
004270  012601                  MOV     (SP)+,R1
004272  000207                  RETURN
                                ;
                                ;  Error on read
                                ;
004274                  10$:    .PRINT  #TSXHD
004302                          .PRINT  #RDERR
004310                          .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.
                                ;
004312  010246                  INIOVL: MOV     R2,-(SP)
004314  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize pointer to start of memory area for overlays
                                ;
004316  013705  000140'                 MOV     OVLBAS,R5       ;Start of area for overlays
004322  072527  177772                  ASH     #-6,R5          ;Convert to 64-byte #
004326  042705  176000                  BIC     #176000,R5      ;Clear possible propagated sign bits
                                ;
                                ;  Begin loop to load each overlay that goes over TSINIT
                                ;
004332  012702  000516'                 MOV     #OSTABL,R2      ;Point to 1st overlay segment entry
004336  005762  000002          1$:     TST     OS$FLG(R2)      ;Does this segment go over TSINIT?
004342  001406                          BEQ     2$              ;Br if not
004344  004737  004522'                 CALL    KEYSEG          ;Remember base of some segments
004350  004737  004160'                 CALL    LODINI          ;Load the segment
004354  066205  000000                  ADD     OS$SIZ(R2),R5   ;Advance memory pointer
004360  062702  000006          2$:     ADD     #OS$$SZ,R2      ;Point to entry for next segment
004364  020237  000744'                 CMP     R2,OSLAST       ;Finished all segments?
004370  103762                          BLO     1$              ;Loop if not
                                ;
                                ;  Initialize entry point vector for TSTIOX segment
                                ;
004372  013702  000000G                 MOV     TIOBAS,R2       ;Get addr of base of TSTIOX
004376  072227  000006                  ASH     #6,R2           ;Convert to byte address
004402  012705  000000G                 MOV     #TIOVEC,R5      ;Point to entry point vector
004406  004737  004464'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Initialize entry point vector for TSCASH segment
                                ;
004412  013702  000000G                 MOV     CSHBAS,R2       ;Get addr of base of TSCASH
004416  001406                          BEQ     3$              ;Br if TSCASH not loaded
004420  072227  000006                  ASH     #6,R2           ;Convert to byte address
004424  012705  000000G                 MOV     #CSHVEC,R5      ;Point to entry point vector
004430  004737  004464'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Initialize entry point vector for TSLOCK segment
                                ;
004434  013702  000000G         3$:     MOV     LOKBAS,R2       ;Get addr of base of TSLOCK
004440  001406                          BEQ     9$              ;Br if TSLOCK not loaded
004442  072227  000006                  ASH     #6,R2           ;Convert to byte address
004446  012705  000000G                 MOV     #LOKVEC,R5      ;Point to entry point vector
004452  004737  004464'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Finished
                                ;
004456  012605                  9$:     MOV     (SP)+,R5
004460  012602                          MOV     (SP)+,R2
004462  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)
                                ;
004464  010246                  ENTVEC: MOV     R2,-(SP)
004466  010446                          MOV     R4,-(SP)
004470  010546                          MOV     R5,-(SP)
004472  010204                          MOV     R2,R4           ;Get addr of base of segment
004474  062704  000004                  ADD     #4,R4           ;Point to start of vector in segment
004500  005722                          TST     (R2)+           ;Get value to use to relocate offsets
004502  012415                  1$:     MOV     (R4)+,(R5)      ;Get offset to entry point within segment
004504  060225                          ADD     R2,(R5)+        ;Convert to absolute address
004506  005715                          TST     (R5)            ;Any more words to initialize?
004510  001774                          BEQ     1$              ;Br if yes
                                ;
                                ;  Finished
                                ;
004512  012605                          MOV     (SP)+,R5
004514  012604                          MOV     (SP)+,R4
004516  012602                          MOV     (SP)+,R2
004520  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.
                                ;
004522  010446                  KEYSEG: MOV     R4,-(SP)
                                ;
                                ;  Get the name of the segment out of the linker-built segment block
                                ;
004524  016200  000004                  MOV     OS$OVL(R2),R0   ;Point to linker-built entry
004530  016004  000000G                 MOV     O.ADR(R0),R4    ;Get the name of the segment
                                ;
                                ;  See if this is a segment whose address we want to remember
                                ;
004534  020427  052077                  CMP     R4,#R50MSG      ;Is this the TSMSG segment?
004540  001003                          BNE     1$              ;Br if not
004542  010537  000000G                 MOV     R5,MSGBAS       ;Remember base of TSMSG segment
004546  000436                          BR      8$
004550  020427  110466          1$:     CMP     R4,#R50WIN      ;Is this the TSWIN segment?
004554  001003                          BNE     3$              ;Br if not
004556  010537  000000G                 MOV     R5,WINBAS       ;Remember base of TSWIN segment
004562  000430                          BR      8$
004564  020427  103112          3$:     CMP     R4,#R50USR      ;Is this the TSUSR segment?
004570  001003                          BNE     4$              ;Br if not
004572  010537  000000G                 MOV     R5,USRBAS       ;Remember base of TSUSR segment
004576  000422                          BR      8$
004600  020427  046543          4$:     CMP     R4,#R50LOK      ;Is this the TSLOCK segment?
004604  001003                          BNE     5$              ;Br if not
004606  010537  000000G                 MOV     R5,LOKBAS       ;Remember base of TSLOCK segment
004612  000414                          BR      8$
004614  020427  012700          5$:     CMP     R4,#R50CSH      ;Is this the TSCASH segment?
004620  001003                          BNE     6$              ;Br if not
004622  010537  000000G                 MOV     R5,CSHBAS       ;Remember base of TSCASH segment
004626  000406                          BR      8$
004630  020427  077167          6$:     CMP     R4,#R50TIO      ;Is this the TSTIOX segment?
004634  001003                          BNE     8$              ;Br if not
004636  010537  000000G                 MOV     R5,TIOBAS       ;Remember base of module
004642  000400                          BR      8$
                                ;
                                ;  Finished
                                ;
004644  012604                  8$:     MOV     (SP)+,R4
004646  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.
                ;
                SETUMP: MOV     R2,-(SP)
                        MOV     R3,-(SP)
                        MOV     @#4,-(SP)       ;SAVE TRAP VECTOR
                        MOV     #9$,@#4         ;CATCH TRAPS
                ;
                ;  See if this is a type of maching that needs unibus mapping
                ;
                        TSTB    UBUSMP          ;Is UNIBUS mapping needed?
                        BEQ     9$              ;Br if not
                ;
                ;  Unibus mapping is needed
                ;  Load unibus map registers # 0-4 to point to low 48Kb of memory.
                ;
                2$:     MOV     #UMRADR,R5      ;POINT TO CONTROL REGISTER FOR UNIBUS MAP 0
                        CLR     R4              ;START MAPPING TO BOTTOM OF MEMORY
                        MOV     #5,R0           ;LOAD 5 MAP REGISTERS
                1$:     MOV     R4,(R5)+        ;SET LOW-ORDER VALUE IN MAP REGISTER
                        CLR     (R5)+           ;CLEAR HIGH-ORDER VALUE IN MAP REGISTER
                        ADD     #8192.,R4       ;ADVANCE MEMORY ADDRESS
                        SOB     R0,1$           ;LOOP IF MORE MAP REGISTERS TO LOAD
                ;
                ;  Turn on Unibus mapping
                ;
                        BIS     #IOMAP,@#SR3MMR ;ENABLE UNIBUS MAPPING
                ;
                ;  Finished
                ;
                9$:     MOV     (SP)+,@#4       ;RESTORE TRAP VECTOR
                        MOV     (SP)+,R5
                        MOV     (SP)+,R4
                        RETURN
                        .IFF    ;NE,<PROASM-1>  ;Following code for Pro-only assembly
                ;
                ;  Define dummy SETUMP routine for Pro
                ;
 004650  000207 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.
                                ;
004652  010146                  DEVVEC: MOV     R1,-(SP)
004654  010346                          MOV     R3,-(SP)
004656  010546                          MOV     R5,-(SP)
004660  013746  000000G                 MOV     @#KPAR5,-(SP)   ;Save PAR 5 mapping
                                ;
                                ;  Begin loop to set up vectors for each device
                                ;
004664  012701  000002                  MOV     #2,R1           ;Get index # of 1st device after TT
004670  016103  000000G         1$:     MOV     HANENT(R1),R3   ;Get handler entry point address
004674  020327  000006                  CMP     R3,#6           ;Is this a real device?
004700  101436                          BLOS    6$              ;Br if not
                                ;
                                ;  See if we need to map PAR 5 to this handler
                                ;
004702  016100  000000G                 MOV     HANPAR(R1),R0   ;Get PAR 5 base for this handler
004706  001402                          BEQ     2$              ;Br if this is not a mapped handler
004710  010037  000000G                 MOV     R0,@#KPAR5      ;Map PAR 5 to the handler
                                ;
                                ;  Clear CQE and LQE in handler header
                                ;
004714  005023                  2$:     CLR     (R3)+           ;Clear LQE (4th word in handler)
004716  005013                          CLR     (R3)            ;Clear CQE (5th word in handler)
004720  162703  000010                  SUB     #10,R3          ;Point to 1st word of handler
                                ;
                                ;  Set up interrupt vectors for this handler
                                ;
004724  005005                          CLR     R5              ;Assume vector base address is 0
004726  005713                          TST     (R3)            ;Any vectors to set up?
004730  001422                          BEQ     6$              ;Br if no vectors to set up
004732  002403                          BLT     5$              ;Br if multiple-vector
004734  004737  005024'                 CALL    SETVEC          ;Set up the vector
004740  000416                          BR      6$              ;Finished
                                ;
                                ;  Multiple vectors.
                                ;
004742  012300                  5$:     MOV     (R3)+,R0        ;Get offset to list of vector info
004744  006300                          ASL     R0              ;Get byte offset to vector list
004746  060003                          ADD     R0,R3           ;Get absolute address of vector table
004750  005713                          TST     (R3)            ;Is this a PRO device with floating vectors?
004752  002005                          BGE     7$              ;Br if not
004754  005723                          TST     (R3)+           ;Point to word with device ID
004756  012346                          MOV     (R3)+,-(SP)     ;Get device ID
004760  004777  000000G                 CALL    @RPRVEC         ;Get base vector location for device
004764  012605                          MOV     (SP)+,R5        ;This is base of vector locations
004766  004737  005024'         7$:     CALL    SETVEC          ;Set up the vector
004772  005713                          TST     (R3)            ;Any more vectors to set up?
004774  003374                          BGT     7$              ;Br if yes
                                ;
                                ;  See if there are more devices to set up.
                                ;
004776  062701  000002          6$:     ADD     #2,R1           ;Advance device table index
005002  020137  000000G                 CMP     R1,NUMDEV       ;More to do?
```

```
     58 005006  101730                          BLOS    1$              ;Br if yes
     59                                 ;
     60                                 ;  Finished
     61                                 ;
     62 005010  012637  000000G                 MOV     (SP)+,@#KPAR5
     63 005014  012605                          MOV     (SP)+,R5
     64 005016  012603                          MOV     (SP)+,R3
     65 005020  012601                          MOV     (SP)+,R1
     66 005022  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
                                ;
005024  010446                  SETVEC: MOV     R4,-(SP)
                                ;
                                ;  See if this is a mapped handler
                                ;
005026  005761  000000G                 TST     HANPAR(R1)      ;Is this a mapped handler
005032  001010                          BNE     1$              ;Br if yes
                                ;
                                ;  This is an unmapped handler.
                                ;  Vector interrupts directly to the handler.
                                ;
005034  012300                          MOV     (R3)+,R0        ;Get address of interrupt vector
005036  060500                          ADD     R5,R0           ;Add base address to vector location
005040  010310                          MOV     R3,(R0)         ;Store address of cell in handler
005042  062320                          ADD     (R3)+,(R0)+     ;Add offset to interrupt entry point
005044  012310                          MOV     (R3)+,(R0)      ;Set PS for interrupt
005046  052710  000340                  BIS     #340,(R0)       ;Make sure priority = 7
005052  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.
                                ;
005054  013704  000122'         1$:     MOV     XMVBAS,R4       ;Point to area where we store interrupt rtn
005060  012300                          MOV     (R3)+,R0        ;Get address of interrupt vector
005062  060500                          ADD     R5,R0           ;Add base address to interrupt location
005064  010420                          MOV     R4,(R0)+        ;Direct interrupt to our routine
005066  012724  013746                  MOV     #013746,(R4)+   ; [ MOV @#KPAR5,-(SP) ]
005072  012724  000000G                 MOV     #KPAR5,(R4)+
005076  012724  012737                  MOV     #012737,(R4)+   ; [ MOV #par5val,@#KPAR5 ]
005102  016124  000000G                 MOV     HANPAR(R1),(R4)+
005106  012724  000000G                 MOV     #KPAR5,(R4)+
005112  012724  012746                  MOV     #012746,(R4)+   ; [ MOV #340,-(SP) ]
005116  012724  000340                  MOV     #340,(R4)+
```

```
     58 005122  012724  012746                    MOV     #012746,(R4)+   ; [ MOV #HANXIT,-(SP) ]
     59 005126  012724  000000G                   MOV     #HANXIT,(R4)+
     60 005132  012724  000257                    MOV     #000257,(R4)+   ; [ CCC - Clear all condition codes ]
     61 005136  016314  000002                    MOV     2(R3),(R4)      ; [ SEx - Set condition codes specified in PS]
     62 005142  042714  177760                    BIC     #^C17,(R4)
     63 005146  052724  000260                    BIS     #260,(R4)+
     64 005152  012724  000137                    MOV     #000137,(R4)+   ; [ JMP @#handler_entry ]
     65 005156  010314                            MOV     R3,(R4)         ; Store address of int entry point
     66 005160  062324                            ADD     (R3)+,(R4)+
     67 005162  012310                            MOV     (R3)+,(R0)      ;Set PS for interrupt entry
     68 005164  052710  000340                    BIS     #340,(R0)       ;Make sure priority = 7
     69                                ;
     70                                ;  Save address beyond end of compiled interrupt catcher routine
     71                                ;
     72 005170  010437  000122'                   MOV     R4,XMVBAS       ;Save address beyond end of routine
     73                                ;
     74                                ;  Finished
     75                                ;
     76 005174  012604                 9$:       MOV     (SP)+,R4
     77 005176  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.
                                ;
005200  010146                  PIDVEN: MOV     R1,-(SP)
                                ;
                                ;  Increase number of defined devices and get device table entry index
                                ;  to use for the PI device.
                                ;
005202  062737  000002  000000G         ADD     #2,NUMDEV       ;One more device
005210  013701  000000G                 MOV     NUMDEV,R1       ;Get device table index
                                ;
                                ;  Set up information about the PI device
                                ;
005214  012761  062550  000000G         MOV     #R50PI,PNAME(R1) ;Set device name
005222  012761  000000C 000000G         MOV     #<DS$SFN!DI$PI>,DVSTAT(R1) ;Set device status flags
005230  005061  000000G                 CLR     DVFLAG(R1)      ;Clear other flags
005234  005061  000000G                 CLR     DEVSIZ(R1)      ;Clear device size
005240  012761  000006G 000000G         MOV     #PIHAN+6,HANENT(R1);Set handler entry point (4th word)
005246  012761  000000G 000000G         MOV     #PROSIZ,HANSIZ(R1) ;Set handler size
                                ;
                                ;  Finished
                                ;
005254  012601                          MOV     (SP)+,R1
005256  000207                          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.
                ;
LININI: MOV     R1,-(SP)
        MOV     R2,-(SP)
        MOV     R3,-(SP)
        MOV     R4,-(SP)
        MOV     R5,-(SP)
;
;  Set up interrupt vectors for DL11 lines
;
        MOV     #2,R1           ;Index for 1st line
        MOV     #340,R4         ;Priority 7 PS
1$:     BIT     #$DEAD,LSW3(R1) ;Is this line uninstalled?
        BNE     8$              ;Br if yes
        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
        BEQ     8$              ;Br if not
        CMP     LCDTYP(R1),#CDX$DL ;Is this a DL11 line?
        BNE     8$              ;Br if not
;
;  DL-11 line
;
        MOV     INVEC(R1),R5    ;GET ADDRESS OF INPUT VECTOR
        MOV     #INRECV,R2      ;END OF RECEIVING VECTOR
        MOV     #<OTRECV-6>,R3  ;START OF INPUT INTERRUPT ENTRY POINTS
        MOV     R1,R0           ;GET LINE NUMBER
        ASL     R0              ;4 BYTES PER INPUT INTERRUPT ENTRY POINT
        SUB     R0,R2           ;GET ADDRESS OF INPUT INTERRUPT ENTRY POINT
        ADD     R1,R0           ;6 BYTES PER OUTPUT INTERRUPT ENTRY POINT
        ADD     R0,R3           ;GET ADDRESS OF OUTPUT INTERRUPT ENTRY POINT
        MOV     R2,(R5)+        ;SET PC FOR INPUT INTERRUPT ENTRY POINT
        MOV     R4,(R5)+        ;SET PS FOR INPUT INTERRUPT
        MOV     R3,(R5)+        ;SET PC FOR OUTPUT INTERRUPT
        MOV     R4,(R5)+        ;SET PS FOR OUTPUT INTERRUPT
;
;  Try next line
;
8$:     ADD     #2,R1           ;ADVANCE LINE INDEX NUMBER
        CMP     R1,#LSTHL       ;MORE TO DO?
        BLOS    1$              ;BR IF YES
;
;  Initialize multiplexers.
;
SETMUX: MOV     #LSTMX,R1       ;Get last mux index #
        BEQ     SETLIN          ;Br if there are no mux lines
3$:     CMP     MXTYPE(R1),#CDX$DZ ;Is this a DZ11, DH11, or DHV11?
        BEQ     1$              ;Br if DZ11
        CMP     MXTYPE(R1),#CDX$VH ;Is this a DHV11?
        BNE     2$              ;Br if not
        CALL    VHINIT          ;Initialize a DHV11
        BR      4$
2$:     CALL    DHINIT          ;Initialize a DH11
        BR      4$
```

```
                                1$:     CALL    DZINIT          ;Initialize a DZ11
                                4$:     SUB     #2,R1           ;More to enable?
                                        BNE     3$              ;Br if yes
                                ;
                                ;  Enable all lines
                                ;
                                SETLIN: MOV     #LSTHL,R1       ;INDEX # OF LAST REAL LINE
                                4$:     BIT     #$DEAD,LSW3(R1) ;IS THIS LINE INSTALLED?
                                        BNE     2$              ;BR IF NOT
                                        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
                                        BEQ     2$              ;Br if not
                                        BIT     #$PHONE,ILSW2(R1);IS THIS A DIAL-UP LINE?
                                        BEQ     3$              ;BR IF NOT
                                        BIS     #$NOIN,LSW3(R1) ;IGNORE INPUT TILL DIAL UP OCCURS
                                3$:     MOV     LCDTYP(R1),R5   ;Get comm device type code
                                        MOV     LMXNUM(R1),R0   ;IS THIS A DL-11 OR MUX LINE?
                                        BEQ     1$              ;BR IF DL-11
                                        CMP     R5,#CDX$DZ      ;Is this a DZ11 or DH11?
                                        BEQ     6$              ;Br if DZ11
                                        CMP     R5,#CDX$VH      ;Is this a DH11 or DHV11?
                                        BEQ     7$              ;Br if DHV11
                                ;
                                ;  DH11 line
                                ;
                                        CALL    DHLPRM          ;Set line parameters for DH11 line
                                        BR      2$
                                ;
                                ;  DHV11 line
                                ;
                                7$:     CALL    VHLPRM          ;Set line parameters for DHV11 line
                                        BR      2$
                                ;
                                ;  DZ-11 line
                                ;
                                6$:     MOV     LMXLN(R1),R2    ;Get line # within mux group
                                        BIS     #017030,R2      ;Set line enable flags
                                        MOV     R2,@MXLPR(R0)   ;Enable the line
                                        BR      2$
                                ;
                                ;  DL-11 line
                                ;
                                1$:     MOV     TSR(R1),R2      ;ADDRESS OF TRANSMITTER STATUS REGISTER
                                        MOV     (R2),R3         ;CLEAR TRANSMITTER STATUS REGISTER
                                        CLR     (R2)
                                        MOV     RBR(R1),R2      ;ADDRESS OF RECEIVER BUFFER REGISTER
                                        MOV     (R2),R3         ;CLEAR RECEIVER BUFFER REGISTER
                                        MOV     RSR(R1),R2      ;ADDRESS OF RECEIVER STATUS REGISTER
                                        CLR     (R2)
                                        MOV     #RDINT,(R2)     ;ENABLE RECEIVER INTERRUPTS
                                ;
                                ;  Do next line
                                ;
                                2$:     SUB     #2,R1
                                        BGT     4$
                                ;
                                ;  Finished
                                ;
```

```
    115                                         MOV     (SP)+,R5
    116                                         MOV     (SP)+,R4
    117                                         MOV     (SP)+,R3
    118                                         MOV     (SP)+,R2
    119                                         MOV     (SP)+,R1
    120                                         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.
                ;
                DHLPRM:
                ;
                ;  Enable DM11 for this line
                ;
                        MOV     LMXNUM(R1),R0   ;Get mux index number
                        TST     DM$CSR(R0)      ;Does this DH11 have DM11 modem control?
                        BEQ     2$              ;Br if not
                        BICB    #MF$LIN,@DM$CSR(R0) ;Clear line # field in DM11 CSR
                        BISB    LMXLN(R1),@DM$CSR(R0);Select line of interest
                        MOV     #MF$LE,@DM$LSR(R0);Enable the line
                ;
                ;  Finished
                ;
                2$:     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.
                ;
                VHLPRM:
                ;
                ;  Enable the line
                ;
                        MOV     LMXNUM(R1),R0   ;Get mux index number
                        BIC     #VF$LIN,@VH$CSR(R0) ;Clear line # field in mux CSR
                        BISB    LMXLN(R1),@VH$CSR(R0) ;Set our line #
                        MOV     #<VF$RE>,@VH$LCR(R0) ;Enable the line
                ;
                ;  Finished
                ;
                        RETURN
```

```
                        .SBTTL  DZINIT -- Initialize a DZ11 multiplexer
                ;-----------------------------------------------------------------------
                ;  DZINIT is called to initialize a DZ11 multiplexer.
                ;
                ;  Inputs:
                ;   R1 = Mux index number.
                ;
                DZINIT:
                ;
                ;  See if this DZ11 is installed
                ;
                        TST     MXCSR(R1)       ;Is this DZ-11 installed?
                        BEQ     4$              ;Br if not
                ;
                ;  Set up interrupt vector connections for this MUX
                ;
                        CALL    MUXVEC          ;Set up interrupt vectors for this DZ11
                ;
                ;  Start up the mux operation
                ;
                        BIS     #ZCLR,@MXCSR(R1);Do master clear on DZ-11
                1$:     BIT     #ZCLR,@MXCSR(R1);Wait for clear to finish
                        BNE     1$
                2$:     MOV     @MXRBUF(R1),R0  ;Clear silo
                        BMI     2$
                        CLRB    @MXDTR(R1)      ;Disable all data sets
                ;
                ;  Finished
                ;
                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.
                ;
MUXVEC: MOV     R2,-(SP)
        MOV     R3,-(SP)
        MOV     R5,-(SP)
                ;
                ;  Set interrupt vector for mux
                ;
        MOV     MXVEC(R1),R5    ;Get address of input interrupt vector
        MOV     #INMXV,R2       ;End of receiving vector
        MOV     #<OTMXV-6>,R3   ;Output interrupt table
        MOV     R1,R0           ;Get mux index number
        ASL     R0              ;4 bytes per line in input int table
        SUB     R0,R2           ;Get address of input entry point
        ADD     R1,R0           ;6 bytes per mux in output entry point table
        ADD     R0,R3           ;Get address of output int entry point
        MOV     R2,(R5)+        ;Set PC for input interrupt
        MOV     #340,(R5)+      ;Set PS for output interrupt
        MOV     R3,(R5)+        ;Set PC for output interrupt
        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.
                ;
        MOV     #004537,(R3)+   ;JSR R5,@#x
        MOV     #DZOINT,R0      ;Assume this is a DZ11
        CMP     MXTYPE(R1),#CDX$DZ ;Is this a DZ11?
        BEQ     1$              ;Br if yes
        MOV     #VHOINT,R0      ;Assume this is a DHV11
        CMP     MXTYPE(R1),#CDX$VH ;Is this a DHV11?
        BEQ     1$              ;Br if yes
        MOV     #DHOINT,R0      ;Get interrupt routine for DH11's
1$:     MOV     R0,(R3)+        ;Store address of interrupt routine
        MOV     R1,(R3)         ;Store mux index number
                ;
                ;  Finished
                ;
9$:     MOV     (SP)+,R5
        MOV     (SP)+,R3
        MOV     (SP)+,R2
        RETURN
```

```
                        .SBTTL  DHINIT -- Initialize a DH11 multiplexer
                ;-----------------------------------------------------------------------
                ;  DHINIT is called to initialize a DH11 multiplexer
                ;
                ;  Inputs:
                ;   R1 = Mux index number
                ;
                DHINIT:
                ;
                ;  See if this DH11 is installed
                ;
                        TST     MH$SCR(R1)      ;Is this DH11 installed?
                        BEQ     9$              ;Br if not
                ;
                ;  Connect interrupt vector to DH11
                ;
                        CALL    MUXVEC          ;Set up interrupt vectors for DH11
                ;
                ;  Clear the multiplexer
                ;
                        MOV     #HF$MC,@MH$SCR(R1) ;Set the master-clear flag
                1$:     BIT     #HF$MC,@MH$SCR(R1) ;Wait for the master clear to be completed
                        BNE     1$
                ;
                ;  Clear the DM11 scanner
                ;
                        MOV     DM$CSR(R1),R0   ;Is there an associated DM11?
                        BEQ     3$              ;Br if not
                        MOV     #MF$CS,(R0)     ;Clear the scanner
                        BIS     #MF$CM,(R0)     ;Clear the multiplexer
                3$:
                ;
                ;  Finished
                ;
                9$:
                        RETURN
```

```
                        .SBTTL  VHINIT -- Initialize a DHV11 multiplexer
                ;-----------------------------------------------------------------------
                ;  Perform initialization for a DHV11 mux.
                ;
                ;  Inputs:
                ;   R1 = Mux index number.
                ;
                VHINIT:
                ;
                ;  See if this DHV11 is installed
                ;
                        TST     VH$CSR(R1)      ;Is this DHV11 installed?
                        BEQ     9$              ;Br if not
                ;
                ;  Connect interrupt vector to DHV11
                ;
                        CALL    MUXVEC          ;Set up interrupt vectors
                ;
                ;  Clear the multiplexer
                ;
                        MOV     #VF$MR,@VH$CSR(R1) ;Reset the multiplexer
                1$:     BIT     #VF$MR,@VH$CSR(R1) ;Wait for reset to finish
                        BNE     1$
                ;
                ;  Clean out the FIFO buffer in the mux
                ;
                2$:     MOV     @MXRBUF(R1),R0  ;Get contents of receiver buffer register
                        BLT     2$              ;Loop until RBUF empty
                ;
                ;  Finished
                ;
                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.
                                ;
005260                          LINCHK:
005260                          DHLPRM:
005260                          VHLPRM:
005260                          DZINIT:
005260                          MUXVEC:
005260                          DHINIT:
005260                          VHINIT:
005260  000207                          RETURN
                                ;
                                ;  LININI routine for Pro systems
                                ;
005262  010146                  LININI: MOV     R1,-(SP)
005264  012701  000000G                 MOV     #LSTLIN,R1      ;Get # of last line
                                ;
                                ;  Determine if this line is connected to hardware
                                ;
005270  004737  005340'         1$:     CALL    LINTYP          ;Determine the type of this line
005274  032761  000000G 000000G        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
005302  001411                        BEQ     2$              ;Br if not
                                ;
                                ;  Call Pro line initialization routine
                                ;
005304  004737  000000G              CALL    PROLIN          ;Initialize Pro line
                                ;
                                ;  Do some special init for phone lines
                                ;
005310  032761  000000G 000000G      BIT     #$PHONE,ILSW2(R1);Is this a dialup line?
005316  001403                      BEQ     2$              ;Br if not
005320  052761  000000G 000000G      BIS     #$NOIN,LSW3(R1) ;Ignore input till dial up occurs
                                ;
                                ;  Check next line
                                ;
005326  162701  000002          2$:     SUB     #2,R1           ;Get index # of next line
005332  003356                          BGT     1$              ;Loop if more lines to do
                                ;
                                ;  Finished
                                ;
005334  012601                        MOV     (SP)+,R1
005336  000207                        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
                                ;
005340  020127  000000G       LINTYP: CMP     R1,#LSTPL       ;Is this a time-sharing line?
005344  101422                        BLOS    1$              ;Br if yes
005346  020127  000000G               CMP     R1,#FSTIOL      ;Is this a CL line?
005352  103004                        BHIS    2$              ;Br if yes
005354  012761  177777  000000G       MOV     #-1,LCLUNT(R1)  ;Say line not in use as a CL line
005362  000432                        BR      9$
                                ;
                                ;  This is a CL line
                                ;
005364  016100  000000G       2$:     MOV     LCLUNT(R1),R0   ;Get the CL unit index number
005370  010160  000000G               MOV     R1,CL$LIX(R0)   ;Say which line is assoc with this CL unit
005374  012761  000000G 000000G       MOV     #CLOTIR,LOUTIR(R1) ;Set terminal output interrupt routine
005402  012761  000000G 000000G       MOV     #CLINCP,LINIR(R1)  ;Set terminal input interrupt routine
005410  000414                        BR      8$
                                ;
                                ;  This is a time-sharing line
                                ;
005412  012761  177777  000000G 1$:     MOV     #-1,LCLUNT(R1)  ;Say line is not in use as a CL unit
005420  012761  177777  000000G       MOV     #-1,LXCL(R1)    ;Line is not cross-connected to CL unit
005426  012761  000000G 000000G       MOV     #NEDCHR,LOUTIR(R1) ;Set terminal output interrupt routine
005434  012761  000000G 000000G       MOV     #TTINCP,LINIR(R1)  ;Set terminal input interrupt routine
005442  052761  000000G 000000G 8$:     BIS     #$HARD,LSW3(R1) ;This line is connected to hardware
                                ;
                                ;  Finished
                                ;
005450  000207                9$:     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.
                ;
005452          INITGO:
                ;
                ;  Save some RT-11 pointers in case we abort the initialization
                ;
005452  013737  000004  000044'        MOV     @#4,RTTRP4      ;Save trap 4 vector
005460  013737  000000G 000046'        MOV     @#RMON,RTMNVC   ;RT-11 monitor pointer
005466  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.
                ;
005474                                 .GTIM   #AREA,#SYTIMH   ;Get the current time of day
                ;
                ;  Trap ^C for later test so we can restore clock vector
                ;
005514                                 .SCCA   #AREA,#CCAFLG   ;Catch control-C
                ;
                ;  Check for TSGEN size overflow
                ;
005534  012700  000000G                MOV     #GENTOP,R0      ;Get top of TSGEN
005540  162700  037776                 SUB     #<40000-2>,R0   ;Will TSKMON have problems?
005544  003422                         BLE     15$             ;Continue if not
005546  010046                         MOV     R0,-(SP)        ;Save overflow size
005550                                 .PRINT  #TSXHD          ;Print error message
005556                                 .PRINT  #TOOBIG
005564                                 .PRINT  #REDUCE
005572  012600                         MOV     (SP)+,R0        ;Recover amount of overflow
005574  004737  025230'                CALL    PRTDEC
005600                                 .PRINT  #BYTES
005606  000137  004134'                JMP     INISTP
                ;
                ;  Initialize the system stack (below 1000)
                ;
005612  012701  000000G        15$:    MOV     #SSEND,R1       ;Point to bottom of stack
005616  012700  123456                 MOV     #123456,R0      ;Get initialization value
005622  010021                 13$:    MOV     R0,(R1)+        ;Initialize the stack
005624  020127  000000G                CMP     R1,#SS          ;Reached top of the stack area?
005630  103774                         BLO     13$             ;Loop if not
005632  010106                         MOV     R1,SP           ;Run on system stack
                ;
                ;  Make sure we are not already running under TSX.
                ;
005634                                 .SERR                   ;DON'T DIE ON ERRORS
005642  012700  000176'                MOV     #GTLIN,R0       ;TSX EMT TO GET LINE NUMBER
005646  104375                         EMT     375             ;TRY A TSX EMT
005650  103410                         BCS     1$              ;BR IF NOT UNDER TSX
005652                                 .PRINT  #TSXHD          ;ALREADY UNDER TSX
005660                                 .PRINT  #TSXRUN
005666  000137  004134'                JMP     INISTP
005672                          1$:    .HERR                   ;RENABLE FATAL ERRORS
```

```
                                        ;
                                        ;  Make sure this machine has memory management facilities
                                        ;
005700                                          .TRPSET #AREA,#NOXM     ;CATCH TRAPS
005720  005737  000000G                         TST     @#SROMMR        ;TRY TO ACCESS MEMORY MANAGEMENT REGISTER
005724                                          .TRPSET #AREA,#0        ;Release trap control
                                        ;
                                        ;  Request all available memory from RT-11.
                                        ;
005742                                          .SETTOP #-2             ;REQUEST ALL AVAILABLE MEMORY
005750  010037  000132'                         MOV     R0,TOPMEM       ;REMEMBER WHERE TOP OF MEMORY IS
005754  020027  000000G                         CMP     R0,#VPAR5       ;TSX CANNOT EXTEND ABOVE PAR5 BASE ADDRESS
005760  101402                                  BLOS    3$              ;BR IF RT-11 IS BELOW THAT
005762  012700  000000G                         MOV     #VPAR5,R0       ;SET PAR5 BASE AS UPPER LIMIT ON TSX SIZE
005766  010037  000236'                3$:     MOV     R0,MEMLIM       ;TSX MAY NOT EXCEED THIS UPPER LIMIT
                                        ;
                                        ;  Lock USR in memory for speed
                                        ;  (Set USR to swap over TSEMT to get out of the way)
                                        ;
005772  012705  177776'                         MOV     #TSINIT-2,R5    ;GET THE BASE OF TSINIT
005776                                          .GVAL   #AREA,#374      ;GET SIZE OF RT-11 USR MODULE
006016  160005                                  SUB     R0,R5           ;ALLOCATE SPACE BELOW TSINIT FOR USR
006020  010537  000046                         MOV     R5,@#46         ;SET USR TO SWAP OVER TSEMT
006024                                  5$:     .LOCK                   ;LOCK USR IN MEMORY
                                        ;
                                        ;  Determine if we are to run system with the system debugger
                                        ;
006026  032737  0000000G 0000000G               BIT     #CHAIN,@#JSWLOC ;WERE WE CHAINED TO?
006034  001406                                  BEQ     10$             ;BR IF NOT
006036  023727  000510  057164                 CMP     @#510,#R50ODT   ;SHOULD WE RUN UNDER ODT?
006044  001002                                  BNE     10$             ;BR IF NOT
006046  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.
                                        ;
006052                                  10$:
                                        .IF     NE,PROCID               ;** Do if assembling for pro only **
006052  004737  025354'                         CALL    INSCHK          ;PERFORM VERIFICATION AND DECRYPTION FOR PRO
                                        .ENDC   ;NE,PROCID
                                        ;
                                        ;  Allocate non-initialized buffer space over TSINIT.
                                        ;
006056  012705  000000'                         MOV     #TSINIT,R5      ;Allocate buffer space over TSINIT
006062  004737  011652'                         CALL    ALOCBF          ;Do allocation
006066  004737  012316'                         CALL    ALCSLO          ;Allocate silo buffers for lines
006072  020527  005452'                         CMP     R5,#INITGO      ;Are we beyond code that takes over control?
006076  103002                                  BHIS    12$             ;Br if yes
006100  012705  005452'                         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
006104  010537  0000000G                12$:    MOV     R5,INTSND       ;Ptr to base of interrupt stack
006110  062705  000002                         ADD     #2,R5           ;Always leave last word of stack for flag val
```

```
    115 006114  013701  000000G                 MOV     @#RMON,R1           ;Get pointer to RT-11 RMON base
    116 006120  032761  000000G 000370          BIT     #CW$PRO,370(R1)     ;Are we running on a PRO?
    117 006126  001407                          BEQ     11$                 ;Br if not
    118 006130  105237  000000G                 INCB    PROFLG              ;Set flag saying this is a PRO-350
    119 006134  010537  000150'                 MOV     R5,PROBUF           ;Save pointer to buffer area
    120 006140  062705  001274                  ADD     #PIINSZ,R5          ;Allocate space for buffer
    121 006144  000402                          BR      14$
    122 006146  062705  000000G         11$:    ADD     #INTSSZ,R5          ;Allocate space for interrupt stack
    123 006152  010537  000000G         14$:    MOV     R5,INTSTK           ;Address of top of interrupt stack
    124                                 ;
    125                                 ;  Allocate space for those overlays that go over TSINIT
    126                                 ;
    127 006156  004737  021400'                 CALL    OVLPOS              ;Determine how much space to alloc for overlay
    128                                 ;
    129                                 ;  Note: from this point onward we are carrying the address of the
    130                                 ;  base of the free memory area in R5.
    131                                 ;
    132 006162  020527  025350'                 CMP     R5,#INITOP          ;Have we allocated up to top of TSINIT?
    133 006166  103002                          BHIS    4$                  ;Br if yes
    134 006170  012705  025350'                 MOV     #INITOP,R5          ;Advance to top of TSINIT
    135                                 ;
    136                                 ;  Allocate a 2048. byte work buffer
    137                                 ;
    138 006174  004737  007526'         4$:     CALL    ALCWRK              ;Allocate work buffer
    139                                 ;
    140                                 ;  Allocate empty Region Control Blocks for use by handlers
    141                                 ;
    142 006200  004737  007562'                 CALL    ALCHRB
    143                                 ;
    144                                 ;  If we were started in debug mode, load ODT.
    145                                 ;
    146                                         .IF     EQ,PROCID           ;Don't allow ODT for production PRO version
    147                                         TST     ODTFLG              ;Are we to load system debugger?
    148                                         BEQ     2$                  ;Br if not
    149                                         CALL    GETODT              ;Load ODT and start it
    150                                         .ENDC   ;EQ,PROCID
    151                                 ;
    152                                 ;  Initialize memory management registers for 1-to-1 mapping but
    153                                 ;  leave memory management turned off
    154                                 ;
    155 006204  004737  015230'         2$:     CALL    MEMINI              ;Initialize memory management
    156                                 ;
    157                                 ;  Extract information from RT-11 configuration and sysgen words.
    158                                 ;
    159 006210  013701  000000G                 MOV     @#RMON,R1           ;GET POINTER TO BASE OF RMON
    160 006214  016102  000300                  MOV     300(R1),R2          ;GET RT-11 CONFIGURATION WORD
    161 006220  042702  000000C                 BIC     #CW$GDH+CW$BTH+CW$LGS,R2 ;RESET A FEW FLAGS
    162 006224  052702  000000C                 BIS     #CW$FB+CW$FGJ+CW$USR+CW$XM,R2 ;SET A FEW FLAGS
    163 006230  010237  000000G                 MOV     R2,CONFIG           ;INITIALIZE OUR CONFIGURATION WORD
    164                                 ;  Now get extended configutation word.
    165 006234  016137  000370  000000G         MOV     370(R1),CONFG2      ;EXTENDED CONFIGURATION WORD
    166 006242  052737  000000G 000000G         BIS     #CW$ESP,CONFG2      ;SET EXIT NO SWAP FLAG
    167 006250  123727  000000G 000000G         CMPB    VBUSTP,#QBUS        ;Is this a Q-bus machine?
    168 006256  001003                          BNE     25$                 ;Br if not
    169 006260  052737  000000G 000000G         BIS     #CW$QBS,CONFG2      ;Set QBUS flag
    170                                 ;  And sysgen option word.
    171 006266  016102  000372          25$:    MOV     372(R1),R2
```

```
     172 006272  042702  000000C                 BIC     #SG$ELG+SG$PAR+SG$MTS,R2
     173 006276  052702  000000C                 BIS     #SG$MMU+SG$MTM+SG$IOT+SG$TSX,R2
     174 006302  010237  000000G                 MOV     R2,SYSGEN       ;INITIALIZE OUR SYSGEN WORD
     175                                 ;
     176                                 ;  If a system version number was specified, use it.
     177                                 ;  Else, get version number from RT-11, but limit to default.
     178                                 ;
     179 006306  013700  000000G                 MOV     SYSVER,R0       ;Has user specified version to emulate?
     180 006312  001015                          BNE     30$             ;If so, keep SYSVER
     181 006314  012737  000314' 000156'        MOV     #RTVDEF,RTVPTR  ;If using RT version, set cutoff
     182 006322                                  .GVAL   #AREA,#276      ;GET RT-11 SYSTEM VERSION NUMBER
     183 006342  010037  000000G                 MOV     R0,SYSVER       ;SET AS TSX-PLUS VERSION NUMBER
     184                                 ;
     185                                 ;  Now scan the known version number table and try to locate a match.
     186                                 ;  R0 contains version # in low byte, udpate # in high byte
     187                                 ;
     188 006346  012702  000264'         30$:    MOV     #RTVER,R2       ;Get ptr to first entry in table
     189 006352  120062  000000          31$:    CMPB    R0,RT$VER(R2)   ;Does main version match?
     190 006356  001005                          BNE     32$             ;Br if not
     191 006360  000300                          SWAB    R0              ;Main version matches, get update to low byte
     192 006362  120062  000001                  CMPB    R0,RT$UPD(R2)   ;Does the update match also?
     193 006366  001425                          BEQ     34$             ;Its a match! Use this entry
     194 006370  000300                          SWAB    R0              ;Get SYSVER back to low byte
     195 006372  062702  000003          32$:    ADD     #RTV$SZ,R2      ;If not, step up to the next entry
     196 006376  020227  000322'                 CMP     R2,#RTVEND      ;Past end of table?
     197 006402  103763                          BLO     31$             ;Keep checking if not
     198                                 ;
     199                                 ;  Couldn't find version in tables.  If we picked it up from RT, reset
     200                                 ;  everything to the default.  If it was user-specified, then keep
     201                                 ;  SYSVER, but use last entry ptr.
     202                                 ;
     203 006404  023727  000156' 000322'         CMP     RTVPTR,#RTVEND  ;Was a limit specified? (Got ver. from RT?)
     204 006412  103404                          BLO     33$             ;Br if so
     205                                 ;  Unknown user-specified version, keep user-specified SYSVER,
     206                                 ;  but use ptr to last known version
     207 006414  012737  000317' 000156'        MOV     #<RTVEND-RTV$SZ>,RTVPTR ;Use latest known defaults
     208 006422  000414                          BR      36$
     209                                 ;  Got version from RT, but don't recognize it, reset SYSVER and use defaults
     210                                 ;  RTVPTR was already set to default RTVDEF when we got RT version
     211 006424  113737  000314' 000000G 33$:     MOVB    RTVDEF+RT$VER,SYSVER ;Set SYSVER to default
     212 006432  113737  000315' 000000G         MOVB    RTVDEF+RT$UPD,SYSUPD ;and update
     213 006440  000405                          BR      36$
     214                                 ;
     215                                 ;  Version was identified in tables.  RTVPTR contains limiting version ptr
     216                                 ;  (RTVDEF if got vers from RT, RTVEND if user specified version)
     217                                 ;
     218 006442  020237  000156'         34$:    CMP     R2,RTVPTR       ;Is it past limit?
     219 006446  103366                          BHIS    33$             ;If so, keep limit, and go limit SYSVER
     220 006450  010237  000156'                 MOV     R2,RTVPTR       ;If not, use what we found
     221                                 ;
     222                                 ;  Now set some information that depends on the emulated version number
     223                                 ;
     224 006454  105737  000000G         36$:    TSTB    CLVERS          ;Use table value for CL version number?
     225 006460  001005                          BNE     38$             ;Br if not, use value supplied in TSGEN
     226 006462  013702  000156'                 MOV     RTVPTR,R2       ;Get ptr to version table
     227 006466  116237  000002  000000G         MOVB    CL$VER(R2),CLVERS ;Auto set CLVERS from version table
     228 006474  105737  000000G         38$:    TSTB    LDVERS          ;Auto-select LD translation table type?
```

```
    229 006500  001011                          BNE     39$             ;Br if not, use value supplied in TSGEN
    230 006502  112737  000001  000000G         MOVB    #1,LDVERS       ;Assume old translation table format
    231 006510  023727  000156' 000314'         CMP     RTVPTR,#RT54    ;At or beyond 5.4?
    232 006516  103402                          BLO     39$             ;Br if not, retain old format
    233 006520  105237  000000G                 INCB    LDVERS          ;LD translation table format changed at 5.4
    234 006524                          39$:
```

```
                                        ;
                                        ;  Set up a few clock constants based on clock frequency.
                                        ;  See if we have a 50 or 60 Hz clock
                                        ;
006524  032737  000000G 000000G INICLK: BIT     #CW$50H,CONFIG  ;50 or 60 Hz clock?
006532  001017                          BNE     2$              ;Br if 50 Hz
                                        ;
                                        ;  60 Hz clock
                                        ;
006534  012737  000074  000000G         MOV     #60.,TK1SEC     ;Clock ticks per 1 second
006542  012737  000036  000000G         MOV     #30.,TK5VAL     ;Clock ticks per 0.5 seconds
006550  012737  000264  000000G         MOV     #180.,TK3SVL    ;Clock ticks per 3 seconds
006556  012737  000006  000000G         MOV     #6.,TK1VAL      ;Clock ticks per 0.1 seconds
006564  012700  001130                  MOV     #600.,R0        ;Get # clock ticks per 10 seconds
006570  000416                          BR      8$
                                        ;
                                        ;  50 Hz clock
                                        ;
006572  012737  000062  000000G 2$:     MOV     #50.,TK1SEC     ;Clock ticks per 1 second
006600  012737  000031  000000G         MOV     #25.,TK5VAL     ;Clock ticks per 0.5 seconds
006606  012737  000226  000000G         MOV     #150.,TK3SVL    ;Clock ticks per 3 seconds
006614  012737  000005  000000G         MOV     #5.,TK1VAL      ;Clock ticks per 0.1 seconds
006622  012700  000764                  MOV     #500.,R0        ;Get # clock ticks per 10 seconds
                                        ;
                                        ;  Set number of clock ticks per day
                                        ;
006626  012702  020700          8$:     MOV     #8640.,R2       ;(# seconds per day) / 10.
006632  070200                          MUL     R0,R2           ;Get # clock ticks per day
006634  010237  000000G                 MOV     R2,DATIMH       ;High-order value
006640  010337  000000G                 MOV     R3,DATIML       ;Low-order value
                                ;
                                ;  Do a fast check to make sure specified T/S line addresses are ok.
                                ;
006644  004737  005260'         CKLIN:  CALL    LINCHK          ;CHECK T/S LINE ADDRESSES
                                ;
                                ;  Do PRO-350 system initialization
                                ;
                                .IF     NE,PROASM
006650  032737  000000G 000000G         BIT     #CW$PRO,CONFG2  ;Are we running on a PRO-350?
006656  001405                          BEQ     INIDEV          ;Br if not
006660  004737  000000G                 CALL    PROINI          ;Do PRO-350 initialization
006664  012737  000000G 000000G         MOV     #PIDRIV,PIDPTR  ;Set up pointer to clock driven PI routine
                                .ENDC   ;NE,PROASM
                                ;
                                ;  Make entry in device handler table for TT device.
                                ;
006672  012737  100040  000000G INIDEV: MOV     #R50TT,PNAME    ;PERMANENT NAME "TT"
006700  012737  000000G 000000G         MOV     #DI$TT,DVSTAT   ;SET DEVICE STATUS FLAGS FOR TT
006706  005037  000000G                 CLR     DVFLAG
006712  005037  000000G                 CLR     DEVSIZ
006716  012737  000002  000000G         MOV     #2,HANENT       ;SET UP HANENT SO HANDLER LOOKS RESIDENT
006724  005037  000000G                 CLR     NUMDEV          ;IT IS DEVICE # 0
                                ;
                                ;  Make device table entry for LD (logical disk) device
                                ;
006730  012737  177777  000000G         MOV     #-1,LDDEVX      ;ASSUME LD SUPPORT NOT WANTED
006736  105737  000000G                 TSTB    VLDSYS          ;IS LD SUPPORT GENNED IN?
```

```
                                        BEQ     6$                  ;BR IF NOT
                                        ADD     #2,NUMDEV           ;ONE MORE DEVICE
                                        MOV     NUMDEV,R1           ;GET DEVICE TABLE INDEX
                                        MOV     R1,LDDEVX           ;REMEMBER INDEX NUMBER FOR LD DEVICE
                                        MOV     #R50LD,PNAME(R1) ;SET DEVICE NAME ("LD")
                                        MOV     #<DS$DIR+DS$SFN+DS$VSZ+DI$LD>,DVSTAT(R1);SET DEV STATUS FLAGS
                                        MOV     #DX$EBA,DVFLAG(R1);Say buffers must be on even byte boundaries
                                        CLR     DEVSIZ(R1)
                                        MOV     #2,HANENT(R1)       ;SAY HANDLER IS RESIDENT
                                ;
                                ;  Make device table entry for CL (communications line) device
                                ;
                        6$:     TST     #CLTOTL             ;Are there any communications lines?
                                BEQ     8$                  ;Br if not
                                CALL    CLINIT              ;Initialize CL handler
                                ;
                                ;  Disable clock interrupts.
                                ;
                        8$:     MOV     #2,@#0              ;LOAD RTI IN LOCATION 0
                                CLR     @#CLKVEC            ;ATTACH CLOCK INTERRUPT TO 0
                                BIT     #CW$PRO,CONFG2      ;ARE WE RUNNING ON A PRO?
                                BEQ     1$                  ;BR IF NOT
                                CLR     @#230               ;380 CLOCK INTERRUPT VECTOR
                                ;
                                ;  Set up memory parity control
                                ;
                        1$:     CALL    PARSET              ;SET UP MEMORY PARITY CONTROL
                                ;
                                ;  Determine how much memory is installed on machine
                                ;
                                CALL    MEMTST              ;FIND OUT HOW MUCH PHYSICAL MEMORY THERE IS
                                ;
                                ;  Set up information about the size of the job context area
                                ;
                                CALL    CXTALC              ;Determine size of job context area
                                ;
                                ;  Load TSX-Plus device handlers that go in low memory
                                ;
                                CALL    GETHNL              ;Load low memory handlers
                                ;
                                ;  Reserve space for interrupt vector intercept routines for mapped handlers
                                ;
                                MOV     R5,XMVBAS           ;Save address of base of area for XM vectors
                                MOV     NMXHAN,R1           ;Get # mapped handlers
                                ASL     R1                  ;Reserve room for 2 interrupts per handler
                                ADD     #NXIVMH,R1          ;Add # requested extra interrupt vectors
                                MUL     #MPIVSZ,R1          ;Calc space needed for interrupt entry code
                                ADD     R1,R5               ;Advance the address of free memory
                                ;
                                ;  Set up device index number and unit number for "SY:" device.
                                ;
                                CALL    SETSY               ;SET UP INFO ABOUT SY DEVICE
                                ;
                                ;  Open channel to TSKMON and set up information about it.
                                ;
                                CALL    OPNKMN              ;OPEN CHANNEL TO TSKMON
                                ;
```

```
                                        ;  Set up information about the IND program
                                        ;
007132  004737  014110'                       CALL    INDINI          ;INITIALIZE FOR IND PROGRAM
                                        ;
                                        ;  Initialize the TSXUCL data file
                                        ;
007136  004737  014652'                       CALL    UCLINI
                                        ;
                                        ;  Set name of device that UCL program is to be run from
                                        ;
007142  013737  000000G 000000G               MOV     SYNAME,UCLNAM   ;SET DEVICE NAME FOR UCL PROGRAM
                                        ;
                                        ;  Initialize spooling system
                                        ;
007150  004737  010522'                       CALL    SPLINI          ;INITIALIZE SPOOLING SYSTEM
                                        ;
                                        ;  Open system swap file
                                        ;
007154  105737  000000G                       TSTB    VSWPFL          ;IS JOB SWAPPING ALLOWED?
007160  001402                               BEQ     3$              ;BR IF NOT
007162  004737  007564'                       CALL    OPNSWP          ;OPEN THE SYSTEM SWAP FILE
                                        ;
                                        ;  Open swap file used for PLAS regions
                                        ;
007166  004737  010216'                3$:    CALL    OPNRSF          ;Open PLAS region swap file
                                        ;
                                        ;  Set up information about which devices need to have their I/O mapped
                                        ;
007172  004737  021376'                       CALL    SETMIO          ;Set up information about mapped I/O
                                        ;
                                        ;  We are finished allocating low-memory buffer space.
                                        ;
007176  010500                               MOV     R5,R0           ;ENSURE WE DON'T OVERFLOW 40KB
007200  004737  025104'                       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.
                                        ;
007204  010537  000000G                       MOV     R5,UMSYTP       ;SAVE ADDRESS OF NON-EXTENDED SYSTEM TOP
007210  062705  000077                       ADD     #77,R5          ;BOUND UP TO 64-BYTE BOUNDARY
007214  072527  177772                       ASH     #-6,R5          ;CONVERT TO 64-BYTE BLOCK #
007220  042705  176000                       BIC     #176000,R5      ;KILL SIGN EXTENSION
                                        ;
                                        ;  Allocate buffer space that is not contrained to 40Kb TSX-Plus region.
                                        ;
007224  004737  012536'                       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.
                                        ;
007230  010504                               MOV     R5,R4           ;Save the base of free memory in R4
007232  010437  000136'                       MOV     R4,FMEMLO       ;Save pointer above top of alloc low memory
007236  013705  000134'                       MOV     FMEMHI,R5       ;Get 64-byte block # of free high memory
                                        ;
                                        ;  Load any mapped system code
                                        ;
007242  004737  022010'                       CALL    GETMAP          ;LOAD USR, EMT, MSG, LOCK, SPOOL, etc.
```

```
                                        ;
                                        ;  Load any shared run-time systems
                                        ;
007246  005727  000000G                       TST     #NUMRDB         ;Do we need to load any shared run-times?
007252  001415                               BEQ     4$              ;Br if not
007254  012701  000000G                       MOV     #RDB,R1         ;Point to 1st run-time descriptor block
007260  010502                               MOV     R5,R2           ;Save initial memory pointer
007262  004737  023472'               5$:     CALL    GETSRT          ;Load a shared run-time system
007266  062701  000000G                       ADD     #RT$$SZ,R1      ;Point to next shared run-time descriptor
007272  020127  000000G                       CMP     R1,#RDBEND      ;Are there more to load?
007276  103771                               BLO     5$              ;Br if yes
007300  160502                               SUB     R5,R2           ;Compute amt of space used by run-times
007302  010237  000000G                       MOV     R2,SRTSIZ       ;Save total run-time size
007306                                  4$:
                                        .IF     NE,PROASM
                                        ;
                                        ;  If we are running on a Pro, load the PI handler like a shared run-time
                                        ;
007306  105737  000000G                       TSTB    PROFLG          ;Are we running on a Pro?
007312  001410                               BEQ     10$             ;Br if not
007314  012701  000244'                       MOV     #PISRT,R1       ;Point to dummy shared run-time block for PI
007320  010502                               MOV     R5,R2           ;Save current memory pointer
007322  004737  023472'                       CALL    GETSRT          ;Load PI handler like a shared run-time
007326  160502                               SUB     R5,R2           ;Calculate amt of space used by PI handler
007330  060237  000000G                       ADD     R2,MHNSIZ       ;Count in mapped-handler size
                                        .ENDC   ;NE,PROASM
                                        ;
                                        ;  Load any mapped handlers
                                        ;
007334  004737  017634'               10$:    CALL    GETHNH          ;Load mapped handlers
                                        ;
                                        ;  Allocate space for data cache buffers and control tables
                                        ;
007340  004737  024104'                       CALL    CSHBUF          ;Allocate space for data cache
                                        ;
                                        ;  We have finished allocating all of the memory used by the system.
                                        ;  Allocate and initialize a memory map table that will be used to
                                        ;  show which pages are available for user jobs.
                                        ;
007344  004737  016006'                       CALL    MAPALC          ;Allocate memory map table
                                        ;
                                        ;  Set up info about maximum memory space available to jobs
                                        ;
007350  004737  016172'                       CALL    SETJSZ          ;SET JOB SIZE INFO
                                        ;
                                        ;  Set up date and time
                                        ;
007354  013702  000000G                       MOV     SYTIML,R2       ;Save time we got at start of init
007360                                        .GTIM   #AREA,#SYTIMH   ;SET TIME OF DAY
007400                                        .DATE                   ;GET DATE
007406  010037  000000G                       MOV     R0,SYSDAT       ;SET SYSTEM DATE
007412  020237  000000G                       CMP     R2,SYTIML       ;Make sure some time has elapsed
007416  001010                               BNE     11$             ;Br if clock is running
007420                                        .PRINT  #TSXHD          ;Print error message heading
007426                                        .PRINT  #NOCLOK         ;Print clock-not-working message
007434  000137  004134'                       JMP     INISTP          ;Abort initialization
                                        ;
```

```
                                        ;  Unlock the USR so that TSEMT will be swapped back in.
                                        ;
007440                                  11$:    .UNLOCK                 ;RELEASE USR
                                        ;
                                        ;  Read back into memory that part of the resident portion of TSX
                                        ;  that we overlayed with our work buffer.
                                        ;
007442  013702  000154'                         MOV     WRKSIZ,R2       ;Get size of work buffer
007446  006202                                  ASR     R2              ;Convert to # words
007450  013703  000152'                         MOV     WRKBUF,R3       ;Get address of work buffer area
007454  000241                                  CLC                     ;Convert to block # in TSX.SAV file
007456  006003                                  ROR     R3
007460  000303                                  SWAB    R3
007462                                          .READW  #AREA,#17,WRKBUF,R2,R3 ;Read back TSX over work buffer
                                        ;
                                        ;  See if user requested control-C abort
                                        ;
007516  004737  025144'                         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
                                        ;
007522  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).
                                ;
007526  010246                  ALCWRK: MOV     R2,-(SP)
                                ;
                                ;  Get address of start of area where buffer can go and then bound
                                ;  up to a block boundary.
                                ;
007530  012702  000000G                 MOV     #EXCBUF,R2      ;Get address of base of buffer area
007534  062702  000777                  ADD     #777,R2         ;Bound up to block boundary
007540  042702  000777                  BIC     #777,R2         ;Set to block boundary
007544  010237  000152'                 MOV     R2,WRKBUF
007550  012737  004000  000154'         MOV     #2048.,WRKSIZ
                                ;
                                ;  Finished
                                ;
007556  012602                          MOV     (SP)+,R2
007560  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.
                ;
                ALCHRB: MOV     R2,-(SP)
                ;
                ;  Get count of # RCB's to build
                ;
                        MOV     NDVRCB,R0       ;Get # RCB's to build for handlers
                ;
                ;  Store pointer to start of RCB area and store -1 at beginning of area
                ;
                        MOV     R5,HANRCB       ;Start of RCB area
                        MOV     R5,HANRCO       ;Store offset relative to MONVEC
                        SUB     #MONVEC,HANRCO  ;Convert address to offset
                        MOV     #-1,(R5)+       ;Store -1 at start of area
                ;
                ;  Allocate and initialize to zero the RCB's
                ;
                1$:     MOV     #5.,R2          ;Each RCB has 5 words
                2$:     CLR     (R5)+           ;Zero the RCB
                        SOB     R2,2$
                ;
                ;  See if there are more RCB's to build
                ;
                        DEC     R0              ;More RCB's to initialize?
                        BGT     1$              ;Loop if yes
                ;
                ;  Store -1 at end of RCB area
                ;
                        MOV     #-1,(R5)+       ;Mark end of RCB list
                ;
                ;  Finished
                ;
                        MOV     (SP)+,R2
                        RETURN

                        .IFF    ;NE,<PROASM-1>  ;Dummy for Pro-only
007562  000207  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 $DEAD flag for the line if INIABT=0.
                ;
                LINCHK: MOV     R1,-(SP)
                        MOV     R2,-(SP)
                        MOV     R3,-(SP)
                        MOV     R4,-(SP)
                        MOV     @#4,-(SP)       ;SAVE ORIGINAL TRAP VECTOR
                ;
                ;  Take over trap control
                ;
                        MOV     #6$,@#4         ;CATCH TRAPS
                ;
                ;  Loop through the test for each line.
                ;
                        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.
                ;
                1$:     CALL    LINTYP          ;Determine the type of this line
                        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
                        BEQ     31$             ;Br if not
                        MOV     LMXNUM(R1),R3   ;IS THIS A DL-11 OR MULTIPLEXER LINE?
                        BEQ     2$              ;BR IF DL-11
                        MOV     MXCSR(R3),R2    ;GET DZ11 OR DH11 STATUS REGISTER ADDRESS
                        BEQ     11$             ;BR IF ALREADY MARKED AS DEAD
                        MOV     MXVEC(R3),R4    ;GET MUX INTERRUPT VECTOR ADDRESS
                        BR      3$
                11$:    CALL    4$              ;MARK LINE AS DEAD
                        BR      31$             ;CONTINUE CHECKING TERMINALS
                2$:     MOV     RSR(R1),R2      ;GET DL-11 STATUS REGISTER ADDRESS
                        MOV     INVEC(R1),R4    ;GET DL-11 INTERRUPT VECTOR ADDRESS
                ;  Check validity of status register address.
                3$:     CMP     R2,#160000      ;IS IT IN I/O PAGE?
                        BLO     LINTRP          ;ERROR IF NOT
                        BIT     #7,R2           ;IS IT ON 8-BYTE BOUNDARY?
                        BNE     LINTRP          ;ERROR IF NOT
                        TST     @R2             ;TRY TO ACCESS IT AND SEE IF WE TRAP
                ;  Check validty of interrupt vector address.
                        CMP     R4,#60          ;CAN'T BE BELOW 60
                        BLO     BADVEC
                        CMP     R4,#500         ;OR ABOVE 500
                        BHIS    BADVEC
                        BIT     #7,R4           ;MUST BE ON 8-BYTE BOUNDARY
                        BNE     BADVEC
                ;  This line looks good.  Check next.
                31$:    SUB     #2,R1           ;MORE TO CHECK?
                        BGT     1$              ;BR IF YES
                ;
                ;  Finished -- all lines look ok.
                ;
```

```
                        MOV     (SP)+,@#4       ;RESTORE TRAP VECTOR
                        MOV     (SP)+,R4
                        MOV     (SP)+,R3
                        MOV     (SP)+,R2
                        MOV     (SP)+,R1
                        RETURN
                ;
                ;  See if we should abort or just mark the line as dead.
                ;
4$:             TSTB    VINABT          ;DOES HE WANT TO ABORT?
                BNE     LINTRP          ;YES
                BIS     #$DEAD,LSW3(R1) ;MARK LINE AS DEAD
                TST     R3              ;IS THIS A DL11 OR A MUX LINE?
                BEQ     5$              ;BR IF DL11
                CLR     MXCSR(R3)       ;MARK DZ OR DH AS DEAD
5$:             RETURN

                ;
                ;  Trap occured while trying to access status register.
                ;
6$:             CALL    4$              ;REPORT ERROR OR MARK AS DEAD LINE
                RTI                     ;RETURN TO LINE CHECKING
                ;
                ;  Error: Invalid status register address.
                ;  R1 = Line number,  R2 = status register address
                ;
LINTRP:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
                .PRINT  #BADLIN
                BR      ERP
                ;
                ;  Error: Invalid interrupt vector address.
                ;  R1 = Line number,  R4 = interrupt vector address
                ;
BADVEC:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
                .PRINT  #BDVMSG
                MOV     R4,R2           ;GET VECTOR ADDRESS TO R2
ERP:            MOV     R2,R0           ;GET ADDRESS TO R0
                CALL    PRTOCT          ;PRINT OCTAL VALUE
                .PRINT  #CRLF
                .PRINT  #BDLMSG         ;LINE # =
                MOV     R1,R0           ;GET LINE NUMBER
                ASR     R0              ;# 1
                CALL    PRTDEC          ;PRINT LINE NUMBER
                .PRINT  #CRLF
                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.
                                ;
007564  010146                  OPNSWP: MOV     R1,-(SP)
007566  010246                          MOV     R2,-(SP)
007570  010346                          MOV     R3,-(SP)
                                ;
                                ;  Load RT-11 handler for swap device.
                                ;
007572  013700  000000G                 MOV     SWDBLK,R0       ;Get name of device
007576  004737  024770'                 CALL    RTFTCH          ;Fetch the RT-11 handler
007602  103546                          BCS     11$             ;Br if invalid device
                                ;
                                ;  Compute the maximum number of slots in swap file that we could need
                                ;
007604  012703  000000C                 MOV     #NSL+NDL,R3     ;Get # virtual lines and detached jobs
007610  012701  000000G                 MOV     #LSTPL,R1       ;Get index to last primary line
007614  032761  000000G 000000G 1$:     BIT     #$DEAD,LSW3(R1) ;Is this line installed?
007622  001001                          BNE     5$              ;Br if not
007624  005203                          INC     R3              ;Count another primary line
007626  162701  000002          5$:     SUB     #2,R1           ;Get next line index
007632  003370                          BGT     1$              ;Loop if more lines to check
007634  020337  000000G                 CMP     R3,VSWPSL       ;Compare with # slots specified
007640  002002                          BGE     6$              ;Br if VSWPSL value is ok
007642  010337  000000G                 MOV     R3,VSWPSL       ;Reduce number of slots in swap file
                                ;
                                ;  Determine how many blocks are needed for each slot in swap file.
                                ;
007646  013703  000000G          6$:     MOV     VHIMEM,R3       ;GET # BLOCKS NEEDED FOR LARGEST JOB SIZE
007652  006303                          ASL     R3
007654  063703  000000G                 ADD     CXTPAG,R3       ;ADD # BLOCKS NEEDED FOR JOB CONTEXT AREA
007660  010337  000000G                 MOV     R3,SLTSIZ       ;Save size of swap file slot
                                ;
                                ;  Compute the total number of blocks needed for the swap file.
                                ;
007664  070337  000000G                 MUL     VSWPSL,R3       ;Multiply by # slots in swap file
                                ;
                                ;  R3 now contains total number of blocks needed in swap file.
                                ;  See if swap file already exists on disk.
                                ;
007670                          4$:     .LOOKUP #AREA,#1,#SWDBLK;DOES SWAP FILE EXIST NOW?
007710  103415                          BCS     2$              ;BR IF NOT
                                ;  Swap file exists.  See if it is the right size.
007712  020003                          CMP     R0,R3           ;IS SWAP FILE THE RIGHT SIZE?
007714  001453                          BEQ     3$              ;BR IF YES
                                ;  Old swap file is not of correct size.
                                ;  Delete it and open a new swap file.
007716                                  .CLOSE  #1
```

```
 58 007724                                      .DELETE #AREA,#1,#SWDBLK;DELETE THE OLD SWAP FILE
 59                                     ;
 60                                     ;  Create a new swap file.
 61                                     ;
 62 007744                              2$:     .ENTER  #AREA,#1,#SWDBLK,R3 ;CREATE A NEW SWAP FILE
 63 007770  103443                              BCS     9$                  ;BR IF SOME ERROR ON OPEN
 64                                     ;
 65                                     ;  Swap file has been created.
 66                                     ;  Write to last block to reserve full space in file then close
 67                                     ;  and reopen the channel using a .lookup.
 68                                     ;
 69 007772  005303                              DEC     R3                  ;GET # OF LAST BLOCK IN FILE
 70 007774                                      .WRITW  #AREA,#1,#TSINIT,#256.,R3 ;WRITE TO LAST BLOCK IN FILE
 71 010032  005203                              INC     R3                  ;GET BACK # BLOCKS IN FILE
 72 010034                                      .CLOSE  #1                  ;CLOSE FILE WE CREATED
 73 010042  000712                              BR      4$                  ;NOW GO REOPEN USING A .LOOKUP
 74                                     ;
 75                                     ;  Swap file has been successfully opened using a .lookup.
 76                                     ;  Now copy channel status to TSX swap channel.
 77                                     ;
 78 010044  012700  000000G             3$:     MOV     #SWPCHN,R0          ;POINT TO SWAP CHANNEL BLOCK
 79 010050  013702  000000G                     MOV     SWDBLK,R2           ;GET DEVICE NAME
 80 010054  004737  024410'                     CALL    SETCHN              ;SET UP SWAP CHANNEL INFO
 81                                     ;
 82                                     ;  Release the RT-11 device handler
 83                                     ;
 84 010060                                      .RELEAS #SWDBLK             ;Release RT-11 device handler
 85                                     ;
 86                                     ;  Finished
 87                                     ;
 88 010070  012603                              MOV     (SP)+,R3
 89 010072  012602                              MOV     (SP)+,R2
 90 010074  012601                              MOV     (SP)+,R1
 91 010076  000207                              RETURN
 92                                     ;
 93                                     ;  Error: Cannot open swap file
 94                                     ;
 95 010100                              9$:     .PRINT  #TSXHD              ;PRINT ERROR MESSAGE
 96 010106                                      .PRINT  #BADOPN
 97 010114  004737  010142'                     CALL    SPNEED              ;Print info about number of blocks needed
 98                                     ;
 99                                     ;  Error: Invalid device specification.
100                                     ;
101 010120  010001                      11$:    MOV     R0,R1               ;Save device name
102 010122                                      .PRINT  #TSXHD              ;Print error message
103 010130                                      .PRINT  #BADOPN
104 010136  004737  010170'                     CALL    BADDEV              ;Print invalid device specification
105                                    ;
106                                    ;  Error: Number of contiguous blocks required.
107                                    ;
108 010142                             SPNEED: .PRINT  #CONSPC             ;Print contiquous blocks needed
109 010150  010300                             MOV     R3,R0               ;GET # BLOCKS NEEDED FOR FILE
110 010152  004737  025230'                    CALL    PRTDEC              ;DISPLAY # BLOCKS NEEDED
111 010156                                     .PRINT  #CRLF
112 010164  000137  004134'                    JMP     INISTP              ;ABORT INITIALIZATION
113                                    ;
114                                    ;  Bad file specification.
```

```
                                        ;
010170                          BADDEV: .PRINT  #CFHMSG         ;Print invalid device specification
010176  010100                          MOV     R1,R0           ;Get the rad50 device name
010200  004737  025274'                 CALL    PRTR50          ;Print rad50 device name
010204                                  .PRINT  #CRLF           ;Print carriage return/line feed
010212  000137  004134'                 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.
                                ;
010216  010346                  OPNRSF: MOV     R3,-(SP)
                                ;
                                ;  Return if this is a non-swapping system or if region swap file is
                                ;  not wanted.
                                ;
010220  105737  000000G                 TSTB    VSWPFL          ;Is this a non-swapping system?
010224  001513                          BEQ     9$              ;Br if yes
010226  005737  000000G                 TST     VPLAS           ;Is a PLAS swap file wanted?
010232  001510                          BEQ     9$              ;Br if not
                                ;
                                ;  Load RT-11 device handler for swap device
                                ;
010234  013700  000000G                 MOV     RSFBLK,R0       ;Get name of device
010240  004737  024770'                 CALL    RTFTCH          ;Try to fetch the RT-11 device handler
010244  103515                          BCS     11$             ;Br if error on handler fetch
010246  013703  000000G                 MOV     VPLAS,R3        ;Get # blocks in PLAS swap file
                                ;
                                ;  See if PLAS swap file already exists on disk
                                ;
010252                          4$:     .LOOKUP #AREA,#1,#RSFBLK ;Try to find existing PLAS swap file
010272  103416                          BCS     2$              ;Br if file does not now exist
                                ;
                                ;  PLAS swap file exists.
                                ;  See if it is the correct size.
                                ;
010274  020037  000000G                 CMP     R0,VPLAS        ;Is swap file of the correct size?
010300  001453                          BEQ     3$              ;Br if yes
                                ;
                                ;  Old PLAS swap file is not of correct size.
                                ;  Delete it and open a new swap file.
                                ;
010302                                  .CLOSE  #1              ;Close and delete the old file
010310                                  .DELETE #AREA,#1,#RSFBLK;Delete the old file
                                ;
                                ;  Create new swap file
                                ;
010330                          2$:     .ENTER  #AREA,#1,#RSFBLK,VPLAS ;Create a new PLAS swap file
010356  103440                          BCS     10$             ;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.
                                ;
010360  005303                          DEC     R3              ;Get # of last block in file
010362                                  .WRITW  #AREA,#1,#TSINIT,#256.,R3
010420                                  .CLOSE  #1
010426  000711                          BR      4$              ;Go back and lookup file
```

```
                                        ;
                                        ;  Swap file has been successfully opened using lookup.
                                        ;  Copy channel status to TSX channel block.
                                        ;
010430  012700  000000G         3$:     MOV     #SEGCHN,R0      ;Point to TSX PLAS swap channel
010434  013703  000000G                 MOV     RSFBLK,R3       ;Get device name
010440  004737  024410'                 CALL    SETCHN          ;Set up TSX channel block
                                        ;
                                        ;  Release the RT-11 device handler
                                        ;
010444                                  .RELEAS #RSFBLK         ;Release RT-11 device handler
                                        ;
                                        ;  Finished
                                        ;
010454  012603                  9$:     MOV     (SP)+,R3
010456  000207                          RETURN
                                        ;
                                        ;  Error -- Cannot open PLAS swap file
                                        ;
010460                          10$:    .PRINT  #TSXHD          ;Print error prefix
010466                                  .PRINT  #RSFERR         ;Print error message
010474  004737  010142'                 CALL    SPNEED          ;Print information about amt of space needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
010500  010001                  11$:    MOV     R0,R1           ;Save device name
010502                                  .PRINT  #TSXHD          ;Print error message
010510                                  .PRINT  #RSFERR
010516  004737  010170'                 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.
                                ;
 010522  005727  000000G        SPLINI: TST     #SPLND          ;Are there any spooled devices?
 010526  001401                         BEQ     13$             ;Br if not
 010530  000401                         BR      10$             ;Initialize the spooled devices
 010532  000207                 13$:    RETURN
                                ;
                                ;  There are some spooled devices
                                ;
 010534  010146                 10$:    MOV     R1,-(SP)
 010536  010246                         MOV     R2,-(SP)
 010540  010346                         MOV     R3,-(SP)
 010542  010446                         MOV     R4,-(SP)
 010544  010546                         MOV     R5,-(SP)
                                ;
                                ;  Open each spooled device
                                ;
 010546  105037  000000G                CLRB    NSPLDV          ;INIT COUNT OF # ACTUAL SPOOLED DEVICES
 010552  012701  000000G                MOV     #SDCB,R1        ;POINT TO 1ST SDCB
 010556  012703  000000G                MOV     #SPLDEV,R3      ;POINT TO TABLE OF RAD50 DEV NAMES
 010562  012704  000000G                MOV     #SPLANM,R4      ;POINT TO TABLE OF ASCII DEV NAMES
 010566  004737  011574'         2$:    CALL    FORCE0          ;FORCE UNIDENTIFIED UNIT #S TO 0
 010572  011302                         MOV     (R3),R2         ;GET RAD50 NAME OF SPOOLED DEVICE
 010574  010261  000000G                MOV     R2,SDNAME(R1)   ;SET NAME IN SDCB
 010600  010100                         MOV     R1,R0           ;GET ADDRESS OF SDCB
 010602  062700  000000G                ADD     #SDANAM,R0      ;POINT TO CELL FOR ASCII NAME
 010606  112420                         MOVB    (R4)+,(R0)+     ;MOVE IN ASCII DEVICE NAME
 010610  112420                         MOVB    (R4)+,(R0)+
 010612  112420                         MOVB    (R4)+,(R0)+
 010614  020227  000000G                CMP     R2,#DMYDEV      ;Is this a dummy entry for later patching?
 010620  001451                         BEQ     1$              ;Br if yes -- Ignore it
 010622  010200                         MOV     R2,R0           ;Get name to R0
 010624  004737  011476'                CALL    CVTDVU          ;Convert name to device # and unit #
 010630  010061  000000G                MOV     R0,SDDVU(R1)    ;Store device # and unit # in SDCB
 010634  010200                         MOV     R2,R0           ;Get device name
 010636  004737  011372'                CALL    CHKCLD          ;See if this is a CL device?
 010642  103406                         BCS     14$             ;Br if not
 010644  004737  011306'                CALL    SPLCLD          ;Set up for spooling to CL device
 010650  103414                         BCS     3$              ;Br if invalid unit
 010652  105237  000000G                INCB    NSPLDV          ;Count # of actual spooled devices
 010656  000432                         BR      1$              ;Process next device
 010660  010100                 14$:    MOV     R1,R0           ;Get address of SDCB
 010662  062700  000000G                ADD     #SDCHAN,R0      ;Point to channel block within SDCB
 010666  004737  024322'                CALL    OPNCHN          ;Set TSX-Plus channel block open to device
 010672  103403                         BCS     3$              ;Br if did not recognize device
 010674  105237  000000G                INCB    NSPLDV          ;Count # actual spooled devices
 010700  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.
                                ;
 010702  012761  000000G 000000G 3$:     MOV     #DMYDEV,SDNAME(R1);SAY THIS DEVICE IS NOT SPOOLED
```

```
     58 010710  105737  000000G                 TSTB    VINABT          ;ABORT OR CONTINUE ON ERRORS?
     59 010714  001413                          BEQ     1$              ;BR TO IGNORE DEVICE AND CONTINUE INIT
     60 010716                                  .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     61 010724                                  .PRINT  #BDSPOP
     62 010732  010200                          MOV     R2,R0           ;GET RAD50 DEVICE NAME
     63 010734  004737  025274'                 CALL    PRTR50          ;PRINT DEVICE NAME
     64 010740  000137  004134'                 JMP     INISTP          ;ABORT INITIALIZATION
     65                                 ;
     66                                 ;  Process next spooled device
     67                                 ;
     68 010744  062701  000000G         1$:     ADD     #SDCBSZ,R1      ;POINT TO NEXT SDCB
     69 010750  005723                          TST     (R3)+           ;POINT TO NEXT DEVICE NAME
     70 010752  020327  000000G                 CMP     R3,#SPLDVN      ;OPENED ALL SPOOLED DEVICES?
     71 010756  103703                          BLO     2$              ;BR IF MORE TO DO
     72                                 ;
     73                                 ;  Open the spool file
     74                                 ;
     75 010760  105737  000000G                 TSTB    NSPLDV          ;ARE THERE ANY ACTUAL SPOOLED DEVICES?
     76 010764  001521                          BEQ     12$             ;BR IF THERE ARE NO ACTUAL SPOOLED DEVICES
     77 010766  013700  000000G                 MOV     SPLBLK,R0       ;Get name of device for spool file
     78 010772  004737  024770'                 CALL    RTFTCH          ;Fetch the RT-11 device handler
     79 010776  103532                          BCS     11$             ;Br if cannot fetch handler
     80 011000  013702  000000G                 MOV     NSPLBL,R2       ;GET # BLOCKS TO ALLOCATE FOR FILE
     81 011004  005202                          INC     R2              ;Add 1 extra block
     82                                 ;
     83                                 ;  See if spool file already exists
     84                                 ;
     85 011006                          5$:     .LOOKUP #AREA,#1,#SPLBLK;SEE IF SPOOL FILE ALREADY EXISTS
     86 011026  103415                          BCS     6$              ;BR IF IT DOES NOT EXIST
     87 011030  020002                          CMP     R0,R2           ;IS IT THE RIGHT SIZE?
     88 011032  001453                          BEQ     7$              ;BR IF YES
     89 011034                                  .CLOSE  #1              ;IT EXISTS BUT IS OF WRONG SIZE
     90 011042                                  .DELETE #AREA,#1,#SPLBLK;DELETE CURRENT FILE AND OPEN NEW ONE
     91                                 ;
     92                                 ;  Open new spool file
     93                                 ;
     94 011062                          6$:     .ENTER  #AREA,#1,#SPLBLK,R2;CREATE A NEW SPOOL FILE
     95 011106  103456                          BCS     8$              ;BR IF ERROR ON ENTER
     96                                 ;  Write to last block in file to reserve full file space
     97 011110  010203                          MOV     R2,R3           ;Get # of blocks in file
     98 011112  005303                          DEC     R3              ;Get # of last block in file
     99 011114                                  .WRITW  #AREA,#1,#TSINIT,#256.,R3
    100                                 ;  Now close and reopen using a lookup
    101 011152                                  .CLOSE  #1              ;CLOSE SPOOL FILE
    102 011160  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 011162  012700  000000G         7$:     MOV     #SPLCHN,R0      ;SAVE CHANNEL STATUS HERE
    108 011166  013702  000000G                 MOV     SPLBLK,R2       ;GET DEVICE NAME
    109 011172  004737  024410'                 CALL    SETCHN          ;SAVE CHANNEL STATUS
    110 011176                                  .RELEAS #SPLBLK         ;Release the RT-11 device handler
    111                                 ;
    112                                 ;  Set number of free public blocks in spool file
    113                                 ;
    114 011206  113703  000000G                 MOVB    NSPLDV,R3       ;Get # spooled devices
```

```
    115 011212  070327  000000G                 MUL     #PVSPBL,R3      ;Times number of private blocks per dev
    116 011216  005403                          NEG     R3
    117 011220  063703  000000G                 ADD     NSPLBL,R3               ;Get # public spool blocks
    118 011224  010337  000000G                 MOV     R3,NFRESB       ;This is number of public free spool blocks
    119                             ;
    120                             ;  Finished
    121                             ;
    122 011230  012605              12$:        MOV     (SP)+,R5
    123 011232  012604                          MOV     (SP)+,R4
    124 011234  012603                          MOV     (SP)+,R3
    125 011236  012602                          MOV     (SP)+,R2
    126 011240  012601                          MOV     (SP)+,R1
    127 011242  000207              9$:         RETURN
    128                             ;
    129                             ;  Error: Cannot open spool file.
    130                             ;
    131 011244                      8$:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
    132 011252                                  .PRINT  #BOSF
    133 011260  000137  004134'                 JMP     INISTP          ;ABORT INITIALIZATION
    134                             ;
    135                             ;  Error: Invalid device specification.
    136                             ;
    137 011264  010001              11$:        MOV     R0,R1           ;Save device name
    138 011266                                  .PRINT  #TSXHD          ;Print error message
    139 011274                                  .PRINT  #BOSF
    140 011302  004737  010170'                 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
                                ;
011306  010546                  SPLCLD: MOV     R5,-(SP)
                                ;
                                ;  Make sure CL unit number is valie
                                ;
011310  010005                          MOV     R0,R5           ;Get CL unit number
011312  020527  000000G                 CMP     R5,#CLTOTL      ;Is this a valid unit #
011316  103022                          BHIS    8$              ;Br if invalid
                                ;
                                ;  Set up channel control block in SDCB
                                ;
011320  005061  000000C                 CLR     SDCHAN+C.SBLK(R1);Starting block # = 0
011324  020527  000007                  CMP     R5,#7           ;Is this a CL or C1 unit?
011330  101405                          BLOS    1$              ;Br if CL unit
011332  162705  000010                  SUB     #8.,R5          ;Remove C1 unit # bias
011336  013700  000000G                 MOV     C1DEVX,R0       ;Get C1 device index number
011342  000402                          BR      2$
011344  013700  000000G         1$:     MOV     CLDEVX,R0       ;Get CL device index number
011350  010061  000000C         2$:     MOV     R0,SDCHAN+C.CSW(R1);Set device index number
011354  110561  000000C                 MOVB    R5,SDCHAN+C.DEVQ(R1);Set unit #
                                ;
                                ;  We successfully set up a CL unit
                                ;
011360  000241                          CLC                     ;Signal success on error
011362  000401                          BR      9$
                                ;
                                ;  We cannot open this CL unit
                                ;
011364  000261                  8$:     SEC                     ;Signal error
                                ;
                                ;  Finished
                                ;
011366  012605                  9$:     MOV     (SP)+,R5
011370  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)
                                ;
011372                          CHKCLD:
                                ;
                                ;  See if this is a CL unit
                                ;
011372  020027  012240                  CMP     R0,#R50CL       ;Is name "CL"?
011376  001411                          BEQ     1$              ;Br if yes
011400  020027  012276                  CMP     R0,#R50CL0      ;Is name in the range CL0 to CL7?
011404  103432                          BLO     8$              ;Br if not
011406  020027  012305                  CMP     R0,#R50CL7
011412  101005                          BHI     2$
011414  162700  012276                  SUB     #R50CL0,R0      ;Get CL unit number
011420  000422                          BR      7$
011422  005000                  1$:     CLR     R0              ;CL = CL0
011424  000420                          BR      7$
                                ;
                                ;  See if this is a C1 unit
                                ;
011426  020027  013630          2$:     CMP     R0,#R50C1       ;Is name "C1"?
011432  001413                          BEQ     3$              ;Br if yes
011434  020027  013666                  CMP     R0,#R50C10      ;Is name in the range C10 to C17?
011440  103414                          BLO     8$              ;Br if not
011442  020027  013675                  CMP     R0,#R50C17
011446  101011                          BHI     8$
011450  162700  013666                  SUB     #R50C10,R0      ;Get unit number
011454  062700  000010                  ADD     #8.,R0          ;Bias by 8 for C1 units
011460  000402                          BR      7$
011462  012700  000010          3$:     MOV     #8.,R0          ;C1 = CL8
                                ;
                                ;  This is a CL or C1 unit
                                ;
011466  000241                  7$:     CLC                     ;Signal success on return
011470  000401                          BR      9$
                                ;
                                ;  This is not a CL or C1 unit
                                ;
011472  000261                  8$:     SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
011474  000207                  9$:     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).
                                ;
011476  010246                  CVTDVU: MOV     R2,-(SP)
011500  010346                          MOV     R3,-(SP)
                                ;
                                ;  Split the unit number off of the full device name
                                ;
011502  010003                          MOV     R0,R3           ;Get full device name
011504  005002                          CLR     R2              ;Set up for divide
011506  071227  000050                   DIV     #50,R2          ;Split name and unit (R0=name, R1=unit)
011512  005703                          TST     R3              ;Was a unit number specified?
011514  001402                          BEQ     1$              ;Br if not
011516  162703  000036                   SUB     #36,R3          ;Convert unit number to binary value
011522  010300                  1$:     MOV     R3,R0           ;Get unit number
011524  000300                          SWAB    R0              ;Position to high-order byte
                                ;
                                ;  Look up the device name to get the device index
                                ;
011526  070227  000050                  MUL     #50,R2          ;Now get the device name without unit number
011532  013702  0000000G                MOV     NUMDEV,R2       ;Get index number of last device
011536  020362  0000000G          2$:   CMP     R3,PNAME(R2)    ;Search for device in name table
011542  001407                          BEQ     3$              ;Br if found it
011544  162702  000002                  SUB     #2,R2           ;Try next device
011550  002372                          BGE     2$              ;Loop if more to check
                                ;
                                ;  Error, cannot find device name in tables
                                ;
011552  012700  177777                  MOV     #-1,R0          ;Set device # = unit # = -1
011556  000261                          SEC                     ;Signal error on return
011560  000402                          BR      9$
                                ;
                                ;  Found the device in the tables
                                ;
011562  050200                  3$:     BIS     R2,R0           ;Combine device # and unit #
011564  000241                          CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
011566  012603                  9$:     MOV     (SP)+,R3
011570  012602                          MOV     (SP)+,R2
011572  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
                                ;
011574  010346                  FORCE0: MOV     R3,-(SP)
011576  010446                          MOV     R4,-(SP)
011600  010546                          MOV     R5,-(SP)
011602  011305                          MOV     (R3),R5         ;MOVE CURRENT DEV NAME TO R5
011604  005004                          CLR     R4              ;SET UP FOR DIVIDE
011606  071427  000050                  DIV     #50,R4          ;SEPARATE INTO NAME AND UNIT
011612  005705                          TST     R5              ;WAS 3RD CHAR BLANK?
011614  001012                          BNE     9$              ;RETURN IF NOT
011616  010405                          MOV     R4,R5           ;GET HIGH 2 CHARS
011620  005004                          CLR     R4              ;SET UP FOR ANOTHER DIVIDE
011622  071427  000050                  DIV     #50,R4          ;SEPARATE 1 & 2 CHARS
011626  005704                          TST     R4              ;WAS CHAR 1 BLANK?
011630  001404                          BEQ     9$              ;EMPTY OR INVALID DEV NAME!
011632  005705                          TST     R5              ;WAS CHAR 2 BLANK?
011634  001402                          BEQ     9$              ;1-CHAR DEV NAME SHOULD BE INVALID???
011636  062713  000036                  ADD     #^R  0,(R3)     ;FORCE TO UNIT 0
011642  012605                  9$:     MOV     (SP)+,R5
011644  012604                          MOV     (SP)+,R4
011646  012603                          MOV     (SP)+,R3
011650  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.
                ;
011652  010146  ALOCBF: MOV     R1,-(SP)
                ;
                ;  Assign space for I/O queue elements.
                ;
011654  010537  000000G         MOV     R5,FREIOQ       ;START OF I/O QUEUE SPACE
011660  062705  000000C         ADD     #IOQSIZ*NUMIOQ,R5;RESERVE SPACE FOR I/O QUEUE ELEMENTS
                ;
                ;  Assign space for shared PLAS region control blocks
                ;
011664  010537  000000G         MOV     R5,SHRRCB       ;Start of area for RCB's
011670  013701  000000G         MOV     VNGR,R1         ;Get number of RCB's wanted
011674  020137  000000G         CMP     R1,VMXWIN       ;Must have one for each display window
011700  103002                  BHIS    13$             ;Br if ok
011702  013701  000000G         MOV     VMXWIN,R1       ;Force one for each window
011706  070127  000000G 13$:    MUL     #RC$$SZ,R1      ;Multiply by size of each block
011712  060105                  ADD     R1,R5           ;Allocate space for RCB's
011714  010537  000000G         MOV     R5,SHRRCN       ;Address of end of region
                ;
                ;  Assign space for fork blocks
                ;
011720  012700  000000C         MOV     #<NUMFRK-FRKGEN>,R0 ;Get # fork blocks to allocate
011724  003404                  BLE     11$             ;Br if none to allocate
011726  010537  000000G         MOV     R5,FRKINI       ;Set pointer to start of area
011732  062705  000000C         ADD     #<<NUMFRK-FRKGEN>*FQ$$SZ>,R5 ;Reserve space for fork blocks
                ;
                ;  Assign space for job monitoring control blocks
                ;
011736  013701  000000G 11$:    MOV     VMXMON,R1       ;Any job monitoring blocks wanted?
011742  001405                  BEQ     10$             ;Br if not
011744  010537  000000G         MOV     R5,MONFQH       ;Start of job monitoring control blocks
011750  070127  000000G         MUL     #JM$$SZ,R1      ;Compute space needed for control blocks
011754  060105                  ADD     R1,R5           ;Allocate the space
                ;
                ;  Allocate space for system message buffers.
                ;
011756  010537  000000G 10$:    MOV     R5,SNMSHD       ;HEAD OF SYSTEM MESSAGE BUFFER AREA
011762  062705  000000C         ADD     #<NMSNMB*SB$$SZ>,R5;RESERVE ROOM FOR MESSAGE BUFFERS
                ;
                ;  Allocate space for INSTALLed program table
                ;
011766  010537  000000G         MOV     R5,INSTBL       ;Base of table
011772  013701  000000G         MOV     VNUIP,R1        ;# slots for user installed programs
011776  062701  000000G         ADD     #NSIP,R1        ;Add # slots for system programs
012002  070127  000000G         MUL     #II$$SZ,R1      ;Multiply by size of each slot
```

```
                        060105                          ADD     R1,R5           ;Allocate space for table
                        010537  000000G                 MOV     R5,INSTBN       ;Pointer past end of INSTALL table
                                                ;
                                                ;  Allocate space for device mount entries
                                                ;
                        010537  000000G                 MOV     R5,CSHDEV       ;Point to start of area
                        012701  000000G                 MOV     #CD$$SZ,R1      ;Get size of each entry
                        070137  000000G                 MUL     VMXCSH,R1       ;Multiply by number of entries
                        060105                          ADD     R1,R5           ;Reserve space for table
                        010537  000000G                 MOV     R5,CSHDVN       ;Save pointer past end of table
                                                ;
                                                ;  Allocate space for data cache control blocks
                                                ;
                        005737  000000G                 TST     CSHALC          ;Is data caching wanted?
                        001404                          BEQ     12$             ;Br if not
                        010537  000000G                 MOV     R5,CCBHD        ;Head of free list area
                        062705  000000C                 ADD     #NUMCCB*CC$$SZ,R5 ;Allocate space for control blocks
                                                ;
                                                ;  Allocate space for spool file control blocks
                                                ;
                        013701  000000G         12$:    MOV     NSPLFL,R1       ;Get # spool file control blocks needed
                        001407                          BEQ     1$              ;Br if none needed
                        070127  000000G                 MUL     #SFCBSZ,R1      ;Compute space needed by control blocks
                        010537  000000G                 MOV     R5,SFCB         ;Base of control block area
                        060105                          ADD     R1,R5           ;Allocate space for control blocks
                        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.
                                                ;
                        012701  000002          1$:     MOV     #2,R1           ;START WITH FIRST MUX
                        020127  000000G         2$:     CMP     R1,#LSTMX       ;HAVE WE DONE ALL MUX'S?
                        101007                          BHI     5$              ;BR IF YES
                        010561  000000G                 MOV     R5,MXLNT(R1)    ;SET ADDRESS OF START OF VECTOR
                        062705  000020                  ADD     #16.,R5         ;RESERVE SPACE FOR VECTOR
                        062701  000002                  ADD     #2,R1           ;ADVANCE TO NEXT MUX
                        000766                          BR      2$
                                                ;
                                                ;  Allocate buffers to hold characters for DMA transfers to DH11 multiplexers
                                                ;
                        012701  000002          5$:     MOV     #2,R1           ;Start with first line
                        020127  000000G         6$:     CMP     R1,#LSTPL       ;Is this a primary time-sharing line?
                        101403                          BLOS    3$              ;Br if yes
                        020127  000000G                 CMP     R1,#FSTIOL      ;Is this a CL line?
                        103422                          BLO     7$              ;Br if not
                        026127  000000G 000000G 3$:     CMP     LCDTYP(R1),#CDX$DH ;Is this line connected to a DH11?
                        001404                          BEQ     8$              ;Br if yes
                        026127  000000G 000000G         CMP     LCDTYP(R1),#CDX$VH ;Is this line connected to a DHV11?
                        001012                          BNE     7$              ;Br if not
                        010561  000000G         8$:     MOV     R5,LDHB1B(R1)   ;Set address of start of buffer 1
                        010561  000000G                 MOV     R5,LDHB1P(R1)   ;Initialize pointer into buffer 1
                        062705  000000G                 ADD     #DH8FSZ,R5      ;Reserve space for buffer
                        010561  000000G                 MOV     R5,LDHB2B(R1)   ;Set address of start of buffer 2
                        062705  000000G                 ADD     #DH8FSZ,R5      ;Reserve space for buffer
                        062701  000002          7$:     ADD     #2,R1           ;Get # of next line
                        020127  000000G                 CMP     R1,#LSTHL       ;Have we checked all lines?
```

```
    115 012224  101743                          BLOS    6$              ;Br if not
    116 012226  005205                          INC     R5              ;Bound up to next word
    117 012230  042705  000001                  BIC     #1,R5
    118                                 ;
    119                                 ;  Allocate space for tables that keep track of space in swap file
    120                                 ;
    121 012234  105737  000000G                 TSTB    VSWPFL          ;Is this a swapping system?
    122 012240  001415                          BEQ     14$             ;Br if not
    123 012242  013700  000000G                 MOV     VSWPSL,R0       ;Get # slots in swap file
    124 012246  006300                          ASL     R0              ;Allocate 2 bytes per slot
    125 012250  010537  000000G                 MOV     R5,SWPPOS       ;Start of table with starting block #'s
    126 012254  060005                          ADD     R0,R5           ;Allocate space
    127 012256  010537  000000G                 MOV     R5,SWPJOB       ;Start of table with job #'s
    128 012262  060005                          ADD     R0,R5           ;Allocate space
    129 012264  010537  000000G                 MOV     R5,SCPFHD       ;Pointer to area with command packets
    130 012270  062705  000000C                 ADD     #NSCP*SP$$SZ,R5 ;Allocate space for swap command packets
    131                                 ;
    132                                 ;  Allocate a 512-byte buffer to use to access job context blocks
    133                                 ;
    134 012274  010537  000000G         14$:    MOV     R5,CXTBUF       ;Set address of buffer
    135 012300  062705  001400                  ADD     #1400,R5        ;Reserve space for the buffer
    136                                 ;
    137                                 ;  Make sure TSX is not too big.
    138                                 ;
    139 012304  010500                          MOV     R5,R0           ;GET CURRENT MEMORY ADDRESS
    140 012306  004737  025104'                 CALL    CHKMEM          ;CHECK FOR SPACE OVERFLOW
    141                                 ;
    142                                 ;  Finished
    143                                 ;
    144 012312  012601                          MOV     (SP)+,R1
    145 012314  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.
                                ;
012316  010146                  ALCSLO: MOV     R1,-(SP)
012320  010246                          MOV     R2,-(SP)
                                ;
                                ;  Begin loop to check each line
                                ;
012322  012701  000000G                 MOV     #LSTHL,R1       ;Get index to last hardware line
                                ;
                                ;  Only allocate silo buffers for real lines
                                ;
012326  012702  000040          1$:     MOV     #32.,R2         ;Set minimum size for time-sharing lines
012332  020127  000000G                 CMP     R1,#LSTPL       ;Is this a primary line?
012336  101405                          BLOS    2$              ;Br if yes
012340  020127  000000G                 CMP     R1,#FSTIOL      ;Is this a CL line?
012344  103463                          BLO     9$              ;Br if not
012346  012702  000020                  MOV     #16.,R2         ;Set minimum size for CL lines
                                ;
                                ;  Determine how much space to allocate
                                ;
012352  016100  000000G         2$:     MOV     LHIRBA(R1),R0   ;Get requested size
012356  001002                          BNE     8$              ;Br if a size was specified
012360  013700  000000G                 MOV     VNCSLO,R0       ;Use default value
012364  020027  000000G         8$:     CMP     R0,#MAXSLO      ;Constrain size to 255 bytes
012370  101402                          BLOS    3$              ;Br if ok
012372  012700  000000G                 MOV     #MAXSLO,R0      ;Reduce size
012376  020002                  3$:     CMP     R0,R2           ;Compare with acceptable minimum
012400  103001                          BHIS    4$              ;Br if ok
012402  010200                          MOV     R2,R0           ;Use minimum size allowed
012404  010061  000000G         4$:     MOV     R0,LHIRBA(R1)   ;Set # bytes to be allocated for silo
                                ;
                                ;  Allocate the space
                                ;
012410  010561  000000G                 MOV     R5,LHIRBB(R1)   ;Set base address of buffer
012414  010561  000000G                 MOV     R5,LHIRBP(R1)   ;Init pointer where to store next char
012420  010561  000000G                 MOV     R5,LHIRBG(R1)   ;Init pointer where to get next char
012424  010061  000000G                 MOV     R0,LHIRBS(R1)   ;Set # free bytes in buffer
012430  060005                          ADD     R0,R5           ;Allocate space for buffer
012432  010561  000000G                 MOV     R5,LHIRBE(R1)   ;Set address beyond end of buffer
                                ;
                                ;  Set up control information about when to send XON and XOFF
                                ;
012436  006200                          ASR     R0              ;Get 1/2 of buffer size
012440  162700  000002                  SUB     #2,R0           ;Minus two characters
012444  116102  000000G                 MOVB    LHIRBC(R1),R2   ;Get specified size for XOFF point
012450  001002                          BNE     5$              ;Br if value specified
012452  113702  000000G                 MOVB    VNCXOF,R2       ;Try default
012456  020200                  5$:     CMP     R2,R0           ;Is specified value ok?
```

```
     58 012460  101401                          BLOS    6$                ;Br if yes
     59 012462  010002                          MOV     R0,R2             ;No, use size/2-2
     60 012464  110261  000000G       6$:      MOVB    R2,LHIRBC(R1)     ;Set # of chars when XOFF sent
     61 012470  116102  000001G                 MOVB    LHIRBC+1(R1),R2 ;Get specified size for XON point
     62 012474  001002                          BNE     7$                ;Br if a value was specified
     63 012476  113702  000000G                 MOVB    VNCXON,R2         ;Try default
     64 012502  020200                  7$:     CMP     R2,R0             ;Is specified value ok?
     65 012504  101401                          BLOS    10$               ;Br if ok
     66 012506  010002                          MOV     R0,R2             ;No, use size/2-2
     67 012510  110261  000001G       10$:     MOVB    R2,LHIRBC+1(R1) ;Set # of chars when XON sent
     68                                 ;
     69                                 ;  Do the next line
     70                                 ;
     71 012514  162701  000002          9$:     SUB     #2,R1             ;Get next line index number
     72 012520  003302                          BGT     1$                ;Loop if more to do
     73                                 ;
     74                                 ;  Finished
     75                                 ;
     76 012522  005205                          INC     R5                ;Force R5 to be even
     77 012524  042705  000001                  BIC     #1,R5
     78 012530  012602                          MOV     (SP)+,R2
     79 012532  012601                          MOV     (SP)+,R1
     80 012534  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.
                                ;
012536  010146                  ALBFX:  MOV     R1,-(SP)
012540  010246                          MOV     R2,-(SP)
012542  010346                          MOV     R3,-(SP)
                                ;
                                ;  Allocate character buffers for all lines
                                ;  Note: Character buffer space will be accessed by mapping through PAR 6.
                                ;
012544  012701  000002                  MOV     #2,R1           ;GET 1ST JOB INDEX NUMBER
012550  032761  000000G 000000G 3$:     BIT     #$DEAD,LSW3(R1) ;IS THIS LINE INSTALLED?
012556  001047                          BNE     2$              ;BR IF NOT -- DON'T ALLOCATE ANY BUFFER SPACE
012560  020127  000000G                 CMP     R1,#FSTDL       ;IS THIS A DETACHED JOB LINE?
012564  103403                          BLO     1$              ;BR IF NOT
012566  020127  000000G                 CMP     R1,#LSTDL       ;DETACHED JOB LINE?
012572  101441                          BLOS    2$              ;BR IF DETACHED JOB -- DON'T ALLOCATE  BUFFERS
012574  010561  000000G         1$:     MOV     R5,LTTPAR(R1)   ;SET PHYSICAL MEMORY PAR OFFSET FOR BUFFER
012600  012702  000000G                 MOV     #VPAR6,R2       ;GET VIRTUAL MEMORY ADDRESS FOR BASE OF PAR6
012604  010261  000000G                 MOV     R2,LINBUF(R1)   ;INPUT BUFFER STARTS AT BASE OF PAR6 REGION
012610  016100  000000G                 MOV     LINSIZ(R1),R0   ;GET # BYTES FOR INPUT BUFFER
012614  010061  000000G                 MOV     R0,LINSPC(R1)   ;SET # FREE BYTES IN INPUT BUFFER
012620  060002                          ADD     R0,R2           ;ADVANCE VIRTUAL ADDRESS
012622  010261  000000G                 MOV     R2,LINEND(R1)   ;POINTS PAST END OF INPUT BUFFER
012626  010003                          MOV     R0,R3           ;Get # bytes in input buffer
012630  062703  000007                  ADD     #7,R3           ;Bound up to multiple of 8
012634  072327  177775                  ASH     #-3,R3          ;Get # bytes needed in activation-flag buffer
012640  060302                          ADD     R3,R2           ;Reserve space for activation-flag buffer
012642  060300                          ADD     R3,R0           ;Accumulate total buffer space
012644  010261  000000G                 MOV     R2,LOTBUF(R1)   ;POINTS TO START OF OUTPUT BUFFER AREA
012650  066100  000000G                 ADD     LOTSIZ(R1),R0   ;ACCUMULATE # BYTES IN BOTH BUFFERS
012654  066102  000000G                 ADD     LOTSIZ(R1),R2   ;ADVANCE VIRTUAL ADDRESS COUNTER
012660  010261  000000G                 MOV     R2,LOTEND(R1)   ;SAVE ADDRESS OF END OF OUTPUT BUFFER
012664  062700  000077                  ADD     #77,R0          ;BOUND UP TO MULTIPLE OF 64 BYTES
012670  072027  177772                  ASH     #-6,R0          ;CONVERT TO # 64-BYTE BLOCKS ALLOCATED
012674  060005                          ADD     R0,R5           ;ADVANCE PHYSICAL MEMORY PAR ADDRESS
012676  062701  000002          2$:     ADD     #2,R1           ;ADVANCE LINE NUMBER
012702  020127  000000G                 CMP     R1,#LSTSL       ;HAVE WE DONE ALL LINES YET?
012706  101720                          BLOS    3$              ;BR IF MORE TO DO
                                ;
                                ;  Allocate space for shared file record locking data structures
                                ;
012710  005737  000000G                 TST     VMXSF           ;Shared file support wanted?
012714  001451                          BEQ     5$              ;Br if not
012716  013737  000000G 000000G         MOV     VNUMDC,NUMDCD   ;Set number of data cache blocks
012724  013737  000000G 000000G         MOV     VMXSFC,NUMCDB   ;Set number of free CDB's
012732  010537  000000G                 MOV     R5,LOKMEM       ;Set phys address of base of area
012736  012701  000000G                 MOV     #FF$$SZ,R1      ;Size of an FDB
012742  070137  000000G                 MUL     VMXSF,R1        ;Times number of FDB's
```

```
     58 012746  062701  000000C                ADD     #<NLINES*FW$$SZ>,R1 ;Space needed for wait blocks
     59 012752  013703  000000G                MOV     VMLBLK,R3           ;Max blocks a CDB may hold locked
     60 012756  006303                         ASL     R3                  ;Two bytes per entry
     61 012760  062703  000000G                ADD     #FC$LBN,R3          ;Add base size of a CDB
     62 012764  070337  000000G                MUL     VMXSFC,R3           ;Times number of shared file channels
     63 012770  060301                         ADD     R3,R1               ;Accumulate space needed
     64 012772  012703  000000G                MOV     #DC$$SZ,R3          ;Size of a data cache descriptor
     65 012776  070337  000000G                MUL     VNUMDC,R3           ;Times number of data cache entries
     66 013002  060301                         ADD     R3,R1               ;Reserve space for data cache descriptors
     67 013004  062701  000100                 ADD     #64.,R1             ;Bound up to 64 byte unit
     68 013010  072127  177772                 ASH     #-6,R1              ;Convert to # 64 byte units
     69 013014  042701  176000                 BIC     #176000,R1          ;Clear sign extension
     70 013020  060105                         ADD     R1,R5               ;Reserve space for data structures
     71 013022  010537  000000G                MOV     R5,LOKCSH           ;Save pointer to start of cache buffer area
     72 013026  013701  000000G                MOV     VNUMDC,R1           ;# shared-file data cache blocks wanted
     73 013032  070127  000010                 MUL     #8.,R1              ;8 64-byte blocks each (512 bytes each)
     74 013036  060105                         ADD     R1,R5               ;Reserve space for data cache buffers
     75                                ;
     76                                ;  Allocate space for mapped I/O buffers
     77                                ;
     78 013040  105737  000000G        5$:     TSTB    MIOFLG              ;Are any mapped I/O buffers needed?
     79 013044  001415                         BEQ     7$                  ;Br if not
     80 013046  013701  000000G                MOV     MIOBHD,R1           ;Point to 1st mapped I/O control block
     81 013052  001412                         BEQ     7$                  ;Br if no more buffers needed
     82 013054  010561  000000G        6$:     MOV     R5,MI$SBP(R1)       ;Set address of base of buffer
     83 013060  113703  000000G                MOVB    VMIOSZ,R3           ;Get # blocks for buffer
     84 013064  072327  000003                 ASH     #3,R3               ;Convert to # 64-byte pages
     85 013070  060305                         ADD     R3,R5               ;Allocate space for buffer
     86 013072  016101  000000G                MOV     MI$LNK(R1),R1       ;Get address of next control block
     87 013076  001366                         BNE     6$                  ;Loop if more to allocate
     88                                ;
     89                                ;  Allocate space for performance monitor data buffer if it is wanted.
     90                                ;
     91 013100  013701  000000G        7$:     MOV     VPMSIZ,R1           ;DID USER GEN IN PERFORMANCE MONITOR FEATURE?
     92 013104  001412                         BEQ     9$                  ;BR IF NOT
     93 013106  006201                         ASR     R1                  ;CONVERT BYTES TO WORDS
     94 013110  010137  000000G                MOV     R1,PMCELS           ;SET INTO CELL IN TSEXEC
     95 013114  010537  000000G                MOV     R5,PMPAR            ;SET BASE ADDRESS OF PM BUFFER
     96 013120  062701  000037                 ADD     #37,R1              ;BOUND SIZE UP TO 64-BYTE MULTIPLE
     97 013124  072127  177773                 ASH     #-5,R1              ;CONVERT WORDS TO # 64-BYTE BLOCKS
     98 013130  060105                         ADD     R1,R5               ;ADVANCE FREE MEMORY POINTER
     99                                ;
    100                                ;  Finished
    101                                ;
    102 013132  012603                 9$:     MOV     (SP)+,R3
    103 013134  012602                         MOV     (SP)+,R2
    104 013136  012601                         MOV     (SP)+,R1
    105 013140  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.
                                ;
013142  010246                  OPNKMN: MOV     R2,-(SP)
                                ;
                                ;  Lookup TSKMON file.
                                ;
013144                                  .LOOKUP #AREA,#1,#KMNNAM ;TRY TO FILE KMON SAV FILE
013164  103517                          BCS     9$              ;BR IF NOT THERE
                                ;
                                ;  Read block 0 of save file and extract some information.
                                ;
013166  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
013172                                  .READW  #AREA,#1,R2,#256.,#0 ;READ BLOCK 0 OF SAV FILE
                                ;  Determine size of kmon
013226  016200  000050                  MOV     50(R2),R0       ;GET TOP ADDRESS OF KMON
013232  062700  000003                  ADD     #3,R0           ;BOUND UP TO NEXT WORD
013236  042700  000001                  BIC     #1,R0           ;FORCE EVEN
013242  010037  000000G                 MOV     R0,KMNTOP
                                ;  Determine number of 256-word memory pages needed while kmon running.
013246  162700  000000G                 SUB     #KMNBAS,R0      ;BASE ADDRESS OF KMON
013252  010037  000000G                 MOV     R0,KMNHI        ;TOP OF TSKMON - KMNBAS
013256  062700  000777                  ADD     #511.,R0        ;BOUND UP TO PAGE SIZE
013262  000241                          CLC
013264  006000                          ROR     R0              ;CVT TO # WORDS
013266  000300                          SWAB    R0              ;CVT TO # PAGES
013270  042700  177400                  BIC     #^C377,R0
013274  063700  000000G                 ADD     CXTPAG,R0       ;# PAGES NEEDED FOR JOB CONTEXT AREA
013300  010037  000000G                 MOV     R0,KMNPGS       ;# 256-wd pages needed for kmon + context area
                                ;  Determine Kmon stack pointer.
013304  016237  000042  000000G         MOV     42(R2),KMNSTK   ;INITIAL STACK POINTER FOR KMON
                                ;  Determine Kmon starting address.
013312  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)
                                ;
013320  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.
                                ;
013326  012700  000000G                 MOV     #KMNCHN,R0      ;GET KMON CHANNEL SAVE AREA
013332  013702  000072'                 MOV     KMNNAM,R2       ;GET KMON DEVICE NAME
```

```
     58 013336  004737  024410'                 CALL    SETCHN          ;SAVE CHANNEL STATUS
     59                                 ;
     60                                 ;  Lookup CCL.SAV and save channel status for it.
     61                                 ;
     62 013342                                  .LOOKUP #AREA,#1,#CCLNAM;LOOKUP SY:CCL.SAV
     63 013362  103410                          BCS     8$              ;BR IF CAN'T FIND CCL
     64 013364  012700  000000G                 MOV     #CCLSAV,R0      ;CHANNEL SAVE AREA
     65 013370  013702  000102'                 MOV     CCLNAM,R2       ;DEVICE NAME
     66 013374  004737  024410'                 CALL    SETCHN          ;SAVE CHANNEL STATUS
     67                                 ;
     68                                 ;  Finished
     69                                 ;
     70 013400  012602                  10$:    MOV     (SP)+,R2
     71 013402  000207                          RETURN
     72                                 ;
     73                                 ;  Error:  We could not find SY:CCL.SAV
     74                                 ;
     75 013404                          8$:     .PRINT  #TSXHD
     76 013412                                  .PRINT  #NOCCL          ;PRINT ERROR MESSAGE
     77 013420  000137  004134'                 JMP     INISTP          ;ABORT INITIALIZATION
     78                                 ;
     79                                 ;  Error: We could not locate TSKMON SAV file.
     80                                 ;
     81 013424                          9$:     .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     82 013432                                  .PRINT  #NOKMON
     83 013440  000137  004134'                 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.
                                ;
013444  010146                  CLINIT: MOV     R1,-(SP)
013446  010246                          MOV     R2,-(SP)
013450  010346                          MOV     R3,-(SP)
                                ;
                                ;  Initialize tables for each CL unit
                                ;
013452  005003                          CLR     R3              ;Accumulate ring buffer sizes in R3
013454  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.
                                ;
013460  016102  000000G         1$:     MOV     CL$LIX(R1),R2   ;Is this CL unit associated with a line?
013464  001416                          BEQ     5$              ;Br if not
013466  012762  000000G 000000G        MOV     #$SXON,LSW10(R2);Send XON when we start the line
013474  010162  000000G                 MOV     R1,LCLUNT(R2)   ;Associate the CL unit with this line
013500  005762  000000G                 TST     RSR(R2)         ;Does this unit have a specified RSR addr?
013504  001006                          BNE     5$              ;Br if yes
013506  005762  000000G                 TST     LMXNUM(R2)      ;Is this a mux line?
013512  001003                          BNE     5$              ;Br if yes
013514  005061  000000G                 CLR     CL$EPS(R1)      ;Say no endstring buffer
013520  000472                          BR      4$              ;Line is not genned in
                                ;
                                ;  Allocate and set up pointers for the output ring buffers
                                ;
013522  010561  000000G         5$:     MOV     R5,CL$ORB(R1)   ;Start of output ring buffer
013526  010561  000000G                 MOV     R5,CL$ORP(R1)   ;Input character pointer
013532  010561  000000G                 MOV     R5,CL$ORG(R1)   ;Next available character pointer
013536  012700  000000G                 MOV     #CLORSZ,R0      ;Get default output ring buffer size
013542  005702                          TST     R2              ;Is this CL unit connected to a line?
013544  001402                          BEQ     6$              ;Br if not
013546  016200  000000G                 MOV     LOTSIZ(R2),R0   ;Get size of output ring buffer
013552  010061  000000G         6$:     MOV     R0,CL$ORA(R1)   ;Set size of output ring buffer
013556  010061  000000G                 MOV     R0,CL$ORS(R1)   ;Available space in ring buffer
013562  060005                          ADD     R0,R5           ;Point beyond end of ring buffer
013564  010561  000000G                 MOV     R5,CL$ORE(R1)   ;Address past end of ring buffer
013570  060003                          ADD     R0,R3           ;Accumulate size of output ring buffers
                                ;
                                ;  Allocate space for end-of-file string buffer
                                ;
013572  010561  000000G                 MOV     R5,CL$EPS(R1)   ;Set pointer to end-of-file string buffer
013576  005061  000000G                 CLR     CL$EPP(R1)      ;No string to print yet
013602  062705  000001G                 ADD     #<CLEOFS+1>,R5  ;Reserve space for buffer
013606  062703  000001G                 ADD     #<CLEOFS+1>,R3  ;Accumulate buffer space
                                ;
                                ;  Initialize end-of-file form-feed count
                                ;
```

```
     58 013612  005061  000000G                    CLR     CL$EPN(R1)       ;Init ENDPAGE=0
     59                                ;
     60                                ;  Initialize option word
     61                                ;
     62 013616  012700  000000G                    MOV     #<CO$DEF>,R0     ;Get default option flags
     63 013622  005702                             TST     R2               ;Is this CL unit connected with a line?
     64 013624  001421                             BEQ     7$               ;Br if not
     65 013626  016202  000000G                    MOV     ILSW2(R2),R2     ;Get line options
     66 013632  032702  000000G                    BIT     #$TAB,R2         ;Does hardware support tabs?
     67 013636  001402                             BEQ     2$               ;Br if not
     68 013640  052700  000000G                    BIS     #CO$TAB,R0       ;Set hardware-tab flag
     69 013644  032702  000000G          2$:       BIT     #$FORM,R2        ;Does hardware support form feeds?
     70 013650  001402                             BEQ     3$               ;Br if not
     71 013652  052700  000000G                    BIS     #CO$FF,R0        ;Set hardware-form-feed flag
     72 013656  032702  000000G          3$:       BIT     #$8BIT,R2        ;Does hardware want 8 bit support?
     73 013662  001402                             BEQ     7$               ;Br if not
     74 013664  052700  000000G                    BIS     #CO$8BT,R0       ;Enable 8 bit support for CL line
     75 013670  010061  000000G          7$:       MOV     R0,CL$OPT(R1)    ;Set options for this CL line
     76                                ;
     77                                ;  Initialize page length
     78                                ;
     79 013674  012761  000102  000000G            MOV     #66.,CL$LEN(R1) ;Say page length = 66 lines
     80                                ;
     81                                ;  Initialize status flags
     82                                ;
     83 013702  005061  000000G                    CLR     CL$STA(R1)       ;Initialize status flags
     84                                ;
     85                                ;  Do next line
     86                                ;
     87 013706  162701  000002          4$:       SUB     #2,R1            ;Get index # of next unit
     88 013712  002262                             BGE     1$               ;Loop if more units to initialize
     89                                ;
     90                                ;  Make a device table entry for "CL" device
     91                                ;
     92 013714  062737  000002  000000G            ADD     #2,NUMDEV        ;One more device
     93 013722  013701  000000G                    MOV     NUMDEV,R1        ;Get device table index
     94 013726  010137  000000G                    MOV     R1,CLDEVX        ;Remember index number of CL device
     95 013732  012761  012240  000000G            MOV     #R50CL,PNAME(R1) ;Set device name ("CL")
     96 013740  012761  000000G 000000G            MOV     #CLSTS,DVSTAT(R1) ;Set dev status flags
     97 013746  012761  000000C 000000G            MOV     #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ;Device info flags
     98 013754  005061  000000G                    CLR     DEVSIZ(R1)       ;Clear device size
     99 013760  012761  000006G 000000G            MOV     #CLHEAD+6,HANENT(R1) ;Set handler entry point (4th word)
    100 013766  062703  000000G                    ADD     #CLSIZE,R3       ;Get size of handler
    101 013772  062703  000000C                    ADD     #<<CLTOTL*46.>+<NIOL*32.>>,R3 ;Add size of tables in TSGEN
    102 013776  010361  000000G                    MOV     R3,HANSIZ(R1)    ;Set size of handler
    103 014002  005205                             INC     R5               ;Make sure free-memory pointer is even
    104 014004  042705  000001                     BIC     #1,R5
    105                               ;
    106                               ;  Make a device table entry for C1 if there are more than 8 CL units
    107                               ;
    108 014010  022727  000000G 000010             CMP     #CLTOTL,#8.      ;Are there more than 8 CL units?
    109 014016  101430                             BLOS    9$               ;Br if not -- Don't need C1
    110 014020  062737  000002  000000G            ADD     #2,NUMDEV        ;One more device
    111 014026  013701  000000G                    MOV     NUMDEV,R1        ;Get device table index
    112 014032  010137  000000G                    MOV     R1,C1DEVX        ;Remember index number of CL device
    113 014036  012761  013630  000000G            MOV     #R50C1,PNAME(R1) ;Set device name ("C1")
    114 014044  012761  000000G 000000G            MOV     #CLSTS,DVSTAT(R1) ;Set dev status flags
```

```
    115 014052  012761  000000C 000000G          MOV     #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ;Device info flags
    116 014060  005061  000000G                 CLR     DEVSIZ(R1)      ;Clear device size
    117 014064  012761  000006G 000000G          MOV     #CLHEAD+6,HANENT(R1) ;Set handler entry point (4th word)
    118 014072  012761  000004  000000G          MOV     #4.,HANSIZ(R1)  ;Set size of handler
    119                                 ;
    120                                 ;  Finished
    121                                 ;
    122 014100  012603                  9$:     MOV     (SP)+,R3
    123 014102  012602                          MOV     (SP)+,R2
    124 014104  012601                          MOV     (SP)+,R1
    125 014106  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
                                ;
014110  010246                  INDINI: MOV     R2,-(SP)
014112  010346                          MOV     R3,-(SP)
                                ;
                                ;  Determine if IND support is wanted
                                ;
014114  005037  000000G                 CLR     INDSAV          ;ASSUME IND SUPPORT NOT WANTED
                                ;
                                ;  Lookup SY:IND.SAV file
                                ;
014120  013737  000000G 000220'         MOV     SYNAME,INDNAM   ;LOOK UP IND ON BOOT DEVICE
014126                                  .LOOKUP #AREA,#1,#INDNAM ;TRY TO FIND SY:IND.SAV
014146  103002                          BCC     4$              ;BR IF FOUND IND
014150  000137  014602'                 JMP     9$              ;IF CAN'T FIND IND, THEN NO IND SUPPORT
                                ;
                                ;  Set up information about IND overlay data segment
                                ;
014154  013703  000152'         4$:    MOV     WRKBUF,R3       ;Get pointer to work buffer
014160                                  .READW  #AREA,#1,R3,#256.,#0 ;READ IN BLOCK 0 OF SAV FILE
014214  016302  000064                  MOV     64(R3),R2       ;GET POINTER TO OVERLAY TABLE
014220                                  .READW  #AREA,#1,R3,#256.,#1 ;READ IN BLOCK 1 WITH OVERLAY TABLE
014256  162702  001000                  SUB     #1000,R2        ;GET ADDRESS OF OVERLAY TABLE REL TO BLOCK 1
014262  060302                          ADD     R3,R2           ;ADD BASE ADDRESS WHERE BLOCK 1 DATA IS
014264  011203                          MOV     (R2),R3         ;Get virtual address of segment 0
014266  020312                  5$:     CMP     R3,(R2)         ;Search for 1st segment with different addr
014270  001003                          BNE     6$              ;Br if found it (this is the data segment)
014272  062702  000006                  ADD     #6,R2           ;Point to overlay table entry for next seg
014276  000773                          BR      5$
014300  005722                  6$:     TST     (R2)+           ;Point to word with block # if SAV file
014302  012237  000000G                 MOV     (R2)+,INDDBL    ;GET BLOCK # IN SAV FILE OF DATA OVERLAY
014306  011202                          MOV     (R2),R2         ;GET # OF WORDS IN OVERLAY SEGMENT
014310  062702  000377                  ADD     #255.,R2        ;ROUND UP TO NEXT BLOCK
014314  000302                          SWAB    R2              ;CONVERT # WORDS TO # BLOCKS
014316  042702  177400                  BIC     #^C<377>,R2
014322  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.
                                ;
014326  012700  000000G                 MOV     #INDSAV,R0      ;GET ADDRESS OF SAVESTATUS BLOCK
014332  013702  000220'                 MOV     INDNAM,R2       ;GET RAD50 DEVICE NAME
014336  004737  024410'                 CALL    SETCHN          ;SAVE FILE STATUS
                                ;
                                ;  Determine how much space is needed for SY:TSXIND.TSX swap file
                                ;
014342  013703  000000G                 MOV     INDDBS,R3       ;GET # BLOCKS NEEDED PER JOB
014346  070327  000000C                 MUL     #<LSTSL/2>,R3   ;TIMES TOTAL NUMBER OF JOBS
```

```
                                        ;
                                        ;  Load Rt-11 device handler for ind swap file.
                                        ;
014352  013700  000000G                       MOV     INDFIL,R0       ;Get name of the device
014356  004737  024770'                       CALL    RTFTCH          ;Try to fetch the RT-11 device handler
014362  103522                               BCS     11$             ;Br if error on handler fetch
                                        ;
                                        ;  See if TSXIND file already exists
                                        ;
014364                                       .LOOKUP #AREA,#1,#INDFIL ;DOES SY:TSXIND.TSX FILE EXIST NOW?
014404  103415                               BCS     1$              ;BR IF NOT
014406  020003                               CMP     R0,R3           ;IS IT OF THE CORRECT SIZE?
014410  001462                               BEQ     2$              ;BR IF YES
014412                                       .PURGE  #1              ;FILE IS OF WRONG SIZE
014420                                       .DELETE #AREA,#1,#INDFIL;DELETE OLD FILE
                                        ;
                                        ;  File does not now exist
                                        ;  Create new file
                                        ;
014440                                  1$:  .ENTER  #AREA,#1,#INDFIL,R3 ;CREATE NEW TSXIND FILE
014464  103451                               BCS     10$             ;BR IF ERROR ON CREATE
014466  010302                               MOV     R3,R2           ;# BLOCKS IN FILE
014470  005302                               DEC     R2              ;GET # OF LAST BLOCK IN FILE
014472                                       .WRITW  #AREA,#1,WRKBUF,#256.,R2 ;WRITE TO LAST BLOCK OF FILE
014530                                       .CLOSE  #1              ;NOW CLOSE THE FILE
014536                                       .LOOKUP #AREA,#1,#INDFIL ;REOPEN TSXIND FILE WITH LOOKUP
                                        ;
                                        ;  Do .SAVESTATUS for SY:TSXIND.TSX file
                                        ;
014556  012700  000000G                  2$:  MOV     #INDTSV,R0      ;POINT TO SAVESTATUS BLOCK
014562  013702  000000G                       MOV     INDFIL,R2       ;GET RAD50 DEVICE NAME
014566  004737  024410'                       CALL    SETCHN          ;SAVE FILE INFO
014572                                       .RELEAS #INDFIL         ;Release device handler
                                        ;
                                        ;  Finished
                                        ;
014602  012603                           9$:  MOV     (SP)+,R3
014604  012602                               MOV     (SP)+,R2
014606  000207                               RETURN
                                        ;
                                        ;  Error occurred while opening SY:TSXIND.TSX file
                                        ;
014610                                  10$: .PRINT  #TSXHD          ;Print error message
014616                                       .PRINT  #INDOPN
014624  004737  010142'                       CALL    SPNEED          ;Print info about number of blocks needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
014630  010001                          11$: MOV     R0,R1           ;Save device name
014632                                       .PRINT  #TSXHD          ;Print error message
014640                                       .PRINT  #INDOPN
014646  004737  010170'                       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.
                                ;
014652  010246                  UCLINI: MOV     R2,-(SP)
014654  010346                          MOV     R3,-(SP)
                                ;
                                ;  Determine if TSXUCL data file is needed
                                ;
014656  105737  000000G                 TSTB    VU$CL           ;Is TSXUCL being used at all?
014662  001536                          BEQ     9$              ;Br if not
014664  013702  000000G                 MOV     VUCLMC,R2       ;Get maximum number of commands
014670  001533                          BEQ     9$              ;Br if none allowed
                                ;
                                ;  Determine number of blocks needed in data file for each job
                                ;
014672  012700  000000G                 MOV     #UK$$SZ,R0      ;Size of each keyword descriptor
014676  062700  000000G                 ADD     #US$$SZ,R0      ;Size of each command string descriptor
014702  070200                          MUL     R0,R2           ;Compute total # bytes for keywords+commands
014704  062703  000777G                 ADD     #UC$$SZ+511.,R3 ;Add space for control information & round up
014710  005502                          ADC     R2              ;Propogate carry
014712  071227  001000                  DIV     #512.,R2        ;Convert to # of blocks needed
014716  010237  000000G                 MOV     R2,UCLBLK       ;Save number of blocks needed per job
                                ;
                                ;  Multiply by number of jobs to get total file size
                                ;
014722  070227  000000C                 MUL     #<LSTSL/2>,R2   ;Times total number of jobs
                                ;
                                ;  Load Rt-11 device handler for ind swap file.
                                ;
014726  013700  000000G                 MOV     UCLDAT,R0       ;Get name of the device
014732  004737  024770'                 CALL    RTFTCH          ;Try to fetch the RT-11 device handler
014736  103523                          BCS     11$             ;Br if error on handler fetch
                                ;
                                ;  The total required file size is now in R3.
                                ;  See if the file already exists.
                                ;
014740                                  .LOOKUP #AREA,#1,#UCLDAT ;See if the file exists now
014760  103415                          BCS     1$              ;Br if file does not exist
014762  020003                          CMP     R0,R3           ;Is existing file of correct size?
014764  001446                          BEQ     2$              ;Br if yes -- use the old file
014766                                  .PURGE  #1              ;Purge the channel
014774                                  .DELETE #AREA,#1,#UCLDAT;Delete the old file
                                ;
                                ;  Create a new data file
                                ;
015014                          1$:     .ENTER  #AREA,#1,#UCLDAT,R3 ;Create new data file
015040  103452                          BCS     10$             ;Br if error creating the file
015042  005303                          DEC     R3              ;Get # of last block in the file
015044                                  .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.)
                                        ;
015102                          2$:     .CSTAT  #AREA,#1,#NFSBLK        ;GET CHANNEL STATUS INFORMATION
015122  013702  000032'                 MOV     <NFSBLK+12>,R2  ;FETCH DEVICE NAME IN RAD50
015126  063702  000030'                 ADD     <NFSBLK+10>,R2  ;ADD IN DEVICE UNIT NUMBER
015132  062702  000036                  ADD     #^R  0,R2       ;CONVERT UNIT NUMBER TO RAD50
015136  010237  000000G                 MOV     R2,UCLDAT       ;SET PHYSICAL NAME BACK INTO TSGEN CELL
015142                                  .CLOSE  #1              ;Close the file
015150                                  .RELEAS #UCLDAT         ;Release device handler
                                        ;
                                        ;  Finished
                                        ;
015160  012603                  9$:     MOV     (SP)+,R3
015162  012602                          MOV     (SP)+,R2
015164  000207                          RETURN
                                        ;
                                        ;  Error creating the data file
                                        ;
015166                          10$:    .PRINT  #TSXHD          ;Print error message
015174                                  .PRINT  #UCLOPN
015202  004737  010142'                 CALL    SPNEED          ;Print info about number of blocks needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
015206  010001                  11$:    MOV     R0,R1           ;Save device name
015210                                  .PRINT  #TSXHD          ;Print error message
015216                                  .PRINT  #UCLOPN
015224  004737  010170'                 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.
                                ;
015230  010146                  MEMINI: MOV     R1,-(SP)
015232  010246                          MOV     R2,-(SP)
015234  010346                          MOV     R3,-(SP)
015236  010446                          MOV     R4,-(SP)
015240  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize all pages for a 1-to-1 mapping.
                                ;
015242  012700  000000G         12$:    MOV     #KPAR0,R0       ;Kernel mode PAR 0
015246  012701  000000G                 MOV     #UPAR0,R1       ;User mode PAR 0
015252  012702  000000G                 MOV     #KPDR0,R2       ;Kernel mode PDR 0
015256  012703  000000G                 MOV     #UPDR0,R3       ;User mode PDR 0
015262  012704  000010                  MOV     #8.,R4          ;Initialize 8 pages
015266  005005                          CLR     R5              ;Set initial PAR value
015270  010520                  2$:     MOV     R5,(R0)+        ;Set kernel PAR
015272  010521                          MOV     R5,(R1)+        ;Set user PAR value
015274  012722  077406                  MOV     #077406,(R2)+   ;Set kernel PDR
015300  012723  077406                  MOV     #077406,(R3)+   ;Set user PDR value
015304  062705  000200                  ADD     #200,R5         ;Advance block number
015310  077411                          SOB     R4,2$           ;Init all pages
                                ;
                                ;  Map kernel mode I/O page (160000) to 17760000.
                                ;
015312  012737  000000G 000000G         MOV     #IOPAGE,@#KPAR7 ;Map I/O page
                                ;
                                ;  Finished
                                ;
015320  012605                          MOV     (SP)+,R5
015322  012604                          MOV     (SP)+,R4
015324  012603                          MOV     (SP)+,R3
015326  012602                          MOV     (SP)+,R2
015330  012601                          MOV     (SP)+,R1
015332  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

015334  010146                  MEMTST: MOV     R1,-(SP)
015336  010246                          MOV     R2,-(SP)
015340  010346                          MOV     R3,-(SP)
015342  010446                          MOV     R4,-(SP)
015344  010546                          MOV     R5,-(SP)
015346  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.
                                ;
015352  012737  015606' 000004          MOV     #TRCSET,@#4     ;Catch trap
015360  000240                          NOP                     ;Clean out 11/73 pipeline
015362  000240                          NOP                     ;Before attempting trap
015364  005737  000000G               TST     @#SR3MMR        ;Try to access status register 3
015370  103411                          BCS     22$             ;Br if MMU 3 status register is non-existent
015372  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!
                                ;
015376  000240                          NOP                     ;Clean out the pipeline again?
015400  000240                          NOP
015402  005737  000000G               TST     @#UDDRO         ;It there a user D-space PDRO?
015406  103402                          BCS     22$             ;Br if not
015410  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.
                                ;
015414                          22$:
                                .IF     NE,PROASM
015414  105737  000000G               TSTB    PROFLG          ;Are we running on a Professional?
015420  001410                          BEQ     26$             ;Br if not
015422  005005                          CLR     R5              ;Load byte without sign extension
```

```
     58 015424  153705  173050                  BISB    @#173050,R5     ;Get 32Kb top of system RAM boundary
     59 015430  072527  000011                  ASH     #9.,R5          ;Convert to # 64 byte blocks
     60 015434  162705  000010                  SUB     #10,R5          ;Don't use the last 512 bytes of memory
     61 015440  000400                          BR      7$
     62 015442                          26$:
     63                                 .ENDC   ;NE,PROASM
     64                                 .IF     NE,<PROASM-1>           ;Assemble if could be on a PDP-11
     65                                 ;
     66                                 ;  We are not running on a Professional.
     67                                 ;  Test each page above TSX to see where the top of memory is.
     68                                 ;
     69                                         MOV     #RTNKM,@#20     ;Use IOT instruction to get out of user mode
     70                                         CLR     @#22
     71                                         BIS     #MMENBL,@#SROMMR;Enable memory management
     72                                         TSTB    SR3FLG          ;Does maching have mem management reg # 3?
     73                                         BEQ     4$              ;Br if non-existent
     74                                         BIS     #EMMAP,@#SR3MMR ;Enable 22-bit extended memory
     75                                 ;
     76                                 ;  Map user page 7 to each successive 256-word block and attempt to access.
     77                                 ;
     78                                 4$:     MOV     #1024.,R5       ;Start checking at 64Kb
     79                                 5$:     MOV     R5,@#UPAR7      ;Map user page 7 to page to be tested
     80                                         BIS     #UMODE,@#PSW    ;Go into user mode
     81                                         TST     @#160000        ;Can we access the page?
     82                                 ;
     83                                 ;  Use IOT to get back into kernel mode.
     84                                 ;
     85                                         IOT                     ;Return to kernel mode
     86                                         BCS     6$              ;Br if memory is non-existent
     87                                         ADD     #10,R5          ;Go try next page
     88                                         CMP     R5,#177600      ;Don't enter I/O page
     89                                         BLO     5$
     90                                 ;
     91                                 ;  Check for potential memory wrap (on 18-bit 256K byte computers).
     92                                 ;
     93                                 6$:     CMP     R5,#10000       ;Is physical memory above 256K bytes
     94                                         BLOS    7$              ;Br if below 256K bytes
     95                                         CLR     @#TSTWRD        ;Clear physical location
     96                                         MOV     #10000,@#UPAR7  ;Map to 256K byte boundary
     97                                         BIS     #UMODE,@#PSW    ;Go into user mode
     98                                         MOV     #-1,@#160000+TSTWRD     ;Store -1 at 256K physical location
     99                                         IOT                     ;Return to kernel mode
    100                                         TST     @#TSTWRD        ;Test physical location
    101                                         BEQ     7$              ;Br if physical location is clear
    102                                         MOV     #10000,R5       ;Constrain memory to 256K byte total
    103                                 .ENDC   ;NE,<PROASM-1>
    104                                 ;
    105                                 ;  Reached end of available memory.
    106                                 ;
    107 015442  010537  000000G         7$:     MOV     R5,PHYMEM       ;set physical memory size
    108 015446  020537  000000G                 CMP     R5,MAPSIZ       ;Constrain kernel to user specified cutoff
    109 015452  101402                          BLOS    8$              ;Br if below user specified
    110 015454  013705  000000G                 MOV     MAPSIZ,R5       ;Only use this much memory
    111 015460  010537  000000G         8$:     MOV     R5,$MEMSZ       ;Set # 64-byte blocks of total memory
    112 015464  010537  000134'                 MOV     R5,FMEMHI       ;Save base 64-byte block # of top of free mem
    113                                 ;
    114                                 ;  Turn off memory management
```

```
                                        ;
015470  105737  000000G                 TSTB    SR3FLG          ;Do we have memory management reg # 3?
015474  001403                          BEQ     9$              ;Br if non-existent
015476  042737  000000G 000000G         BIC     #EMMAP,@#SR3MMR ;Disable extended memory management
015504  042737  000000G 000000G 9$:     BIC     #MMENBL,@#SROMMR;Turn off memory management
                                        ;
                                        ;  If this is a Q-bus machine with >256Kb then set EXTLSI flag in ICONFG
                                        ;
015512  023727  000134' 010000          CMP     FMEMHI,#4096.   ;Does machine have at least 256Kb?
015520  103411                          BLO     25$             ;Br if not
015522  105237  000000G                 INCB    MEM256          ;Remember machine has at least 256kb
015526  123727  000000G 000000G         CMPB    VBUSTP,#QBUS    ;Is this a Q-bus machine?
015534  001003                          BNE     25$             ;Br if not
015536  052737  000001  000242'         BIS     #EXTLSI,ICONFG  ;Set extended-LSI flag in ICONFG
                                        ;
                                        ;  See if this machine needs UNIBUS mapping
                                        ;
015544  123727  000000G 000000G 25$:     CMPB    VBUSTP,#UNIBUS  ;Is this a UNIBUS machine?
015552  001005                          BNE     29$             ;Br if not
015554  105737  000000G                 TSTB    MEM256          ;Does machine have at least 256kb of memory?
015560  001402                          BEQ     29$             ;Br if not
015562  105237  000000G                 INCB    UBUSMP          ;Say UNIBUS mapping is needed
                                        ;
                                        ;  Finished
                                        ;
015566  012637  000004          29$:    MOV     (SP)+,@#4       ;Reset trap vector
015572  012605                          MOV     (SP)+,R5
015574  012604                          MOV     (SP)+,R4
015576  012603                          MOV     (SP)+,R3
015600  012602                          MOV     (SP)+,R2
015602  012601                          MOV     (SP)+,R1
015604  000207                          RETURN
                                        ;
                                        ;  Trap - return with C-bit set.
                                        ;
015606  052766  000001  000002  TRCSET: BIS     #1,2(SP)        ;Set c-bit for return
015614  000002                          RTI                     ;Return from trap
                                        ;
                                        ;  IOT - return at kernel mode with c-bit preserved.
                                        ;
015616  042766  000000G 000002  RTNKM:  BIC     #UMODE,2(SP)    ;Clear user mode - return to kernel
015624  000002                          RTI                     ;Return from trap
                                        ;
                                        ;  Error: System does not have memory management hardware.
                                        ;
015626                          NOXM:   .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
015634                                  .PRINT  #NXMMSG
015642  000137  004134'                 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.
                                ;
015646                          CXTALC:
                                ;
                                ;  Get size of base portion of job context area
                                ;
015646  012700  000000G                 MOV     #CXTSIZ,R0      ;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.
                                ;
015652  062700  000077                  ADD     #63.,R0         ;Bound up to 64 byte boundary
015656  042700  000077                  BIC     #77,R0
015662  010037  000000G                 MOV     R0,CXTRMN       ;Offset to start of simulated RMON
015666  062737  000000G 000000G         ADD     #CXTBAS,CXTRMN  ;Add base virtual address of context area
015674  062700  000001G                 ADD     #MVSIZ+1,R0     ;Add space for simulated RMON & channels
                                ;
                                ;  Save number of words needed for context area
                                ;
015700  006200                          ASR     R0              ;Convert to # words
015702  010037  000000G                 MOV     R0,CXTWDS       ;This is # words for whole job context area
                                ;
                                ;  Compute PDR value to use to map to job context area
                                ;
015706  062700  000037                  ADD     #31.,R0         ;Bound up to # 32 word units
015712  072027  177773                  ASH     #-5.,R0         ;Get # 32-word units for context area
015716  000300                          SWAB    R0              ;Put # 32-word units in high-order byte
015720  052700  000006                  BIS     #6,R0           ;Set PDR control flags
015724  010037  000000G                 MOV     R0,CXTPDR       ;This is the PDR value
                                ;
                                ;  Compute # 512-byte pages needed for job context block
                                ;
015730  013700  000000G                 MOV     CXTWDS,R0       ;Get back # words for context area
015734  062700  000377                  ADD     #255.,R0        ;Bound up to # 256-word blocks
015740  072027  177770                  ASH     #-8.,R0         ;Get # 256-word pages for context area
015744  010037  000000G                 MOV     R0,CXTPAG       ;# pages for job context area
                                ;
                                ;  Set up PDR value used when mapping to simulated RMON
                                ;
015750  012700  000000G                 MOV     #MVSIZ,R0       ;Get size of monitor vector table
015754  062700  000077                  ADD     #63.,R0         ;Round up to # 32 word blocks
015760  072027  177772                  ASH     #-6,R0          ;Cvt to # 32-word blocks
015764  005300                          DEC     R0              ;Get # blocks - 1
015766  000300                          SWAB    R0              ;Put # blocks in left byte
015770  052700  000006                  BIS     #6,R0           ;Allow read and write access
015774  042700  100261                  BIC     #100261,R0      ;Make sure unused PDR bits are zero
016000  010037  000000G                 MOV     R0,RMNPDR       ;This is PDR value to map to sim. RMON
                                ;
```

```
                                        ;  Finished
                                        ;
60 016004  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.
                ;
016006  010246  MAPALC: MOV     R2,-(SP)
016010  010346          MOV     R3,-(SP)
016012  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.
                ;
016014  010503          MOV     R5,R3           ;Get 64-byte block # of top of free mem
016016  072327  177775  ASH     #-3,R3          ;Convert to 512-byte page #
016022  042703  160000  BIC     #160000,R3      ;Kill possible sign extension
016026  013702  000136' MOV     FMEMLO,R2       ;Get 64-byte block # of base of free memory
016032  062702  000007  ADD     #7,R2           ;Round up
016036  072227  177775  ASH     #-3,R2          ;Convert to 512-byte page #
016042  042702  160000  BIC     #160000,R2      ;Kill possible sign extension
016046  160203          SUB     R2,R3           ;Get # bytes needed for allocation table
016050  003440          BLE     10$             ;Br if memory overflow
016052  010304          MOV     R3,R4           ;Get # bytes for allocation table
016054  062704  001000  ADD     #512.,R4        ;Add 1 extra byte and round up to 512-byte
016060  072427  177767  ASH     #-9.,R4         ;Get # 512-byte units needed for alloc table
                ;
                ;  Set up virtual address pointers for the allocation table
                ;
016064  012700  000000G MOV     #VPAR5,R0       ;We will map to alloc table through PAR 5
016070  010037  000000G MOV     R0,LOMAP        ;Pointer to 1st entry in alloc table
016074  160200          SUB     R2,R0           ;Get pseudo virtual address for page # 0
016076  010037  000000G MOV     R0,BASMAP       ;This would point to alloc entry for page 0
016102  012700  000000G MOV     #VPAR5,R0       ;Get back base address of table
016106  060300          ADD     R3,R0           ;Add # bytes used by table
016110  160400          SUB     R4,R0           ;Subtract space used by table itself
016112  010037  000000G MOV     R0,HIMAP        ;Virtual address of 1st entry for system page
                ;
```

```
                                        ;  Allocate space for the allocation table
                                        ;
016116  072427  000003                          ASH     #3,R4           ;Get # 64-byte units for alloc table
016122  160405                                  SUB     R4,R5           ;Compute physical 64-byte base for table
016124  020537  000136'                        CMP     R5,FMEMLO       ;Did we run out of memory space?
016130  101410                                  BLOS    10$             ;Br if memory overflow
016132  010537  000000G                        MOV     R5,MAPPAR       ;Use this value to map PAR 5 to alloc table
016136  010537  000134'                        MOV     R5,FMEMHI       ;Save new top of free memory area
                                        ;
                                        ;  Finished
                                        ;
016142  012604                                  MOV     (SP)+,R4
016144  012603                                  MOV     (SP)+,R3
016146  012602                                  MOV     (SP)+,R2
016150  000207                                  RETURN
                                        ;
                                        ;  Error: Generated system is too large
                                        ;
016152                                  10$:    .PRINT  #TSXHD          ;Print error message heading
016160                                          .PRINT  #PHSOVF         ;Physical memory overflow
016166  000137  004134'                        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)
                                ;
016172  010546                  SETJSZ: MOV     R5,-(SP)
                                ;
                                ;  Determine total number of pages of memory available to user jobs
                                ;
016174  013705  000000G                 MOV     HIMAP,R5        ;POINTER ABOVE LAST FREE PAGE ENTRY
016200  163705  000000G                 SUB     LOMAP,R5        ;GET TOTAL # OF FREE PAGES
016204  010537  000000G                 MOV     R5,FREPGS       ;# FREE PAGES AVAILABLE TO USER JOBS
                                ;
                                ;  Make sure there is enough free space to run TSKMON.
                                ;
016210  020537  000000G                 CMP     R5,KMNPGS       ;COMPARE # FREE PAGES TO # PAGES FOR TSKMON
016214  103436                          BLO     1$              ;BR IF INSUFFICIENT MEMORY TO RUN TSKMON
                                ;
                                ;  Set up max memory limit for jobs
                                ;
016216  013700  000000G                 MOV     CXTPAG,R0       ;# PAGES NEEDED FOR JOB CONTEXT AREA
016222  010037  000000G                 MOV     R0,JCXPGS
016226  160005                          SUB     R0,R5           ;LEAVE ROOM FOR JOB CONTEXT AREA
016230  006205                          ASR     R5              ;Convert # pages to # KB
016232  020537  000000G                 CMP     R5,VHIMEM       ;COMPARE WITH TSGEN SPECIFIED MAX SIZE
016236  101402                          BLOS    10$             ;BR IF CONSTRAINED BY PHYSICAL SIZE
016240  013705  000000G                 MOV     VHIMEM,R5       ;LIMIT BY VALUE SPECIFIED IN TSGEN
016244  010537  000000G          10$:   MOV     R5,MXJMEM       ;MAX # K BYTES AVAILABLE TO A JOB
016250  010500                          MOV     R5,R0
016252  072027  000012                  ASH     #10.,R0         ;CONVERT # KB TO BYTE ADDRESS
016256  001002                          BNE     2$              ;BR IF DIDN'T OVERFLOW 64KB
016260  012700  177774                  MOV     #177774,R0      ;GET 64KB TOP ADDRESS
016264  010037  000000G          2$:    MOV     R0,MXJADR       ;ADDRESS ABOVE TOP OF JOB
                                ;
                                ;  Set default memory size of jobs
                                ;
016270  020537  000000G                 CMP     R5,VDFMEM       ;COMPARE TO DEFAULT SPECIFIED IN TSGEN
016274  101402                          BLOS    11$             ;BR IF CONSTRAINED BY PHYSICAL LIMIT
016276  013705  000000G                 MOV     VDFMEM,R5       ;CONSTRAIN BY TSGEN PARAMETER
016302  010537  000000G          11$:   MOV     R5,DFJMEM       ;SET DEFAULT JOB MEMORY SIZE
                                ;
                                ;  Finished
                                ;
016306  012605                          MOV     (SP)+,R5
016310  000207                          RETURN
                                ;
                                ;  Error -- Insufficient memory space available to run TSKMON.
```

```
     58                                     ;
     59 016312  004737  025124'            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.
                ;
                PARSET: TST     #MPARFL         ;Does he want to disable memory parity?
                        BNE     20$             ;Br if not
                        MOV     R2,-(SP)
                        MOV     @#4,-(SP)       ;Save contents of trap vector
                ;
                ;  Catch traps that occur when we access unimplemented parity registers
                ;
                        MOV     #2$,@#4         ;Send traps to 2$
                        NOP                     ;Clean out instruction pipeline
                        NOP
                ;
                ;  Disable parity for each block of memory
                ;
                        MOV     #MPAR0,R2       ;Point to 1st memory control register
                1$:     BIC     #PARENL,(R2)    ;Disable memory parity
                        BR      3$              ;We did not trap
                2$:     ADD     #4,SP           ;Clean trap PS and PC off of stack
                3$:     ADD     #2,R2           ;Point to next parity control register
                        CMP     R2,#MPAR16      ;Have we cleared all registers?
                        BLOS    1$              ;Loop if not
                ;
                ;  Finished
                ;
                        MOV     (SP)+,@#4       ;Restore trap vector
                        MOV     (SP)+,R2
                20$:    RETURN
                        .IFF    ;NE,<PROASM-1>
016316  000207  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.
                                ;
016320  010146                  GETHNL: MOV     R1,-(SP)
016322  010246                          MOV     R2,-(SP)
016324  010446                          MOV     R4,-(SP)
                                ;
                                ;  Begin loop to check all handlers specified in TSGEN with DEVDEF.
                                ;
016326  005001                          CLR     R1                      ;Init device table index
016330  020127  000000C         1$:     CMP     R1,#<AHEND-AUTHAN>      ;Done all devices?
016334  103015                          BHIS    2$                      ;Br if yes
016336  016102  000000G                 MOV     AUTHAN(R1),R2           ;Get the name of the device
016342  001407                          BEQ     3$                      ;Ignore null devices
016344  020227  000000G                 CMP     R2,#DMYDEV              ;Is this a dummy device entry?
016350  001404                          BEQ     3$                      ;Skip it if yes
016352  016104  000000G                 MOV     DTYPE(R1),R4            ;Get flags specified in TSGEN
                                ;
                                ;  Load this handler
                                ;
016356  004737  016430'                 CALL    LDHAND                  ;Try to load handler into memory
                                ;
                                ;  Check next device
                                ;
016362  062701  000002          3$:     ADD     #2,R1                   ;Advance device index
016366  000760                          BR      1$                      ;See if more devices to load
                                ;
                                ;  Now see if there are spooled devices to contend with
                                ;
016370  012704  000000G         2$:     MOV     #SPLND,R4               ;Are there any spooled devices?
016374  001411                          BEQ     9$                      ;Br if not
016376  012701  000000G                 MOV     #SPLDEV,R1              ;Point to spooled device name table
016402  012102                  5$:     MOV     (R1)+,R2                ;Get the name of the next spooled device
016404  010446                          MOV     R4,-(SP)                ;Save device count
016406  005004                          CLR     R4                      ;Say no TSGEN flags for device
016410  004737  016430'                 CALL    LDHAND                  ;Load the handler
016414  012604                          MOV     (SP)+,R4                ;Recover the device count
016416  077407                          SOB     R4,5$                   ;Loop if more handlers to load
                                ;
                                ;  Finished
                                ;
016420  012604                  9$:     MOV     (SP)+,R4
016422  012602                          MOV     (SP)+,R2
016424  012601                          MOV     (SP)+,R1
016426  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$xxx status flags for device from TSGEN.
                                ;   R5 = Address where handler is to be loaded.
                                ;
                                ;  Outputs:
                                ;   R5 = New free memory address.
                                ;   NUMDEV = Incremented by 2.
                                ;   PNAME(i) = Rad-50 name of device.
                                ;   ENTRY(i) = Handler entry point.
                                ;   DVSTAT(i) = Device status flags.
                                ;   DVFLAG(i) = TSX-Plus device status flags.
                                ;   HANPAR(i) = PAR offset if this is a mapped handler.
                                ;
016430  010446                 LDHAND: MOV     R4,-(SP)
                                ;
                                ;  Determine if we should ignore this device
                                ;
016432  004737  016602'                 CALL    INSCK1          ;Should we ignore this device?
016436  103457                         BCS     9$              ;Br if yes
                                ;
                                ;  The initial tests indicate that this handler should be loaded.
                                ;  Now open the handler file and perform some additional checks.
                                ;
016440  004737  016760'                 CALL    INSCK2          ;Perform some additional checks on handler
016444  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.
                                ;
016446  004737  017374'                 CALL    STDVTB          ;Set up info in tables for this device
                                ;
                                ;  Determine if this handler is to be loaded into low memory or
                                ;  extended memory.
                                ;
016452  016400  000000G                 MOV     DVFLAG(R4),R0   ;Get TSX-Plus status flags for device
016456  032700  000000G                 BIT     #DX$NHM,R0      ;Are we never to map this handler?
016462  001036                         BNE     1$              ;Br if handler cannot be mapped
016464  032700  000000G                 BIT     #DX$MPH,R0      ;Is mapping wanted for this handler?
016470  001433                         BEQ     1$              ;Br if not
016472  032700  000000G                 BIT     #DX$IBH,R0      ;Does this handler have an internal I/O buff?
016476  001412                         BEQ     2$              ;Br if not
016500  105737  000000G                 TSTB    UBUSMP          ;Does this machine have a mapped UNIBUS?
016504  001025                         BNE     1$              ;Br if yes -- Don't map this handler
016506  032700  000000G                 BIT     #DX$MAP,R0      ;Does this handler require I/O mapping?
016512  001404                         BEQ     2$              ;Br if not
016514  032737  000001  000242'        BIT     #EXTLSI,ICONFG  ;Is this a Q-bus system with more than 256Kb?
016522  001016                         BNE     1$              ;Br if yes -- Don't map this handler
                                ;
                                ;  This handler can be mapped and will be loaded in extended memory
```

```
     58                                     ;
     59 016524  005237  000124'             2$:     INC     NMXHAN          ;Count # of mapped handlers
     60 016530  012764  000001  000000G             MOV     #1,HANPAR(R4)   ;Set flag saying handler should be mapped
     61                                     ;
     62                                     ;  Make sure size of mapped handler does not exceed 8KB
     63                                     ;
     64 016536  026427  000000G 020000              CMP     HANSIZ(R4),#8192. ;Is mapped handler too big?
     65 016544  101411                              BLOS    8$              ;Br if not too big
     66 016546  012700  002335'                     MOV     #HN2BIG,R0      ;Get error message address
     67 016552  004737  020472'                     CALL    HLERR           ;Abort initialization if iniabt
     68 016556  000404                              BR      8$
     69                                     ;
     70                                     ;  This handler must be loaded into low memory
     71                                     ;
     72 016560  005064  000000G             1$:     CLR     HANPAR(R4)      ;Say this handler is not mapped
     73 016564  004737  017504'                     CALL    LDHNLO          ;Load handler into low memory
     74                                     ;
     75                                     ;  Close the handler file
     76                                     ;
     77 016570                              8$:     .CLOSE  #1              ;Close the handler file
     78                                     ;
     79                                     ;  Finished
     80                                     ;
     81 016576  012604                      9$:     MOV     (SP)+,R4
     82 016600  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.
                                ;
016602  010146                  INSCK1: MOV     R1,-(SP)
                                ;
                                ;  Strip off any specified unit number
                                ;
016604  010201                          MOV     R2,R1           ;Get full device name
016606  005000                          CLR     R0              ;Set for divide
016610  071027  000050                  DIV     #50,R0          ;Split off last digit
016614  070027  000050                  MUL     #50,R0          ;Now correct for divide
016620  010102                          MOV     R1,R2           ;Get device name less 3rd digit
016622  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.
                                ;
016626  020227  045640                  CMP     R2,#R50LD       ;Is device name LD?
016632  001004                          BNE     1$              ;Br if not
016634  105737  0000000G                TSTB    VLDSYS          ;Is standard system LD support included?
016640  001044                          BNE     5$              ;Br if yes -- Don't load LD
016642  000417                          BR      3$              ;Load LD
016644  012701  000160'          1$:     MOV     #SKPDEV,R1      ;Point to table of devices to skip
016650  020221                  2$:     CMP     R2,(R1)+        ;Is this a device to be skipped?
016652  001437                          BEQ     5$              ;Br if yes
016654  005711                          TST     (R1)            ;Reached end of skip list?
016656  001374                          BNE     2$              ;Loop if not
                                ;
                                ;  See if we have already loaded the handler for this device
                                ;
016660  013701  0000000G                MOV     NUMDEV,R1       ;Get index for last device
016664  001406                          BEQ     3$              ;Br if no devices installed yet
016666  020261  0000000G          4$:     CMP     R2,PNAME(R1)    ;See if this device is already installed
016672  001427                          BEQ     5$              ;Br if already installed
016674  162701  000002                  SUB     #2,R1           ;More installed devices to check?
016700  003372                          BGT     4$              ;Loop if yes
                                ;
                                ;  This handler is to be loaded.
                                ;  Get default TSX-Plus control flags for this device.
                                ;
016702  012701  000336'          3$:     MOV     #DVFLBS,R1      ;Point to start of table
016706  020261  000000          6$:     CMP     R2,DV$NAM(R1)   ;Search for device in the table
016712  001003                          BNE     7$              ;Br if this is not it
016714  056104  000002                  BIS     DV$FLG(R1),R4   ;Combine default flags
016720  000405                          BR      8$
```

```
     58 016722  062701  000004          7$:     ADD     #DV$$SZ,R1      ;Point to next entry
     59 016726  020127  000516'                 CMP     R1,#DVFLND      ;Checked all entries?
     60 016732  103765                          BLO     6$              ;Loop if not
     61                                 ;
     62                                 ;  If this is a DMA device, set flag saying buffers must be on
     63                                 ;  even byte boundaries.
     64                                 ;
     65 016734  032704  000000G         8$:     BIT     #DX$DMA,R4      ;Is this a DMA device?
     66 016740  001402                          BEQ     10$             ;Br if not
     67 016742  052704  000000G                 BIS     #DX$EBA,R4      ;Set even-buffer-boundary flag
     68                                 ;
     69                                 ;  Load this handler
     70                                 ;
     71 016746  000241                  10$:    CLC                     ;Set flag saying to load the handler
     72 016750  000401                          BR      9$
     73                                 ;
     74                                 ;  Do not load this handler
     75                                 ;
     76 016752  000261                  5$:     SEC                     ;Set flag saying not to load the handler
     77                                 ;
     78                                 ;  Finished
     79                                 ;
     80 016754  012601                  9$:     MOV     (SP)+,R1
     81 016756  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.
                                ;
016760  010146                  INSCK2: MOV     R1,-(SP)
016762  010246                          MOV     R2,-(SP)
016764  010346                          MOV     R3,-(SP)
016766  010446                          MOV     R4,-(SP)
016770  010546                          MOV     R5,-(SP)
016772  013746  000004                  MOV     @#4,-(SP)       ;Save the bus timeout vector
016776  013746  000010                  MOV     @#10,-(SP)      ;Save illegal instruction vector
                                ;
                                ;  Try to lookup handler file on system disk
                                ;
017002  010237  000202'                 MOV     R2,HANNAM+2     ;Set the device name for the lookup
                                ;;; Don't change channel # without changing STDVTB
017006                                  .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
017026  103006                          BCC     1$              ;Br if we found the handler file
                                ;
                                ;  Error -- Cannot find handler file
                                ;
017030  012700  001510'         2$:     MOV     #CFHMSG,R0      ;Can't find handler
017034  004737  020472'                 CALL    HLERR           ;See if should abort initialization
017040  000137  017350'                 JMP     13$
                                ;
                                ;  We were able to open the handler file.
                                ;  Read in block 0 of handler.
                                ;
017044                          1$:     .READW  #AREA,#1,WRKBUF,#256.,#0 ;Read block 0 into WRKBUF
017102  103006                          BCC     3$              ;Br if read ok
017104  012700  001552'                 MOV     #ERHMSG,R0      ;Error during read
017110  004737  020472'                 CALL    HLERR
017114  000137  017350'                 JMP     13$
                                ;
                                ;  Determine if the handler is supported under the current RT-11 version
                                ;
017120  012700  000322'         3$:     MOV     #HVTBL,R0       ;Point to table with handler version info
017124  013701  000152'                 MOV     WRKBUF,R1       ;Point to buffer with block 0 of handler
017130  116101  000000G                 MOVB    H.DSTS(R1),R1   ;Get device ID code from handler
017134  120160  000000          51$:    CMPB    R1,HV$ID(R0)    ;Compare handler ID code with table entry
017140  001011                          BNE     53$             ;Br if this entry not for this handler
017142  026037  000002  000156'        CMP     HV$VER(R0),RTVPTR ;Is handler valid for this version?
017150  101412                          BLOS    54$             ;Br if OK
017152  012700  001616'                 MOV     #ERHNDV,R0      ;Wrong version of RT for handler
017156  004737  020472'                 CALL    HLERR           ;See if should Report error and abort
017162  000472                          BR      13$
017164  062700  000004          53$:    ADD     #HV$$SZ,R0      ;Point to next handler version table entry
017170  020027  000336'                 CMP     R0,#HVEND       ;Are there more entries?
```

```
     58 017174  103757                          BLO     51$             ;Loop if more to check
     59                                 ;
     60                                 ;  Check handler sysgen options
     61                                 ;
     62 017176  013700  000152'         54$:    MOV     WRKBUF,R0       ;Point to buffer with handler block 0
     63 017202  032760  000000G 000000G        BIT     #SG$MMU,H.GEN(R0);Was handler genned with XM support?
     64 017210  001005                          BNE     4$              ;Br if yes
     65 017212  012700  002027'                MOV     #HSGER,R0       ;Error if not XM version of handler
     66 017216  004737  020472'                CALL    HLERR
     67 017222  000452                          BR      13$
     68 017224  016037  000000G 000240' 4$:    MOV     H.GEN(R0),HGENFL;Save handler sysgen flags for later
     69                                 ;
     70                                 ;  Check the CSR address specified in the handler to see if the
     71                                 ;  hardware device for this handler exists.
     72                                 ;
     73 017232  012737  015606' 000004          MOV     #TRCSET,@#4     ;Catch bus timeout traps
     74 017240  012737  015606' 000010          MOV     #TRCSET,@#10    ;Catch illegal instruction traps
     75 017246  013700  000152'                MOV     WRKBUF,R0       ;Point to start of block 0 of handler
     76 017252  016001  000000G                MOV     H.CSR(R0),R1    ;Get address of CSR for device
     77 017256  001407                          BEQ     5$              ;Br if no CSR specified
     78 017260  005711                          TST     (R1)            ;Is CSR accessible?
     79 017262  103005                          BCC     5$              ;Br if ok
     80 017264  012700  001670'                MOV     #NOCSRR,R0      ;Trap occurred while accessing CSR
     81 017270  004737  020472'                CALL    HLERR           ;See if should report error and abort
     82 017274  000425                          BR      13$
     83                                 ;
     84                                 ;  Execute the device installation code.
     85                                 ;  The installation code will set the C-flag if the handler should
     86                                 ;  not be loaded.
     87                                 ;
     88 017276  062700  000000G         5$:     ADD     #H.INS,R0       ;Offset 200 in block 0
     89 017302  005710                          TST     @R0             ;Does any installation code exist?
     90 017304  001420                          BEQ     11$             ;Br if no driver installation code
     91 017306  013746  000000G                MOV     @#RMON,-(SP)    ;Save RT-11 RMON pointer
     92 017312  012737  000000G 000000G        MOV     #MONVEC,@#RMON  ;Set TSX-Plus RMON pointer
     93 017320  013703  000000G                MOV     RPRVEC,R3       ;Get pointer to Pro vec addr routine
     94 017324  004710                          CALL    @R0             ;Call the installation code
     95 017326  012637  000000G                MOV     (SP)+,@#RMON    ;Restore RT-11 RMON pointer
     96 017332  103006                          BCC     13$             ;C-flag now indicates handler load status
     97 017334  012700  001721'                MOV     #ERHINS,R0      ;Error occured in handler installation code
     98 017340  004737  020472'                CALL    HLERR
     99 017344  000401                          BR      13$
    100                                 ;
    101                                 ;  Finished with installation verification.
    102                                 ;
    103 017346  000241                 11$:    CLC                     ;Clear the c-bit for driver installation
    104 017350  012637  000010         13$:    MOV     (SP)+,@#10      ;Restore illegal instruction vector
    105 017354  012637  000004                 MOV     (SP)+,@#4       ;Restore the bus timeout vector
    106 017360  012605                         MOV     (SP)+,R5
    107 017362  012604                         MOV     (SP)+,R4
    108 017364  012603                         MOV     (SP)+,R3
    109 017366  012602                         MOV     (SP)+,R2
    110 017370  012601                         MOV     (SP)+,R1
    111 017372  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).
                                ;
017374                          STDVTB:
                                ;
                                ;  Increment device counter
                                ;
017374  062737  000002  000000G         ADD     #2,NUMDEV       ;Say another device added to tables
017402  013700  000000G                 MOV     NUMDEV,R0       ;Get device index number
                                ;
                                ;  Set up PNAME and DVFLAG.
                                ;
017406  010260  000000G                 MOV     R2,PNAME(R0)    ;Set permanent device name
017412  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.
                                ;
017416  012760  000001  000000G         MOV     #1,HANDSK(R0)   ;Set block 1 relative offset of handler file
                                ;
                                ;  Extract parameters from handler block 0
                                ;
017424  010004                          MOV     R0,R4           ;Carry device index in R4
017426  013700  000152'                 MOV     WRKBUF,R0       ;Point to block 0 of handler
017432  016064  000000G 000000G         MOV     H.SIZ(R0),HANSIZ(R4) ;Set handler size
017440  016064  000000G 000000G         MOV     H.DVSZ(R0),DEVSIZ(R4) ;Number of blocks on device
017446  016064  000000G 000000G         MOV     H.DSTS(R0),DVSTAT(R4) ;Set device status flags
                                ;
                                ;  Disable MOUNTs and data caching for certain devices
                                ;
017454  032764  000000G 000000G         BIT     #DS$DIR,DVSTAT(R4) ;Is this a directory structured device?
017462  001404                          BEQ     1$              ;Br if not -- No mounts allowed
017464  032764  000000G 000000G         BIT     #DS$NRD,DVSTAT(R4) ;Non RT-11 directory structure (mag tape)?
```

```
     58 017472  001403                            BEQ     9$                  ;Br if not
     59 017474  052764  000000C 000000G 1$:       BIS     #<DX$NMT!DX$NCA>,DVFLAG(R4) ;Disable mounts and data caching
     60                                ;
     61                                ;  Finished
     62                                ;
     63 017502  000207                 9$:       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.
                                ;
017504  010346                  LDHNLO: MOV     R3,-(SP)
                                ;
                                ;  Determine if we have enough free memory space to read the handler
                                ;
017506  005064  000000G                 CLR     HANPAR(R4)      ;Say this handler is not mapped
017512  010500                          MOV     R5,R0           ;Get current top of memory address
017514  016403  000000G                 MOV     HANSIZ(R4),R3   ;Get size of handler
017520  060300                          ADD     R3,R0           ;Get address above top of handler
017522  004737  025104'                 CALL    CHKMEM          ;See if handler will fit in memory
                                ;
                                ;  Handler will fit.  Read it into memory.
                                ;
017526  006203                          ASR     R3              ;Get number of words to read
017530                                  .READW  #AREA,#1,R5,R3,#1
017564  103005                          BCC     1$              ;Br if read ok
017566  012700  001552'                 MOV     #ERHMSG,R0      ;Error reading handler
017572  004737  020472'                 CALL    HLERR           ;See if should Abort initialization
017576  000414                          BR      2$
                                ;
                                ;  Set address of handler entry point and compute address beyond
                                ;  end of the handler.
                                ;
017600  010564  000000G         1$:     MOV     R5,HANENT(R4)   ;Set address of handler entry point
017604  062764  000006  000000G         ADD     #6,HANENT(R4)   ;(Point to fourth word of handler)
017612  006303                          ASL     R3              ;Convert handler size to bytes
017614  060305                          ADD     R3,R5           ;Point beyond end of handler
                                ;
                                ;  Set up table of addresses of support routines at end of handler.
                                ;
017616  010503                          MOV     R5,R3           ;Get address past end of handler
017620  004737  020400'                 CALL    STHNPV          ;Set up pointer vector in handler
                                ;
                                ;  If handler has any load-time exectuion code, run it now
                                ;
017624  004737  020542'                 CALL    DOHNLC          ;Run any load-time code for handler
                                ;
                                ;  Finished
                                ;
017630  012603                  2$:     MOV     (SP)+,R3
017632  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.
                                ;
017634  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.
                                ;
017636  012704  000002                  MOV     #2,R4           ;Get index for first device entry
                                ;
                                ;  See if this device has a mapped handler
                                ;
017642  005764  000000G         1$:     TST     HANPAR(R4)      ;Is this handler mapped?
017646  001402                          BEQ     2$              ;Br if not
                                ;
                                ;  We found an entry for a device with a mapped handler.
                                ;  Load the handler.
                                ;
017650  004737  017672'                 CALL    LDHNHI          ;Load a mapped handler
                                ;
                                ;  Look for more mapped handlers
                                ;
017654  062704  000002          2$:     ADD     #2,R4           ;Increment device index
017660  020437  000000G                 CMP     R4,NUMDEV       ;Checked all devices?
017664  101766                          BLOS    1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
017666  012604                          MOV     (SP)+,R4
017670  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.
                                ;
017672  010146                  LDHNHI: MOV     R1,-(SP)
017674  010246                          MOV     R2,-(SP)
017676  010346                          MOV     R3,-(SP)
017700  010446                          MOV     R4,-(SP)
                                ;
                                ;  Open channel 1 to the handler file.
                                ;
017702  016437  000000G 000202'          MOV     PNAME(R4),HANNAM+2 ;Set the device name for the lookup
017710  016437  000000G 000146'          MOV     PNAME(R4),CURNAM;Set name in case we have an error
017716                                   .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
017736  103006                           BCC     8$                 ;Br if we found the handler file
017740  012700  001510'                  MOV     #CFHMSG,R0         ;Can't find handler
017744  004737  020472'                  CALL    HLERR              ;See if should Abort initialization
017750  000137  020360'                  JMP     9$
                                ;
                                ;  Read block 0 of the handler file and extract some information
                                ;
017754  013702  000152'          8$:     MOV     WRKBUF,R2          ;Get address of work buffer
017760                                   .READW  #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
020014  016237  000000G 000240'          MOV     H.GEN(R2),HGENFL;Save handler sysgen flags
                                ;
                                ;  Set virtual address of handler entry point
                                ;
020022  012764  000006G 000000G          MOV     #VPAR5+6,HANENT(R4) ;Set virtual addr of handler entry point
                                ;
                                ;  Get information about the size of the handler and determine the
                                ;  address in extended memory where the handler is to be loaded.
                                ;
020030  016402  000000G                  MOV     HANSIZ(R4),R2      ;Get size of handler (bytes)
020034  005202                           INC     R2                 ;Make sure handler size is even
020036  042702  000001                   BIC     #1,R2
020042  010200                           MOV     R2,R0
020044  062700  000077                   ADD     #63.,R0            ;Round up to # 64-byte blocks
020050  072027  177772                   ASH     #-6,R0             ;Get # 64-byte blocks for handler
020054  060037  000000G                  ADD     R0,MHNSIZ          ;Accumulate total space for mapped handlers
020060  160005                           SUB     R0,R5              ;Reserve room for handler
020062  010564  000000G                  MOV     R5,HANPAR(R4)      ;Set mapping value for handler
020066  010537  000126'                  MOV     R5,HMAP            ;Set initial PAR base for handler
020072  012737  000001  000142'         MOV     #1,FILBLK          ;Set # of block to read from file
                                ;
                                ;  Begin loop to read handler into memory
                                ;
020100  010203                  1$:     MOV     R2,R3              ;Get remaining size of handler
020102  020327  001000                   CMP     R3,#512.           ;Compare with max we can read at one time
020106  101402                           BLOS    2$                 ;Br if we can read remainder of handler
020110  012703  001000                   MOV     #512.,R3           ;Read one block
```

```
     58 020114  160302                    2$:     SUB     R3,R2           ;Reduce amt of handler left to read
     59                                    ;
     60                                    ;  Read next block of handler
     61                                    ;
     62 020116  006203                            ASR     R3              ;Get # words to read
     63 020120  013701  000152'                    MOV     WRKBUF,R1       ;Get address of buffer for read
     64 020124                                    .READW  #AREA,#1,R1,R3,FILBLK ;Read a block
     65 020160  103005                            BCC     3$              ;Br if read ok
     66 020162  012700  001552'                    MOV     #ERHMSG,R0      ;Get error message
     67 020166  004737  020472'                    CALL    HLERR           ;See if should Abort initialization
     68 020172  000472                            BR      9$
     69                                    ;
     70                                    ;  Move the code we just read into the XM area for the handler
     71                                    ;
     72 020174  012700  000000G           3$:     MOV     #VPAR5,R0       ;Get virtual address of mapped region
     73 020200                                    DISABL                  ;** Disable interrupts **
     74 020206  013746  000000G                    MOV     @#KPAR5,-(SP)   ;;;Save current mapping of PAR 5
     75 020212  013737  000126' 000000G            MOV     HMAP,@#KPAR5    ;;;Set up mapping to get to XM area
     76 020220  052737  000000G 000000G            BIS     #MMENBL,@#SROMMR;;;Enable memory management
     77 020226  105737  000000G                    TSTB    MEM256          ;;;Does machine have > 256KB?
     78 020232  001403                            BEQ     4$              ;;;Br if not
     79 020234  052737  000000G 000000G            BIS     #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
     80 020242  012120                    4$:     MOV     (R1)+,(R0)+     ;;;Move from WRKBUF to XM region
     81 020244  077302                            SOB     R3,4$           ;;;Loop till all moved
     82 020246  105737  000000G                    TSTB    MEM256          ;;;Does machine have > 256KB?
     83 020252  001403                            BEQ     5$              ;;;Br if not
     84 020254  042737  000000G 000000G            BIC     #EMMAP,@#SR3MMR ;;;Disable extended memory addressing
     85 020262  042737  000000G 000000G 5$:       BIC     #MMENBL,@#SROMMR;;;Enable memory management
     86 020270  012637  000000G                    MOV     (SP)+,@#KPAR5   ;;;Replace PAR 5 mapping
     87 020274                                    ENABL                   ; ** Enable interrupts **
     88                                    ;
     89                                    ;  See if there is more to read
     90                                    ;
     91 020302  062737  000010  000126'            ADD     #8.,HMAP        ;Increase XM region base
     92 020310  005237  000142'                    INC     FILBLK          ;Increment file block number
     93 020314  005702                            TST     R2              ;Is there more to read?
     94 020316  001270                            BNE     1$              ;Loop if more to read
     95                                    ;
     96                                    ;  We have finished moving the handler into its XM region.
     97                                    ;  Set up addresses of system routines in a vector at the end of the handler
     98                                    ;
     99 020320  012703  000000G                    MOV     #VPAR5,R3       ;Get virtual address of handler base
    100 020324  066403  000000G                    ADD     HANSIZ(R4),R3   ;Get virtual address beyond end of handler
    101 020330  004737  021242'                    CALL    HANMAP          ;;;Map KPAR5 to the handler
    102 020334  004737  020400'                    CALL    STHNPV          ;;;Set up handler pointer vector
    103 020340  004737  021316'                    CALL    HANUMP          ;Restore mapping
    104                                    ;
    105                                    ;  If handler has any load-time code, run it now
    106                                    ;
    107 020344  010537  000134'                    MOV     R5,FMEMHI       ;Set addr of top of free memory area
    108 020350  004737  020542'                    CALL    DOHNLC          ;Run any load-time code for handler
    109 020354  013705  000134'                    MOV     FMEMHI,R5       ;Get new top of free memory address
    110                                    ;
    111                                    ;  Close the handler file
    112                                    ;
    113 020360                           9$:     .CLOSE  #1              ;Close the handler file
    114                                    ;
```

```
                                        ;  Finished
                                        ;
020366  012604                                MOV     (SP)+,R4
020370  012603                                MOV     (SP)+,R3
020372  012602                                MOV     (SP)+,R2
020374  012601                                MOV     (SP)+,R1
020376  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.
                                ;
020400  010346                  STHNPV: MOV     R3,-(SP)
                                ;
                                ;  Set up addresses in the pointer vector
                                ;
020402  012743  0000006G                MOV     #FORK,-(R3)     ;Address of fork routine
020406  012743  0000006G                MOV     #INTEN,-(R3)    ;Address of inten routine
020412  032737  0000006G 000240'        BIT     #SG$IOT,HGENFL  ;Does handler want timeout support?
020420  001402                          BEQ     2$              ;Br if not
020422  012743  0000006G                MOV     #IOTIMR,-(R3)   ;Set address of timeout support routine
020426  032737  0000006G 000240' 2$:    BIT     #SG$ELG,HGENFL  ;Does handler want error logging support?
020434  001402                          BEQ     3$              ;Br if not
020436  012743  0000006G                MOV     #ERRLOG,-(R3)   ;Set address of error logging routine
020442  012743  0000006G          3$:   MOV     #PTWRD,-(R3)
020446  012743  0000006G                MOV     #PTBYT,-(R3)
020452  012743  0000006G                MOV     #GTBYT,-(R3)
020456  012743  0000006G                MOV     #MPPHY,-(R3)
020462  012743  0000006G                MOV     #RELOC,-(R3)
                                ;
                                ;  Finished
                                ;
020466  012603                          MOV     (SP)+,R3
020470  000207                          RETURN
```

```
 1                                          ;
 2                                          ;  Error occured while loading device handler.
 3                                          ;  R0 = error message address;  CURNAM = device name.
 4                                          ;
 5 020472  010001                  HLERR:   MOV     R0,R1           ;SAVE ERROR MESSAGE ADDRESS
 6 020474  105737  0000006                  TSTB    VINABT          ;ABORT OR JUST PRINT MESSAGE?
 7 020500  001416                           BEQ     9$              ;BR IF NOT ABORT
 8 020502                                   .PRINT  #TSXHD          ;PRINT ERROR MESSAGE HEADING
 9 020510                                   .PRINT  R1              ;PRINT ERROR MESSAGE
10 020514  013700  000146'                  MOV     CURNAM,R0       ;GET RAD50 DEVICE NAME
11 020520  004737  025274'                  CALL    PRTR50          ;PRINT DEVICE NAME
12 020524                                   .PRINT  #CRLF
13 020532  000137  004134'                  JMP     INISTP          ;ABORT INITIALIZATION
14 020536  000261                  9$:      SEC                     ;MAKE SURE CARRY IS SET
15 020540  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.
                                ;
020542  010146                  DOHNLC: MOV     R1,-(SP)
020544  010246                          MOV     R2,-(SP)
020546  010346                          MOV     R3,-(SP)
020550  010446                          MOV     R4,-(SP)
020552  010546                          MOV     R5,-(SP)
                                ;
                                ;  Examine 1st word of handler to see if it could have any load-time code.
                                ;
020554  016405  000000G                 MOV     HANENT(R4),R5   ;Get address of handler entry point
020560  004737  021242'                 CALL    HANMAP          ;;;Map Kpar5 to handler if mapped handler
020564  016500  000004                  MOV     4(R5),R0        ;;;Get 1st instruction located at 4 in handler
020570  004737  021316'                 CALL    HANUMP          ;Restore normal mapping
020574  020027  000240                  CMP     R0,#240         ;Is it a NOP?
020600  103516                          BLO     7$              ;Br if can't be any load code
020602  020027  000277                  CMP     R0,#277         ;
020606  101113                          BHI     7$              ;Br if can't be any load code
020610  132700  000004                  BITB    #4,R0           ;Is there load code?
020614  001510                          BEQ     7$              ;Br if not
                                ;
                                ;  Handler may have load code.
                                ;  Read block 0 of handler and get offset to load code.
                                ;
020616  013702  000152'                 MOV     WRKBUF,R2       ;Get addr of our work buffer
020622                                  .READW  #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
020656  022227  031066                  CMP     (R2)+,#^RHAN    ;Is this a new type handler?
020662  001065                          BNE     7$              ;Br if not
020664  016203  000004                  MOV     4(R2),R3        ;Get offset to load code
020670  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.
                                ;
020672  020327  001000                  CMP     R3,#1000        ;Is load code in block 0 of handler?
020676  103424                          BLO     1$              ;Br if yes
020700  010302                          MOV     R3,R2           ;Get offset to start of load code
020702  072227  177767                  ASH     #-9.,R2         ;Convert to a block number
020706  042702  177400                  BIC     #^C377,R2       ;Clear all but block number
020712                                  .READW  #AREA,#1,WRKBUF,#512.,R2 ;Read 2 blocks from handler file
                                ;
                                ;  The load code is now in WRKBUF.  Set up and execute it.
                                ;
020750  010437  000144'         1$:     MOV     R4,CURDEV       ;Save current device index number
020754  042703  177000                  BIC     #^C777,R3       ;Get offset within block of load code entry pt
020760  010300                          MOV     R3,R0           ;Get entry point offset
020762  063700  000152'                 ADD     WRKBUF,R0       ;Add base address
```

```
     58 020766  013701  000000G                 MOV     RPRVEC,R1       ;Get pointer to GETVEC routine for Pro
     59 020772  012702  000000C                 MOV     #MAXDEV*2,R2    ;Get 2** entries in device tables
     60 020776  012703  000004                  MOV     #4,R3           ;Set code saying this is load code
     61 021002  012705  000000G                 MOV     #HANENT,R5      ;Point to handler entry address vector
     62 021006  060405                          ADD     R4,R5           ;Point to entry cell for this handler
     63 021010  004737  021242'                 CALL    HANMAP          ;;;Map Kpar5 to handler if it is a mapped
     64 021014  012704  021072'                 MOV     #LDREAD,R4      ;;;Get pointer to Read routine
     65 021020  004710                          CALL    (R0)            ;;;Execute the load code
     66 021022  103407                          BCS     2$              ;;;Br if handler load code signaled an error
     67                                 ;
     68                                 ;  Fetch/load code ran ok.
     69                                 ;  Turn off handler mapping.
     70                                 ;
     71 021024  012737  001400  000000G         MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
     72 021032  004737  021316'                 CALL    HANUMP          ;Unmap the handler
     73 021036  000241                  7$:     CLC                     ;Clear the carry flag for return
     74 021040  000406                          BR      9$
     75                                 ;
     76                                 ;  Error occurred in fetch/load code
     77                                 ;
     78 021042  012737  001400  000000G 2$:      MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
     79 021050  004737  021316'                 CALL    HANUMP          ;Unmap the handler
     80 021054  000261                          SEC                     ;Set the carry flag for return
     81                                 ;
     82                                 ;  Finished
     83                                 ;
     84 021056  012605                  9$:     MOV     (SP)+,R5
     85 021060  012604                          MOV     (SP)+,R4
     86 021062  012603                          MOV     (SP)+,R3
     87 021064  012602                          MOV     (SP)+,R2
     88 021066  012601                          MOV     (SP)+,R1
     89 021070  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
                                ;
021072  010046                  LDREAD: MOV     R0,-(SP)
021074  010446                          MOV     R4,-(SP)        ;;;
021076  010004                          MOV     R0,R4           ;;;Get starting block number
                                ;
                                ;  Save current mapping information
                                ;
021100  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256?
021104  001402                          BEQ     1$              ;;;Br if not
021106  013746  000000G                 MOV     @#SR3MMR,-(SP)  ;;;Save extended memory address register
021112  013746  000000G          1$:    MOV     @#SROMMR,-(SP)  ;;;Save memory mapping
021116  013746  000000G                 MOV     @#KPAR5,-(SP)   ;;;Save current KPAR5 mapping
021122  013746  000000G                 MOV     @#KPAR6,-(SP)   ;;;Save current KPAR6 mapping
021126  012737  001400  000000G         MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
                                ;
                                ;  Turn off handler mapping
                                ;
021134  004737  021316'                 CALL    HANUMP          ;Turn off handler mapping
                                ;
                                ;  Read the requested data from the handler
                                ;
021140                                  .READW  #AREA,#1,R2,R1,R4 ;Read the blocks
021172  103420                          BCS     9$              ;Br if read error
                                ;
                                ;  Restore handler mapping
                                ;
021174  013704  000144'                 MOV     CURDEV,R4       ;Get current device index
021200  004737  021242'                 CALL    HANMAP          ;;;Map Kpar5 if necessary
                                ;
                                ;  Restore mapping information
                                ;
021204  012637  000000G                 MOV     (SP)+,@#KPAR6   ;;;Restore KPAR6 mapping
021210  012637  000000G                 MOV     (SP)+,@#KPAR5   ;;;Restore KPAR5 mapping
021214  012637  000000G                 MOV     (SP)+,@#SROMMR  ;;;Restore memory mapping
021220  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256?
021224  001402                          BEQ     2$              ;;;Br if not
021226  012637  000000G                 MOV     (SP)+,@#SR3MMR  ;;;Restore extended memory address register
                                ;
                                ;  Finished
                                ;
021232  000241                  2$:     CLC                     ;;;Signal success on return
021234  012604                  9$:     MOV     (SP)+,R4
021236  012600                          MOV     (SP)+,R0
021240  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
                                ;
021242                          HANMAP:
                                ;
                                ;  Disable interrupts
                                ;
021242                                  DISABL                  ;;;Disable interrupts
                                ;
                                ;  Change RMON pointer to point to TSX-Plus vector
                                ;
021250  012737  000000G 000000G         MOV     #MONVEC,@#RMON  ;;;Say TSX-Plus is the monitor
                                ;
                                ;  See if this handler is mapped
                                ;
021256  005764  000000G                 TST     HANPAR(R4)      ;;;Is this handler mapped?
021262  001403                          BEQ     9$              ;;;Br if not
                                ;
                                ;  This handler is mapped.
                                ;  Set up mapping to access it.
                                ;
021264  016437  000000G 000000G         MOV     HANPAR(R4),@#KPAR5;;;Map KPAR5 to the handler code
021272  052737  000000G 000000G 9$:     BIS     #MMENBL,@#SROMMR;;;Enable memory mapping
021300  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
021304  001403                          BEQ     10$             ;;;Br if not
021306  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
                                ;
                                ;  Finished
                                ;
021314  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.
                                ;
021316                          HANUMP:
                                ;
                                ;  Turn off memory management
                                ;
021316  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
021322  001403                          BEQ     1$              ;;;Br if not
021324  042737  000000G 000000G         BIC     #EMMAP,@#SR3MMR ;;;Turn off extended memory addressing
021332  042737  000000G 000000G 1$:     BIC     #MMENBL,@#SROMMR;;;Turn off memory mapping
021340  012737  001200  000000G         MOV     #1200,@#KPAR5   ;;;Reset KPAR5 to its normal mapping
```

```
                                        ;
                                        ;  Restore RMON pointer to RT11
                                        ;
021346  013737  000046' 000000G               MOV     RTMNVC,@#RMON   ;;;Reset RMON pointer
                                        ;
                                        ;  Enable interrupts
                                        ;
021354                                        ENABL                   ;Enable interrupts
                                        ;
                                        ;  Finished
                                        ;
021362  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
021364  005001                  FNDHRB: CLR     R1              ;Point to next region control block (none)
021366  000401                          BR      1$              ;Go signal error and return
                                ;
021370  005002                  HANXMR: CLR     R2              ;Return largest possible region size (none)
021372  000261                  1$:     SEC                     ;Signal error on request
021374  000207                          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.
                                ;
                                FNDHRB: MOV     R2,-(SP)
                                ;
                                ;  Search for specified RCB and also remember if we see a free RCB
                                ;
                                        CLR     R2              ;Say no free RCB found
                                        MOV     HANRCB,R1       ;Point to start of RCB area
                                        TST     (R1)+           ;Skip over -1 word at front
                                1$:     TST     (R1)            ;What is the status of this RCB
                                        BNE     2$              ;Br if this is not a free RCB
                                        MOV     R1,R2           ;Remember address of a free RCB
                                        BR      3$
                                2$:     CMP     (R1),#-1        ;Are we at the end of the list?
                                        BEQ     4$              ;Br if yes
                                        CMP     (R5),6(R1)      ;Compare the names
                                        BNE     3$              ;Br if don't match
                                        CMP     2(R5),10(R1)    ;Compare 2nd half of name
                                        BEQ     6$              ;Br if found the RCB we were searching for
                                3$:     ADD     #12,R1          ;Point to the next RCB
                                        BR      1$              ;Continue searching
                                ;
                                ;  We could not find the specified RCB.
                                ;  If there was a free one, initialize the name.
                                ;
```

```
                                4$:     MOV     R2,R1           ;Was there a free RCB?
                                        BEQ     5$              ;Br if not
                                        MOV     (R5),6(R1)      ;Set name in the RCB
                                        MOV     2(R5),10(R1)    ;(2nd word of name)
                                        ADD     CURDEV,10(R1)   ;Make name unique to device
                                5$:     SEC                     ;Signal that we did not find the RCB
                                        BR      9$
                                ;
                                ;  We found the RCB
                                ;
                                6$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
                                9$:     MOV     (SP)+,R2
                                        RETURN
```

```
                        .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 the 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.
                ;
                HANXMR: MOV     R0,-(SP)
                ;
                ;  Get the total amount of free memory space available now
                ;  and see if the requested region can be allocated.
                ;
                        MOV     FMEMHI,R0       ;Top of free memory
                        SUB     FMEMLO,R0       ;-Base of free memory
                        CMP     R2,R0           ;Do we have room for the requested region?
                        BHI     8$              ;Br if not
                ;
                ;  There is room for the region so allocate it from the top of memory
                ;
                        SUB     R2,FMEMHI       ;Allocate the region
                        MOV     FMEMHI,R1       ;Return the address of the base of the region
                        CLC                     ;Signal success on return
                        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.
                ;
                8$:     MOV     R0,R2           ;Get the size of the largest possible region
                        SEC                     ;Signal failure on return
                ;
                ;  Finished
                ;
                9$:     MOV     (SP)+,R0
                        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$MAP flag is set in the DVFLAG word for devices that require mapping.
;
;  Inputs:
;   R5 = Pointer to low-memory free area
;
;  Outputs:
;   MIOFLAG = 0==>I/O mapping not required for any device;
;             1==>I/O mapping required for some device.
;   R5 = Pointer to new low-memory free area.
;
SETMIO: MOV     R1,-(SP)
;
;  Determine if this machine requires mapping at all
;
        TST     #MIODBG         ;Are we debugging mapped I/O system?
        BNE     2$              ;Br if yes
        BIT     #EXTLSI,ICONFG  ;Is this a Q-bus machine with more than 256Kb?
        BNE     2$              ;Br if yes
;
;  This is not a Q-bus system with more than 256Kb.
;  Mapping is not required at all.
;
        MOV     #2,R1           ;Get initial device index number
1$:     BIC     #DX$MAP,DVFLAG(R1) ;Clear mapped-I/O flag
        ADD     #2,R1           ;Get next device index
        CMP     R1,NUMDEV       ;More to do?
        BLOS    1$              ;Br if yes
        BR      9$
;
;  This is a Q-bus system with more than 256Kb.
;  See if any devices have requested mapped I/O.
;
2$:     CLR     R0              ;Clear composite flag word
        MOV     #2,R1           ;Initialize device index number
3$:     BIS     DVFLAG(R1),R0   ;Combine flags from all devices
        ADD     #2,R1           ;Get next device index number
        CMP     R1,NUMDEV       ;Checked all devices?
        BLOS    3$              ;Br if not
        BIT     #DX$MAP,R0      ;Does any device require mapping?
        BEQ     9$              ;Br if not
;
;  I/O mapping is required
;
        INCB    MIOFLG          ;Remember that mapping is required
;
;  Zero the area where we will build the control structures
;
        MOVB    VMIOBF,R1       ;Get # buffers wanted
        MUL     #MI$$SZ,R1      ;Times size for each control block
        ADD     #MIONWB*MW$$SZ,R1;Add space for MIO wait blocks
```

```
                                        ASR     R1              ;Get # words to zero
                                        MOV     R5,R0           ;Get pointer to start of area
                                4$:     CLR     (R0)+           ;Zero the entire area
                                        SOB     R1,4$
                                ;
                                ;  Allocate and initialize the control structures
                                ;
                                        MOV     R5,MIOBHD       ;Start of area for I/O mapping control blks
                                        MOVB    VMIOBF,R0       ;Get # buffers wanted
                                5$:     MOV     R5,R1           ;Get pointer to current control block
                                        ADD     #MI$$SZ,R1      ;Get pointer to next control block
                                        DEC     R0              ;Need to link more together?
                                        BLE     6$              ;Br if not
                                        MOV     R1,MI$LNK(R5)   ;Set pointer to next control block
                                        MOV     R1,R5           ;Advance pointer to next block
                                        BR      5$              ;See if more to do
                                6$:     MOV     R1,R5           ;Set pointer past last block
                                        MOV     R5,MIOWHD       ;Start of wait blocks
                                        MOV     #MIONWB-1,R0    ;Get # wait blocks wanted - 1
                                7$:     MOV     R5,R1           ;Get pointer to current block
                                        ADD     #MW$$SZ,R1      ;Get pointer to next block
                                        MOV     R1,MW$LNK(R5)   ;Set pointer to next wait block
                                        MOV     R1,R5           ;Advance pointer to next block
                                        SOB     R0,7$           ;Loop if more to allocate
                                        ADD     #MW$$SZ,R5      ;Allocate space for last block (with 0 link)
                                ;
                                ;  Finished
                                ;
                                9$:     MOV     (SP)+,R1
                                        RETURN
                                        .IFF    ;NE,<PROASM-1>
                                ;
                                ;  Define dummy routine for Pro
                                ;
 021376  000207                 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$FLG(seg) = 0==>Load seg into high memory; 1==>Load over TSINIT.
                                ;   OVLBAS = Address of location within TSINIT where overlays start.
                                ;   R5 = Pointer past last overlay loaded over TSINIT.
                                ;
021400  010146                 OVLPOS: MOV     R1,-(SP)
021402  010246                         MOV     R2,-(SP)
021404  010346                         MOV     R3,-(SP)
021406  010446                         MOV     R4,-(SP)
021410  010504                         MOV     R5,R4           ;Get address where we may load overlays
                                ;
                                ;  Build the table that holds information about the overlays
                                ;
021412  004737  021610'                 CALL    OVLBLD          ;Build overlay information table
                                ;
                                ;  First determine how much space will be used by those overlays that are
                                ;  forced to be loaded over TSINIT.
                                ;
021416  062704  000077                  ADD     #63.,R4         ;Bound address to 64-byte boundary
021422  042704  000077                  BIC     #77,R4
021426  010437  000140'                 MOV     R4,OVLBAS       ;Remember address where we load overlays
021432  012702  000516'                 MOV     #OSTABL,R2      ;Point to start of table
021436  005762  000000          1$:     TST     OS$SIZ(R2)      ;Is this overlay to be loaded?
021442  001423                          BEQ     2$              ;Br if not
021444  016200  000004                  MOV     OS$OVL(R2),R0   ;Point to linker-built entry
021450  016000  000000G                 MOV     0.ADR(R0),R0    ;Get Rad50 segment ID
021454  012701  000746'                 MOV     #LOWOVL,R1      ;Point to table of overlays to go over TSINIT
021460  020021                  6$:     CMP     R0,(R1)+        ;Must this overlay go over TSINIT?
021462  001404                          BEQ     4$              ;Br if yes
021464  020127  000754'                 CMP     R1,#LOWEND      ;End of low-overlay table?
021470  103773                          BLO     6$              ;Br if not
021472  000407                          BR      2$              ;This overlay is not forced over TSINIT
021474  005262  000002          4$:     INC     OS$FLG(R2)      ;Set flag saying load over TSINIT
021500  016200  000000                  MOV     OS$SIZ(R2),R0   ;Get # 64-byte blocks needed for overlay
021504  072027  000006                  ASH     #6,R0           ;Get # bytes needed for overlay
021510  060004                          ADD     R0,R4           ;Advance address within TSINIT
021512  062702  000006          2$:     ADD     #OS$$SZ,R2      ;Point to entry for next segment
021516  020237  000744'                 CMP     R2,OSLAST       ;Have we finished?
021522  103745                          BLO     1$              ;Loop if not
                                ;
                                ;  Determine how much memory space is available in TSINIT for other overlays
                                ;
021524  020427  025266'                 CMP     R4,#INITOP-50.  ;Any space left for other overlays?
021530  103021                          BHIS    9$              ;Br if not
021532  012705  025350'                 MOV     #INITOP,R5      ;Point to top of overlay area
021536  160405                          SUB     R4,R5           ;Total space available for overlays
021540  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.
                                        ;
021544  004737  023020'         3$:     CALL    OVLTRY          ;Try to find largest overlay that will fit
021550  103411                          BCS     9$              ;Br if no more overlays will fit
021552  005262  000002                  INC     OS$FLG(R2)      ;Remember to load over TSINIT
021556  016200  000000                  MOV     OS$SIZ(R2),R0   ;Get # 64-byte blocks needed for overlay
021562  160005                          SUB     R0,R5           ;Reduce remaining free space in TSINIT
021564  072027  000006                  ASH     #6,R0           ;Get # bytes needed for overlay
021570  060004                          ADD     R0,R4           ;Advance overlay address in TSINIT
021572  000764                          BR      3$              ;See if we can find more segments to load
                                        ;
                                        ;  Finished
                                        ;
021574  010405                  9$:     MOV     R4,R5           ;Return top-of-overlay address in R5
021576  012604                          MOV     (SP)+,R4
021600  012603                          MOV     (SP)+,R3
021602  012602                          MOV     (SP)+,R2
021604  012601                          MOV     (SP)+,R1
021606  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.
                                ;
021610  010146                  OVLBLD: MOV     R1,-(SP)
021612  010246                          MOV     R2,-(SP)
021614  010346                          MOV     R3,-(SP)
                                ;
                                ;  Read 1st block of SAV file to get pointer to overlay table
                                ;
021616  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
021622                                  .READW  #AREA,#17,R2,#256.,#0 ;read first block of the save file
021656  103444                          BCS     22$             ;Br if error on read
021660  016201  000064                  MOV     64(R2),R1       ;point to the overlay table
021664  001012                          BNE     15$             ;br if overlays exist
                                ;
                                ;  Must be verion 3B overlays structure at absolute location.
                                ;
021666  012737  000137  001000          MOV     #137,@#1000     ;position jump instrucion over 3b ovly handler
021674  012737  000000G 001002          MOV     #$OVRH,@#1002   ;position overlay intercept location
021702  012701  001104                  MOV     #1104,R1        ;point to the overlay table
021706  010137  000000G                 MOV     R1,OVRADD       ;save the address of the overlay table
                                ;
                                ;  Initialize the table that holds information about the overlays
                                ;
021712  012703  000516'         15$:    MOV     #OSTABL,R3      ;Point to table for overlay info
021716  010163  000004          11$:    MOV     R1,OS$OVL(R3)   ;Save pointer to overlay control block
021722  005063  000002                  CLR     OS$FLG(R3)      ;Assume seg will be loaded in high memory
021726  004737  022130'                 CALL    ALCOVL          ;Determine if we should load this overlay
021732  010263  000000                  MOV     R2,OS$SIZ(R3)   ;Remember total size of overlay+data
021736  062703  000006                  ADD     #OS$$SZ,R3      ;Point to next overlay table entry
021742  062701  000006          12$:    ADD     #6,R1           ;find the next region
021746  021127  004537                  CMP     (R1),#4537      ;compare with a <JSR R5,$OVRH> instruction
021752  001361                          BNE     11$             ;Br if not at end
021754  010337  000744'                 MOV     R3,OSLAST       ;Save pointer past last overlay table entry
                                ;
                                ;  Finished
                                ;
021760  012603                          MOV     (SP)+,R3
021762  012602                          MOV     (SP)+,R2
021764  012601                          MOV     (SP)+,R1
021766  000207                          RETURN
                                ;
                                ;  Error -- Read error occured while reading overlay table
                                ;
021770                          22$:    .PRINT  #TSXHD
021776                                  .PRINT  #RDERR
022004  000137  004134'                 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.
                                ;
022010  010146                  GETMAP: MOV     R1,-(SP)
022012  010246                          MOV     R2,-(SP)
022014  010346                          MOV     R3,-(SP)
022016  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.
                                ;
022022  012703  000516'                  MOV     #OSTABL,R3      ;Point to 1st overlay table entry
022026  004737  022274'          1$:     CALL    OPTOVL          ;See if this segment should be loaded
022032  010263  000000                   MOV     R2,OS$SIZ(R3)   ;Save # 64-byte blocks needed for overlay
022036  062703  000006                   ADD     #OS$$SZ,R3      ;Point to next overlay table entry
022042  020337  000744'                   CMP     R3,OSLAST       ;Checked all entries in overlay table?
022046  103767                           BLO     1$              ;Br if not
                                ;
                                ;  Load those overlays that go into high memory
                                ;
022050  012702  000516'                  MOV     #OSTABL,R2      ;Point to 1st overlay entry
022054  005762  000000          3$:     TST     OS$SIZ(R2)      ;Is this overlay segment wanted?
022060  001405                           BEQ     4$              ;Br if not
022062  005762  000002                   TST     OS$FLG(R2)      ;Load over TSINIT or into high memory?
022066  001002                           BNE     4$              ;Br if load over TSINIT
022070  004737  023116'                   CALL    GETOVL          ;Load overlay into high memory
022074  062702  000006          4$:     ADD     #OS$$SZ,R2      ;Point to next overlay table entry
022100  020237  000744'                   CMP     R2,OSLAST       ;Have we done all overlays?
022104  103763                           BLO     3$              ;Loop if not
                                ;
                                ;  Finished
                                ;
022106  013700  000000G         19$:    MOV     SMRSIZ,R0       ;Get memory pointer at start of allocation
022112  160500                           SUB     R5,R0           ;Calc amt of space allocated
022114  010037  000000G                   MOV     R0,SMRSIZ       ;Save total space used for mapped regions
022120  012603                           MOV     (SP)+,R3
022122  012602                           MOV     (SP)+,R2
022124  012601                           MOV     (SP)+,R1
022126  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$xxx)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> This segment is to be loaded.
                                ;   C-flag set     ==> Do not load this overlay segment.
                                ;   R2 = # 64-Byte blocks needed for segment including data areas within it.
                                ;
022130  010146                  ALCOVL: MOV     R1,-(SP)
                                ;
                                ;  Get pointer to linker-build overlay entry for segment
                                ;
                                        MOV     OS$OVL(R3),R1   ;Get pointer to linker-built entry for seg
                                ;
                                ;  Read in the first block of the overlay segment
                                ;
022136  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
022142                                  .READW  #AREA,#17,R2,#256.,0.BLK(R1) ;read the first block
022200  103415                          BCS     3$              ;Br if read error
                                ;
                                ;  Save the 3 character Rad50 segment ID in the 0.ADR cell of the
                                ;  linker-built overlay table entry for this segment.
                                ;
030202  016261  000002  000000G         MOV     2(R2),0.ADR(R1) ;save the rad50 overlay identifier
                                ;
                                ;  Make sure the segment is not larger than 8Kb
                                ;
022210  016102  000000G                 MOV     0.SIZ(R1),R2    ;get the word count of the code region
022214  006302                          ASL     R2              ;convert to byte count
022216  020227  020000                  CMP     R2,#20000       ;check for 8kb overflow
022222  101014                          BHI     21$             ;Br if region is too big
                                ;
                                ;  Don't load some optional segments if features were not selected
                                ;  in TSGEN.
                                ;
022224  004737  022274'                 CALL    OPTOVL          ;See if we want to load this segment
                                ;
                                ;  Finished
                                ;  The C-flag is set or reset by OPTOVL.
                                ;
022230  012601                          MOV     (SP)+,R1
022232  000207                          RETURN
                                ;
                                ;  Error -- Error on reading from SAV file
                                ;
022234                          3$:     .PRINT  #TSXHD          ;Print heading
022242                                  .PRINT  #RDERR          ;Read error
022250  000137  004134'                 JMP     INISTP          ;Abort initialization
                                ;
                                ;  Error -- Insufficient memory space to load run-time systems
                                ;
```

```
     58 022254                          21$:    .PRINT  #TSXHD          ;PRINT HEADING
     59 022262                                  .PRINT  #TSXSIZ         ;PRINT ERROR MESSAGE
     60 022270  000137  004134'                 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$xxx)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Load this overlay.
                                ;   C-flag set     ==> Do not load this overlay.
                                ;   R2 = # 64-byte blocks needed for code + data for the segment.
                                ;
022274  010346                  OPTOVL: MOV     R3,-(SP)
022276  010446                          MOV     R4,-(SP)
022300  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get the name of the overlay segment
                                ;
022302  016305  000004                  MOV     OS$OVL(R3),R5   ;Get pointer to linker-built entry
022306  016504  000000G                 MOV     O.ADR(R5),R4    ;Get name of the segment
                                ;
                                ;  Get size of code portion of overlay segment
                                ;
022312  016502  000000G                 MOV     O.SIZ(R5),R2    ;Get # words needed by code portion of seg
022316  006302                          ASL     R2              ;Convert to # bytes
                                ;
                                ;  See if this is an optional segment that we need to deal with specially
                                ;
022320  012700  022346'                 MOV     #OVLLST,R0      ;Point to overlay name list
022324  020420                  1$:     CMP     R4,(R0)+        ;Found name of overlay?
022326  001406                          BEQ     2$              ;Br if yes
022330  005720                          TST     (R0)+           ;No -- Skip over address word
022332  020027  022436'                 CMP     R0,#OVLEND      ;Checked all names in the list?
022336  103772                          BLO     1$              ;Loop if not
022340  000137  022756'                 JMP     OOXYES          ;Load this overlay
                                ;
                                ;  Branch off to processing routine
                                ;
022344  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
                                ;
022346                          OVLLST:
022346                                  OVLTBL  USR             ;TSUSR  -- File management
022352                                  OVLTBL  SPL             ;TSSPOL -- Spooling system
022356                                  OVLTBL  SP2             ;TSSPL2 -- Spooler flag pages
022362                                  OVLTBL  LOK             ;TSLOCK -- Shared file record locking
022366                                  OVLTBL  MSG             ;TSMSG  -- Inter-job message communication
022372                                  OVLTBL  SWP             ;TSSWAP -- Job swapper
022376                                  OVLTBL  PLS             ;TSPLAS -- PLAS support
```

```
                                                OVLTBL  SLE             ;TSSLE  -- Single line editor
                                                OVLTBL  WIN             ;TSWIN  -- Display window management
                                                OVLTBL  MIO             ;TSMIO  -- Mapped I/O
                                                OVLTBL  CLO             ;TSCLO  -- CL handler
                                                OVLTBL  DBG             ;TSDBUG -- Program debugger
                                                OVLTBL  CSH             ;TSCASH -- Data caching
                                                OVLTBL  DMP             ;TSDUMP -- Crash dump generator
                                        OVLEND:
                                        ;
                                        ;  File management
                                        ;
022436  013703  000000G                 OORUSR: MOV     VNFCSH,R3       ;Get # file cache entries
022442  070327  000000G                         MUL     #FC$$SZ,R3      ;Multiply by size of each entry
022446  060302                                 ADD     R3,R2           ;Allocate space for directory cache
022450  000542                                 BR      OOXYES          ;Load the segment
                                        ;
                                        ;  Spooling system
                                        ;
022452  005727  000000G                 OORSPL: TST     #SPLND          ;Are there any spooled devices?
022456  001534                                 BEQ     OOXNO           ;Br if not
022460  062702  000000C                         ADD     #<SPLNB*512.>,R2;Reserve room for spool buffers
022464  013703  000000G                         MOV     NSPLBL,R3       ;Get # blocks for spool file
022470  062703  000007                         ADD     #7,R3           ;Bound up to byte boundary
022474  072327  177775                         ASH     #-3,R3          ;Divide by 8 to get # bytes for table
022500  005203                                 INC     R3              ;Round up to word boundary
022502  042703  000001                         BIC     #1,R3
022506  060302                                 ADD     R3,R2           ;Add space for spool file allocation table
022510  000522                                 BR      OOXYES          ;Load the segment
                                        ;
                                        ;  Spooler flag pages?
                                        ;
022512  005727  000000G                 OORSP2: TST     #SPLND          ;Are there any spooled devices?
022516  001514                                 BEQ     OOXNO           ;If not, don't load overlay
022520  000516                                 BR      OOXYES          ;Load the segment
                                        ;
                                        ;  Record locking system
                                        ;
022522  005737  000000G                 OORLOK: TST     VMXSF           ;Any shared files?
022526  001510                                 BEQ     OOXNO           ;Br if not
022530  005737  000000G                         TST     VNUMDC          ;Shared file data caching wanted?
022534  001110                                 BNE     OOXYES          ;Br if yes
022536  162702  000000G                         SUB     #DCCSIZ,R2      ;Reduce size of segment - Leave out cache code
022542  000505                                 BR      OOXYES          ;Load the segment
                                        ;
                                        ;  Message communication system
                                        ;
022544  013703  000000G                 OORMSG: MOV     VMAXMC,R3       ;Is message communication facility wanted?
022550  001477                                 BEQ     OOXNO           ;Br if not
022552  070327  000000G                         MUL     #MB$$SZ,R3      ;Space for message channel blocks
022556  060302                                 ADD     R3,R2
022560  013703  000000G                         MOV     VMXMRB,R3       ;Number of message request blocks
022564  070327  000000G                         MUL     #MR$$SZ,R3      ;Times size of request block
022570  060302                                 ADD     R3,R2
022572  013703  000000G                         MOV     VMSCHR,R3       ;Max # chars in a message
022576  005203                                 INC     R3              ;Bound up to word
022600  042703  000001                         BIC     #1,R3           ;Reserve whole number of words
022604  062703  000000G                         ADD     #MU$TXT,R3      ;Plus space for message header
```

```
    115 022610  070337  000000G                 MUL     VMXMSG,R3       ;Times maximum number of messages
    116 022614  060302                          ADD     R3,R2           ;Space for message buffers
    117 022616  000457                          BR      OOXYES
    118                                 ;
    119                                 ;  PLAS support
    120                                 ;
    121 022620  013703  000000G         OORPLS: MOV     VPLAS,R3        ;PLAS support wanted?
    122 022624  001451                          BEQ     OOXNO           ;Br if not
    123 022626  062703  000021                  ADD     #17.,R3         ;Bound up # blocks
    124 022632  072327  177775                  ASH     #-3,R3          ;Get # bytes needed for swap file bit map
    125 022636  060302                          ADD     R3,R2           ;Reserve room for swap file bit map
    126 022640  000446                          BR      OOXYES          ;Load the segment
    127                                 ;
    128                                 ;  Job swapper
    129                                 ;
    130 022642  105737  000000G         OORSWP: TSTB    VSWPFL          ;Is this a swapping system?
    131 022646  001440                          BEQ     OOXNO           ;Br if not
    132 022650  000442                          BR      OOXYES          ;Br if yes -- Load the segment
    133                                 ;
    134                                 ;  Single line editor
    135                                 ;
    136 022652  105737  000000G         OORSLE: TSTB    VSLEDT          ;Is SL editor wanted?
    137 022656  001434                          BEQ     OOXNO           ;Br if not
    138 022660  000436                          BR      OOXYES          ;Load the segment
    139                                 ;
    140                                 ;  Display windows
    141                                 ;
    142 022662  013703  000000G         OORWIN: MOV     VMXWIN,R3       ;Are any display windows wanted?
    143 022666  001430                          BEQ     OOXNO           ;Br if not
    144 022670  070327  000000G                 MUL     #DW$$SZ,R3      ;Amt of space needed for window control blks
    145 022674  060302                          ADD     R3,R2           ;Add to size of overlay
    146 022676  000427                          BR      OOXYES          ;Load the segment
    147                                 ;
    148                                 ;  Mapped I/O
    149                                 ;
    150 022700  105737  000000G         OORMIO: TSTB    MIOFLG          ;Is I/O mapping needed?
    151 022704  001421                          BEQ     OOXNO           ;Br if not
    152 022706  000423                          BR      OOXYES          ;Load the segment
    153                                 ;
    154                                 ;  CL handler
    155                                 ;
    156 022710  005727  000000G         OORCLO: TST     #CLTOTL         ;Any I/O lines?
    157 022714  001415                          BEQ     OOXNO           ;Br if not
    158 022716  000417                          BR      OOXYES          ;Yes, load the segment
    159                                 ;
    160                                 ;  Program debugger
    161                                 ;
    162 022720  105737  000000G         OORDBG: TSTB    VDBFLG          ;Is the program debugger wanted?
    163 022724  001411                          BEQ     OOXNO           ;Br if not
    164 022726  000413                          BR      OOXYES          ;Load this segment
    165                                 ;
    166                                 ;  Data caching
    167                                 ;
    168 022730  005737  000000G         OORCSH: TST     CSHALC          ;Is data caching wanted?
    169 022734  001405                          BEQ     OOXNO           ;Br if not
    170 022736  000407                          BR      OOXYES          ;Load this segment
    171                                 ;
```

```
                                        ;  Crash dump generator
                                        ;
000000G 105737  VSYDMP  OORDMP: TSTB    VSYDMP          ;Is dump facility wanted?
001401                          BEQ     OOXNO           ;Br if not
000403                          BR      OOXYES          ;Br if yes
                                        ;
                                        ;  Don't load this segment
                                        ;
005002                  OOXNO:  CLR     R2              ;Say no space needed for overlay
000261                          SEC                     ;Signal don't load the segment
000415                          BR      OOXFIN
                                        ;
                                        ;  Load this segment
                                        ;
005202                  OOXYES: INC     R2              ;Make sure size is even
042702  000001                  BIC     #1,R2
020227  020000                  CMP     R2,#8192.       ;Don't allow code + data to exceed 8Kb
101402                          BLOS    1$              ;Br if ok
012702  020000                  MOV     #8192.,R2       ;Note, init code in segment will truncate dat
062702  000077          1$:     ADD     #63.,R2         ;Convert to # 64-byte blocks
072227  177772                  ASH     #-6,R2
000241                          CLC                     ;Signal to load the segment
                                        ;
                                        ;  Finished
                                        ;
012605                  OOXFIN: MOV     (SP)+,R5
012604                          MOV     (SP)+,R4
012603                          MOV     (SP)+,R3
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.
                                ;
023020  010346                  OVLTRY: MOV     R3,-(SP)
                                ;
                                ;  Begin loop to examine all segments
                                ;
023022  005002                          CLR     R2              ;Say we haven't found any segment yet
023024  012703  000516'                 MOV     #OSTABL,R3      ;Point to entry for 1st segment
023030  005763  000000          1$:     TST     OS$SIZ(R3)      ;Is this segment to be loaded?
023034  001415                          BEQ     2$              ;Br if not
023036  005763  000002                  TST     OS$FLG(R3)      ;Is this segment already over TSINIT?
023042  001012                          BNE     2$              ;Br if yes
023044  026305  000000                  CMP     OS$SIZ(R3),R5   ;Will this segment fit?
023050  101007                          BHI     2$              ;Br if not
023052  005702                          TST     R2              ;Have we found any other seg yet?
023054  001404                          BEQ     3$              ;Br if not
023056  026362  000000  000000          CMP     OS$SIZ(R3),OS$SIZ(R2) ;Is new seg larger than old?
023064  101401                          BLOS    2$              ;Br if not
023066  010302                  3$:     MOV     R3,R2           ;Remember largest segment
023070  062703  000006          2$:     ADD     #OS$$SZ,R3      ;Point to entry for next segment
023074  020337  000744'                 CMP     R3,OSLAST       ;Have we checked all segments?
023100  103753                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
023102  000241                          CLC                     ;Assume we found a segment
023104  005702                          TST     R2              ;Did we find a segment that will fit?
023106  001001                          BNE     9$              ;Br if yes
023110  000261                          SEC                     ;Signal failure on return
023112  012603                  9$:     MOV     (SP)+,R3
023114  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.
                                ;
023116                          GETOVL:
                                ;
                                ;  Allocate space for the overlay segment
                                ;
023116  166205  000000                  SUB     OS$SIZ(R2),R5   ;Allocate space for overlay
023122  020527  001600                  CMP     R5,#1600        ;Are we about to run over RT-11?
023126  103405                          BLO     10$             ;Br if yes -- Insufficient memory
                                ;
                                ;  Remember the base address of some key segments
                                ;
023130  004737  004522'                 CALL    KEYSEG          ;Remember address of some segments
                                ;
                                ;  Load the segment
                                ;
023134  004737  023162'                 CALL    LODOVL          ;Load the segment
                                ;
                                ;  Finished
                                ;
023140  000207                          RETURN
                                ;
                                ;  Error: Memory overflow
                                ;
023142                          10$:    .PRINT  #TSXHD
023150                                  .PRINT  #TSXSIZ
023156  000137  004134'                 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.
                                ;
023162  010146                  LODOVL: MOV     R1,-(SP)
023164  010246                          MOV     R2,-(SP)
023166  010346                          MOV     R3,-(SP)
023170  010446                          MOV     R4,-(SP)
023172  010546                          MOV     R5,-(SP)

                                ;
                                ;  Get info about size of the overlay and position within SAV file
                                ;
023174  016201  000004                  MOV     OS$OVL(R2),R1   ;Get pointer to linker-built segment entry
023200  016103  000000G                 MOV     O.SIZ(R1),R3    ;Get size of overlay segment (# words)
023204  016137  000000G 000142'         MOV     O.BLK(R1),FILBLK;Get block in SAV file where segment starts
023212  010302                          MOV     R3,R2           ;Get total number of words in segment
023214  062702  000377                  ADD     #255.,R2        ;round to the nearest number of blocks
023220  000302                          SWAB    R2              ;Divide by 256. words per segment
023222  042702  177400                  BIC     #177400,R2      ;kill sign extension bits
023226  010561  000000G                 MOV     R5,O.PAR(R1)    ;Remember where segment is being loaded
                                ;
                                ;  Read next block of overlay segment into low-memory buffer
                                ;
023232  013704  000152'          10$:   MOV     WRKBUF,R4       ;Point to work buffer
023236                                  .READW  #AREA,#17,R4,#256.,FILBLK       ;read a block
023274  103466                          BCS     22$             ;read error occured
                                ;
                                ;  Move from low buffer to high position in memory
                                ;
023276  012701  000000G                 MOV     #VPAR5,R1       ;get the virtual address of the mapped region
023302  012700  000400                  MOV     #256.,R0        ;obtain the number of words to move
023306  020300                          CMP     R3,R0           ;Do we need to move as many as 256 words?
023310  103001                          BHIS    2$              ;Br if yes
023312  010300                          MOV     R3,R0           ;Get number of words to move for last block
023314  160003                   2$:    SUB     R0,R3           ;Get number of words left after this move
023316                                  DISABL                  ;** Disable interrupts **
023324  013746  000000G                 MOV     @#KPAR5,-(SP)   ;save the contents of the mapping register
023330  010537  000000G                 MOV     R5,@#KPAR5      ;change the mapping register
023334  052737  000000G 000000G         BIS     #MMENBL,@#SROMMR;enable memory management
023342  105737  000000G                 TSTB    MEM256          ;Does machine have at least 256Kb of memory?
023346  001403                          BEQ     11$             ;Br if not
023350  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;enable extended memory addressing
023356  012421                   11$:   MOV     (R4)+,(R1)+     ;move into high memory
023360  077002                          SOB     R0,11$
023362  105737  000000G                 TSTB    MEM256          ;Does this machine have at least 256Kb?
023366  001403                          BEQ     12$             ;Br if not
023370  042737  000000G 000000G         BIC     #EMMAP,@#SR3MMR ;disable extended memory management
023376  042737  000000G 000000G  12$:   BIC     #MMENBL,@#SROMMR;disable memory management
023404  012637  000000G                 MOV     (SP)+,@#KPAR5   ;restore the mapping register
023410                                  ENABL                   ;** Enable interrupts **
023416  062705  000010                  ADD     #10,R5          ;advance 64-byte block # by 512-bytes
023422  005237  000142'                 INC     FILBLK          ;increment file block #
```

```
     58 023426  005302                          DEC     R2              ;More to be copied?
     59 023430  001402                          BEQ     5$              ;Br if not
     60 023432  000137  023232'                 JMP     10$             ;Read and copy rest of mapped segment
     61                                ;
     62                                ;  Finished loading the segment
     63                                ;
     64 023436  012605                 5$:     MOV     (SP)+,R5
     65 023440  012604                         MOV     (SP)+,R4
     66 023442  012603                         MOV     (SP)+,R3
     67 023444  012602                         MOV     (SP)+,R2
     68 023446  012601                         MOV     (SP)+,R1
     69 023450  000207                         RETURN
     70                                ;
     71                                ;  Error occurred on read
     72                                ;
     73 023452                         22$:    .PRINT  #TSXHD          ;Print heading
     74 023460                                 .PRINT  #RDERR          ;Read error
     75 023466  000137  004134'                 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
                                ;
023472  010146                  GETSRT: MOV     R1,-(SP)
023474  010246                          MOV     R2,-(SP)
023476  010346                          MOV     R3,-(SP)
023500  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if this is a dummy run-time entry to allow for patching
                                ;
023502  021127  000000G                 CMP     (R1),#DMYDEV    ;Dummy run-time entry?
023506  001540                          BEQ     7$              ;Br if yes
                                ;
                                ;  Try to open a channel to run-time file
                                ;
023510                                  .LOOKUP #AREA,#1,R1     ;OPEN CHANNEL TO RUN-TIME FILE
023526  103010                          BCC     8$              ;BR IF OPEN WAS SUCCESSFUL
                                ;
                                ;  Cannot open shared run-time file.
                                ;  See if he wants to abort or continue.
                                ;
023530  105737  000000G                 TSTB    VINABT          ;ABORT OR CONTINUE
023534  001132                          BNE     9$              ;BR IF ABORT WANTED
023536  005061  000000G                 CLR     RT$NAM(R1)      ;Mark run-time as not-available
023542  005061  000002G                 CLR     RT$NAM+2(R1)
023546  000520                          BR      7$              ;GO LOAD NEXT RUN-TIME SYSTEM
                                ;
                                ;  Set up information about position of run-time in physical memory
                                ;
023550  116102  000000G         8$:     MOVB    RT$SKP(R1),R2   ;GET # BLOCKS TO SKIP AT FRONT OF RUN-TIME
023554  042702  177400                  BIC     #^C377,R2       ;CLEAR SIGN EXTENSION
023560  160200                          SUB     R2,R0           ;GET # BLOCKS TO READ (LOOKUP SET R0 W  SIZE)
023562  010561  000000G                 MOV     R5,RT$TOP(R1)   ;SET 64-BYTE BLOCK # ABOVE TOP OF RUN-TIME
023566  010003                          MOV     R0,R3           ;GET # 512-BYTE BLOCKS IN RUN-TIME
023570  072027  000003                  ASH     #3,R0           ;CONVERT TO # 64-BYTE BLOCKS
023574  160005                          SUB     R0,R5           ;CALCULATE BASE 64-BYTE BLOCK # OF RUN-TIME
023576  020527  001600                  CMP     R5,#1600        ;ARE WE ABOUT TO RUN OVER RT-11?
023602  103530                          BLO     11$             ;BR IF YES
023604  010561  000000G                 MOV     R5,RT$BAS(R1)   ;SET BASE 64-BYTE BLOCK # OF RUN-TIME
                                ;
                                ;  Read run-time system into memory and position in high-memory
                                ;
023610  010546                          MOV     R5,-(SP)        ;Save address of bottom of run-time
023612  013704  000152'         4$:     MOV     WRKBUF,R4       ;Point to work buffer
023616                                  .READW  #AREA,#1,R4,#256.,R2 ;READ A BLOCK OF RUN-TIME FILE
                                ;  Use memory management to access high-memory area.
023652  012701  000000G                 MOV     #VPAR6,R1       ;GET VIRTUAL ADDRESS OF PAR6 ADDRESS REGION
023656  010537  000000G                 MOV     R5,@#UPAR6      ;SET USER-MODE PAR6 MAP OFFSET VALUE
```

```
     58 023662  012737  077406  000000G          MOV     #077406,@#UPDR6 ;SET PDR TO ALLOW FULL ACCESS TO PAGE
     59 023670  052737  000000G 000000G          BIS     #UPMODE,@#PSW   ;SET PREVIOUS-MODE = USER FOR MTPD ACCESS
     60 023676  012700  000400                   MOV     #256.,R0        ;GET # WORDS TO MOVE
     61 023702                                   DISABL                  ;** Disable interrupts **
     62 023710  052737  000000G 000000G          BIS     #MMENBL,@#SROMMR;enable memory management
     63 023716  105737  000000G                  TSTB    MEM256          ;DOES THIS MACHINE HAVE AT LEAST 256KB?
     64 023722  001403                           BEQ     3$              ;BR IF NOT
     65 023724  052737  000000G 000000G          BIS     #EMMAP,@#SR3MMR ;enable extended memory addressing
     66 023732  012446                  3$:     MOV     (R4)+,-(SP)     ;TRANSFER DATA FROM BUFFER TO HIGH MEMORY
     67 023734  106621                           MTPD    (R1)+
     68 023736  077003                           SOB     R0,3$
     69 023740  105737  000000G                  TSTB    MEM256          ;DOES THIS MACHINE HAVE AT LEAST 256KB?
     70 023744  001403                           BEQ     31$             ;BR IF NOT
     71 023746  042737  000000G 000000G          BIC     #EMMAP,@#SR3MMR ;DISABLE EXTENDED MEMORY MANAGEMENT
     72 023754  042737  000000G 000000G 31$:     BIC     #MMENBL,@#SROMMR;DISABLE MEMORY MANAGEMENT
     73 023762                                   ENABL                   ;** Enable interrupts **
     74 023770  062705  000010                   ADD     #10,R5          ;ADVANCE 64-BYTE BLOCK # BY 512-BYTES
     75 023774  005202                           INC     R2              ;INC FILE BLOCK #
     76 023776  077373                           SOB     R3,4$           ;READ AND COPY REST OF FILE
     77                                 ;
     78                                 ;  Finished loading the run-time system.
     79                                 ;
     80 024000  012605                           MOV     (SP)+,R5
     81 024002                                   .CLOSE  #1
     82                                 ;
     83                                 ;  Finished
     84                                 ;
     85 024010  012604                  7$:     MOV     (SP)+,R4
     86 024012  012603                           MOV     (SP)+,R3
     87 024014  012602                           MOV     (SP)+,R2
     88 024016  012601                           MOV     (SP)+,R1
     89 024020  000207                           RETURN
     90                                 ;
     91                                 ;  Error -- Cannot find run-time system file
     92                                 ;
     93 024022                          9$:      .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
     94 024030                                   .PRINT  #COSRT          ;PRINT ERROR MESSAGE
     95 024036  012702  000004                   MOV     #4,R2           ;PRINT 4 RAD50 VALUES
     96 024042  012100                  10$:     MOV     (R1)+,R0        ;GET PART OF NAME
     97 024044  004737  025274'                  CALL    PRTR50          ;PRINT RAD50 VALUE
     98 024050  077204                           SOB     R2,10$
     99 024052                                   .PRINT  #CRLF           ;END LINE
    100 024060  000137  004134'                  JMP     INISTP          ;ABORT INITIALIZATION
    101                                 ;
    102                                 ;  Error -- Insufficient memory space to load run-time systems
    103                                 ;
    104 024064                          11$:     .PRINT  #TSXHD          ;PRINT HEADING
    105 024072                                   .PRINT  #SRTOVF         ;PRINT ERROR MESSAGE
    106 024100  000137  004134'                  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.
                                ;
024104  010246                  CSHBUF: MOV     R2,-(SP)
024106  010346                          MOV     R3,-(SP)
                                ;
                                ;  See if data caching is wanted
                                ;
024110  013737  000000G 000000G         MOV     CSHALC,VCSHNB   ;Set # blocks in use = # blocks allocated
024116  013702  000000G                 MOV     CSHALC,R2       ;Did used request data caching?
024122  001464                          BEQ     9$              ;Br if not
                                ;
                                ;  Calculate number of 64-byte blocks needed for each cache control table
                                ;
024124  062702  000037                  ADD     #31.,R2         ;Bound up to 32 word block
024130  072227  177773                  ASH     #-5.,R2         ;Convert to # 64-byte blocks
                                ;
                                ;  Compute total space that will be used by all cache data
                                ;
024134  013703  000000G                 MOV     CSHALC,R3       ;Get # blocks in cache
024140  072327  000003                  ASH     #3,R3           ;Get # 64-byte blks used by cache data buffers
024144  012700  000010                  MOV     #8.,R0          ;Get number of cache control tables
024150  060203                  1$:     ADD     R2,R3           ;Accumulate total space needed
024152  077002                          SOB     R0,1$
024154  010337  000000G                 MOV     R3,CSHSIZ       ;Save total space used by cache data
                                ;
                                ;  See if there is enough memory space available for the specified cache
                                ;
024160  010500                          MOV     R5,R0           ;Get top of memory address
024162  160400                          SUB     R4,R0           ;Compute # free 64-byte blocks
024164  020300                          CMP     R3,R0           ;Is there enough total space?
024166  103045                          BHIS    10$             ;Br if not
                                ;
                                ;  Allocate space for cache data buffers
                                ;
024170  013700  000000G                 MOV     CSHALC,R0       ;Get # blocks in data cache
024174  072027  000003                  ASH     #3,R0           ;Get # 64-byte blocks needed for allocation
024200  160005                          SUB     R0,R5           ;Allocate space for cache data buffers
024202  010537  000000G                 MOV     R5,CSHBFP       ;Save pointer to base of buffer area
                                ;
                                ;  Allocate space for each control table
                                ;
024206  160205                          SUB     R2,R5           ;Allocate space for table
024210  010537  000000G                 MOV     R5,CA$BLK       ;Block number associated with entry
024214  160205                          SUB     R2,R5           ;Allocate space for table
024216  010537  000000G                 MOV     R5,CA$DVU       ;Device and unit number
024222  160205                          SUB     R2,R5           ;Allocate space for table
024224  010537  000000G                 MOV     R5,CA$WCT       ;Number of words
024230  160205                          SUB     R2,R5           ;Allocate space for table
```

```
     58 024232  010537  000000G                 MOV     R5,CA$UFL       ;LRU chain forward link
     59 024236  160205                          SUB     R2,R5           ;Allocate space for table
     60 024240  010537  000000G                 MOV     R5,CA$UBL       ;LRU chain backward link
     61 024244  160205                          SUB     R2,R5           ;Allocate space for table
     62 024246  010537  000000G                 MOV     R5,CA$HFL       ;Hash chain forward link
     63 024252  160205                          SUB     R2,R5           ;Allocate space for table
     64 024254  010537  000000G                 MOV     R5,CA$HBL       ;Hash chain backward link
     65 024260  160205                          SUB     R2,R5           ;Allocate space for table
     66 024262  010537  000000G                 MOV     R5,CA$HSH       ;Hash chain list heads
     67 024266  020527  001600                  CMP     R5,#1600        ;Did we run over RT-11?
     68 024272  101403                          BLOS    10$             ;Br if yes
     69                                 ;
     70                                 ;  Finished
     71                                 ;
     72 024274  012603                  9$:     MOV     (SP)+,R3
     73 024276  012602                          MOV     (SP)+,R2
     74 024300  000207                          RETURN
     75                                 ;
     76                                 ;  Insufficent memory space available for cache data
     77                                 ;
     78 024302                          10$:    .PRINT  #TSXHD          ;Print heading
     79 024310                                  .PRINT  #CSHOVF         ;Overlow message
     80 024316  000137  004134'                 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.
                ;
                GETODT: MOV     R1,-(SP)
                        MOV     R2,-(SP)
                        MOV     R3,-(SP)
                        MOV     R4,-(SP)
                ;
                ;  Try to lookup ODT rel file.
                ;
                        .LOOKUP #AREA,#1,#ODTBLK ;LOOKUP ODT REL FILE
                        BCC     1$              ;BR IF FOUND IT
                        .PRINT  #TSXHD          ;CAN'T FIND ODT
                        .PRINT  #NOODT
                        JMP     INISTP          ;ABORT INITIALIZATION
                ;
                ;  Read first block of ODT file and determine size of ODT.
                ;
                1$:     ADD     #200.,R5        ;RESERVE SPACE FOR ODT STACK
                        MOV     R5,R0           ;CHECK MEMORY ADDRESS FOR OVERFLOW
                        ADD     #512.,R0
                        CALL    CHKMEM
                        .READW  #AREA,#1,R5,#256.,#0
                        BCC     2$              ;Br if no read error
                        JMP     ODTRDX          ;Read error
                2$:     MOV     RSZ(R5),R2      ;GET SIZE OF ODT
                        MOV     R2,R3
                        ADD     R5,R3           ;GET ADDRESS ABOVE TOP OF ODT
                        MOV     R3,R0           ;CHECK MEMORY ADDRESS FOR OVERFLOW
                        CALL    CHKMEM
                        CLC
                        ROR     R2              ;GET # WORDS IN ODT
                ; Get starting address of ODT
                        MOV     STA(R5),R0      ;GET OFFSET TO START ADDRESS
                        SUB     #ODTBAS,R0      ;CALCULATE ABSOLUTE STARTING ADDRESS
                        ADD     R5,R0
                        MOV     R0,ODTSTA       ;THIS IS REAL STARTING ADDRESS
                        MOV     RBD(R5),R1      ;GET # OF BLOCK WITH RELOCATION INFO
                ;
                ;  Read in ODT rel file image.
                ;
                        .READW  #AREA,#1,R5,R2,#1
                        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.
                ;
```

```
                        RELFIL: MOV     R3,ODTTOP       ;SAVE ADDRESS ABOVE TOP OF ODT
                                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
                                MOV     WRKBUF,RLBF     ;READ RELOCATION INFO HERE
                        1$:     MOV     RLBF,RLBFND
                                ADD     #1024.,RLBFND   ;GET ADDRESS OF END OF BUFFER AREA
                                MOV     R5,R4           ;GET BASE ADDRESS OF ODT
                                SUB     #ODTBAS,R4      ;SUBTRACT LINK BASE ADDRESS
                        ;  Read in relocation address list.
                        4$:     .READW  #AREA,#1,RLBF,#512.,R1
                                BCC     7$              ;BR IF NO READ ERROR
                                TSTB    @#52            ;END OF FILE IS OK
                                BNE     ODTRDX          ;BR IF READ ERROR
                        ;  Relocate some addresses in ODT.
                        7$:     MOV     RLBF,R2         ;POINT TO RELOCATION INFO
                        3$:     MOV     (R2)+,R3        ;GET ADDRESS OF LOCATION TO RELOCATE
                                CMP     R3,#-2          ;TIME TO STOP?
                                BEQ     9$              ;BR IF FINISHED
                                MOV     (R2)+,R0        ;GET VALUE TO RELOCATE
                                ASL     R3              ;CVT TO BYTE ADDRESS
                                BCC     5$              ;BR IF ADDITIVE RELOCATION
                                SUB     R4,R0           ;RELOCATE THE ADDRESS
                                BR      6$
                        5$:     ADD     R4,R0           ;RELOCATE THE ADDRESS
                        6$:     ADD     R5,R3           ;GET LOCATION WHERE WORD GOES
                                MOV     R0,@R3          ;STORE RELOCATED ADDRESS
                                CMP     R2,RLBFND       ;TIME TO READ NEXT BUFFER FULL?
                                BLO     3$              ;BR IF NOT
                                ADD     #2,R1           ;ADVANCE BLOCK #
                                BR      4$              ;GO READ NEXT BUFFER FULL
                        ;
                        ;  Finished relocation.
                        ;  Close ODT rel file.
                        ;
                        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
                        ;
                            MOV     #TSINIT,R0
                            MOV     #BRKPT,R1
                            MOV     #TSGEN,R2
```

```
                        MOV     #TSEXEC,R3
                        MOV     #TSEMT,R4
                        MOV     #SYSMAP,-(SP)   ;PASS ADDRESS OF SYSMAP CELL TO ODT
                        MOV     #MAPSYS,-(SP)   ;PASS ADDRESS OF MAPSYS ROUTINE
                        MOV     #10$,R5         ;ADDRESS FOR ODT TO RETURN TO
                ;
                ;  Enter ODT
                ;
                        JMP     @ODTSTA         ;JUMP TO START OF ODT
                ;
                ;  Return from ODT.
                ;  Continue initialization of TSX.
                ;
                10$:    MOV     @#14,ODTTRP     ;SAVE ODT BREAKPOINT ENTRY ADDRESS
                        MOV     ODTTOP,R5       ;ADDRESS ABOVE TOP OF ODT
                        MOV     (SP)+,R4
                        MOV     (SP)+,R3
                        MOV     (SP)+,R2
                        MOV     (SP)+,R1
                        RETURN
                ;
                ;  Error while reading ODT rel file.
                ;
                ODTRDX: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
                        .PRINT  #ODTRDM
                        JMP     INISTP          ;ABORT INITIALIZATION
                ;
                ;  RTI instruction to disable interrupts
                ;
                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.
                                ;
024322  010146                  OPNCHN: MOV     R1,-(SP)
024324  010346                          MOV     R3,-(SP)
024326  010003                          MOV     R0,R3           ;Carry channel block address in R3
                                ;
                                ;  Initialize the channel block
                                ;
024330  010301                          MOV     R3,R1           ;Point to the channel block
024332  012700  000000C               MOV     #<CHNSIZ/2>,R0  ;Get # words to zero
024336  005021                  2$:     CLR     (R1)+           ;Zero the channel block
024340  077002                          SOB     R0,2$
024342  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 #
                                ;
024350  010200                          MOV     R2,R0           ;Get the full device name
024352  004737  011476'               CALL    CVTDVU          ;Convert to dev # and unit #
024356  103411                          BCS     9$              ;Br if we don't recognize the device name
024360  010001                          MOV     R0,R1           ;Get index # and unit #
024362  000301                          SWAB    R1              ;Get unit # to low byte
024364  110163  000000G               MOVB    R1,C.DEVQ(R3)   ;Set unit # in channel block
024370  042700  000000C               BIC     #^C<CS$NMX>,R0  ;Clear all but device index number in R0
024374  050063  000000G               BIS     R0,C.CSW(R3)    ;Store device index # into CSW
                                ;
                                ;  Success
                                ;
024400  000241                          CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
024402  012603                  9$:     MOV     (SP)+,R3
024404  012601                          MOV     (SP)+,R1
024406  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.
                                ;
024410  010146                 SETCHN: MOV     R1,-(SP)
024412  010246                         MOV     R2,-(SP)
024414  010346                         MOV     R3,-(SP)
024416  010001                         MOV     R0,R1           ;GET ADDRESS OF TSX CHANNEL BLOCK
                                ;
                                ;  Do .SAVESTATUS to store channel information into TSX channel block.
                                ;
024420                                  .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.
                                ;
024436  011103                         MOV     (R1),R3         ;GET CSW FOR CHANNEL
024440  042703  177701                 BIC     #^C76,R3        ;GET RT-11 DEVICE INDEX NUMBER
024444                                  .GVAL   #AREA,#404      ;GET RT-11 OFFSET TO PNAME TABLE
024464  060003                         ADD     R0,R3           ;GET ADDRESS OF NAME OF DEVICE IN PNAME TABLE
024466                                  .GVAL   #AREA,R3        ;GET NAME OF DEVICE FROM RT-11
024504  013703  000000G                MOV     NUMDEV,R3       ;GET INDEX # FOR LAST TSX DEVICE
024510  020063  000000G        1$:     CMP     R0,PNAME(R3)    ;LOOK FOR DEVICE IN OUR TABLES
024514  001404                         BEQ     2$              ;BR IF FOUND
024516  162703  000002                 SUB     #2,R3           ;CHECK NEXT ENTRY
024522  002372                         BGE     1$              ;BR IF MORE TO CHECK
024524  000407                         BR      MTSXDV          ;VERY STRANGE THAT WE DIDN'T FIND IT
024526  042711  000076         2$:     BIC     #76,(R1)        ;CLEAR OUT RT-11 DEVICE #
024532  050311                         BIS     R3,(R1)         ;STORE TSX DEVICE #
                                ;
                                ;  Finished
                                ;
024534  012603                         MOV     (SP)+,R3
024536  012602                         MOV     (SP)+,R2
024540  012601                         MOV     (SP)+,R1
024542  000207                         RETURN
                                ;
                                ;  Error: Could not locate Rt-11 device number in TSX device table.
                                ;
024544                          MTSXDV: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
024552                                  .PRINT  #REQMIS         ;Missing a required device
024560  010200                         MOV     R2,R0           ;GET RAD50 DEVICE NAME
024562  004737  025274'                 CALL    PRTR50          ;DISPLAY DEVICE NAME
024566                                  .PRINT  #CRLF
```

```
     58 024574  000137  004134'                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
                                ;
024600  010146                  SETSY:  MOV     R1,-(SP)
024602  010246                          MOV     R2,-(SP)
                                ;
                                ;  Set up system device unit number
                                ;
024604                                  .GVAL   #AREA,#274      ;Get system unit # from RT-11 (high byte)
024624  010037  000000G                 MOV     R0,SYUNIT       ;Set system unit number
                                ;
                                ;  Set up system device index number
                                ;
024630                                  .GVAL   #AREA,#364      ;Get RT-11 system device index number
024650  010002                          MOV     R0,R2           ;Save device index number
024652                                  .GVAL   #AREA,#404      ;Get offset within RMON of PNAME table
024672  060002                          ADD     R0,R2           ;Get offset to name of SY device
024674                                  .GVAL   #AREA,R2        ;Get name of RT-11 system device
024712  013701  000000G                 MOV     NUMDEV,R1       ;Get index to last TSX-Plus device entry
024716  020061  000000G          1$:    CMP     R0,PNAME(R1)    ;Search for device in TSX tables
024722  001405                          BEQ     2$              ;Br if found it
024724  162701  000002                  SUB     #2,R1           ;Keep looking if more
024730  002372                          BGE     1$
024732  010002                          MOV     R0,R2           ;Save name of system device
024734  000703                          BR      MTSXDV          ;Missing device error
024736  010137  000000G          2$:    MOV     R1,SYINDX       ;Store index # of TSX-Plus system device
                                ;
                                ;  Set up RAD50 name of SY disk
                                ;
024742  113702  000001G                 MOVB    SYUNIT+1,R2     ;GET SYSTEM UNIT NUMBER
024746  062702  000036                  ADD     #36,R2          ;PUT IN "0" AS 3'RD CHARACTER OF NAME
024752  066102  000000G                 ADD     PNAME(R1),R2    ;ADD DEVICE NAME
024756  010237  000000G                 MOV     R2,SYNAME       ;THIS IS THE FULL SY DISK NAME
                                ;
                                ;  Finished
                                ;
024762  012602                          MOV     (SP)+,R2
024764  012601                          MOV     (SP)+,R1
024766  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.
                                ;
024770  010046                 RTFTCH: MOV     R0,-(SP)
024772  010246                         MOV     R2,-(SP)
024774  010546                         MOV     R5,-(SP)
                                ;
                                ;  Set the name of the device being fetched
                                ;
024776  010037  000130'                 MOV     R0,FETDEV       ;Set name of device whose handler to fetch
                                ;
                                ;  Do a .DSTAT to get information about the handler
                                ;
025002                                  .DSTAT  #DSTBLK,#FETDEV ;Get information about the device handler
025014  103425                          BCS     9$              ;Br if device not recognized
                                ;
                                ;  Determine if the handler is currently resident
                                ;
025016  005737  000116'                 TST     DSTBLK+4        ;Is the handler resident now?
025022  001021                          BNE     8$              ;Br if yes
                                ;
                                ;  The handler is not currently resident.
                                ;  See if it will fit in WRKBUF.
                                ;
025024  013702  000152'                 MOV     WRKBUF,R2       ;Set address where handler will be loaded
025030  013700  000114'                 MOV     DSTBLK+2,R0     ;Get the size of the handler
025034  020037  000154'                 CMP     R0,WRKSIZ       ;Will handler fit in WRKBUF?
025040  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.
                                ;
025042  060500                          ADD     R5,R0           ;Get address above top of area needed
025044  020037  000132'                 CMP     R0,TOPMEM       ;Is there room for handler?
025050  101013                          BHI     10$             ;Br if not
025052  010502                          MOV     R5,R2           ;Set address where handler is to be loaded
                                ;
                                ;  Fetch the handler
                                ;
025054                          1$:     .FETCH  R2,#FETDEV      ;Try to fetch the handler
025064  103401                          BCS     9$              ;Br if error on fetch
                                ;
                                ;  We successfully fetched the handler
                                ;
```

```
     58 025066  000241                  8$:     CLC                     ;Signal success on return
     59                                  ;
     60                                  ;  Finished
     61                                  ;
     62 025070  012605                  9$:     MOV     (SP)+,R5
     63 025072  012602                          MOV     (SP)+,R2
     64 025074  012600                          MOV     (SP)+,R0
     65 025076  000207                          RETURN
     66                                  ;
     67                                  ;  Insufficient memory available to load the handler
     68                                  ;
     69 025100  004737  025124'          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.
                                ;
025104  020037  000236'         CHKMEM: CMP     R0,MEMLIM       ;IS THE ADDRESS OK?
025110  103402                         BLO     1$              ;BR IF OK
025112  004737  025124'                 CALL    SIZERR          ;Generated system is too big -- abort
025116  004737  025144'         1$:     CALL    CCATST          ;CHECK FOR ^C ABORT REQUEST
025122  000207                         RETURN

                                ;-------------------------------------------------------------------------
                                ;  Generated system is too big.  Abort the initialization.
                                ;
025124                          SIZERR: .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
025132                                  .PRINT  #TOOBIG         ;PRINT ERROR MESSAGE
025140  000137  004134'                 JMP     INISTP          ;ABORT INITIALIZATION

                                ;-------------------------------------------------------------------------
                                ;  Check for control-C and abort initialization if requested.
                                ;
025144  005737  000040'         CCATST: TST     CCAFLG          ;DID USER REQUEST ^C ABORT?
025150  001402                         BEQ     1$              ;BRANCH IF NOT
025152  000137  004134'                 JMP     INISTP          ;ELSE ABORT INITIALIZATION
025156  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.
                                ;
025160  010146          PRTOCT: MOV     R1,-(SP)
025162  010246                  MOV     R2,-(SP)
025164  010001                  MOV     R0,R1           ;GET VALUE TO PRINT
025166  012702  000006          MOV     #6,R2           ;PRINT 6 DIGITS
025172  005000                  CLR     R0
025174  073027  000001          ASHC    #1,R0           ;GET 1ST OCTAL DIGIT (1 BIT)
025200  000403                  BR      2$
025202  005000          1$:     CLR     R0              ;INITIALIZE FOR SHIFT
025204  073027  000003          ASHC    #3,R0           ;SHIFT AN OCTAL DIGIT INTO R0
025210  062700  000060  2$:     ADD     #'0,R0          ;CONVERT TO ASCII CHARACTER
025214                          .TTYOUT                 ;PRINT THE CHARACTER
025220  077210                  SOB     R2,1$           ;LOOP AND PRINT MORE DIGITS
025222  012602                  MOV     (SP)+,R2
025224  012601                  MOV     (SP)+,R1
025226  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
                ;
025230  010146  PRTDEC: MOV     R1,-(SP)
025232  005046         CLR     -(SP)           ;NULL ON STACK TO STOP US
                ;
                ;  Convert value to ascii digit string and stack the digits.
                ;
025234  010001         MOV     R0,R1           ;GET VALUE TO BE CONVERTED
025236  005000  1$:    CLR     R0              ;SET HIGH-ORDER PART OF VALUE TO 0
025240  071027  000012         DIV     #10.,R0         ;DIVIDE R0-R1 BY 10.
025244  062701  000060         ADD     #'0,R1          ;CONVERT REMAINDER TO ASCII DIGIT
025250  010146         MOV     R1,-(SP)        ;AND STACK THE DIGIT
025252  010001         MOV     R0,R1           ;GET QUOTIENT
025254  001370         BNE     1$              ;BR IF MORE DIGITS TO CONVERT
                ;
                ;  Finished conversion.  Print result.
                ;
025256  012600  2$:    MOV     (SP)+,R0        ;GET A DIGIT FROM THE STACK
025260  001403         BEQ     3$              ;BR IF REACHED END
025262         .TTYOUT                         ;PRINT THE DIGIT
025266  000773         BR      2$              ;PRINT MORE
                ;
                ;  Finished
                ;
025270  012601  3$:    MOV     (SP)+,R1
025272  000207         RETURN
```

```
                                        .SBTTL  PRTR50 -- Print Rad-50 value
                                ;-------------------------------------------------------------------------
                                ;  PRTR50 is called to print a Rad-50 value.
                                ;
                                ;  Inputs:
                                ;   R0 = value to be printed.
                                ;
025274  010146                  PRTR50: MOV     R1,-(SP)
025276  010246                          MOV     R2,-(SP)
                                ;
                                ;  Convert value to ascii string and stack the characters.
                                ;
025300  012702  000003                  MOV     #3,R2           ;GET # CHARS TO CVT
025304  010001                          MOV     R0,R1           ;GET VALUE TO BE CONVERTED
025306  005000                  1$:     CLR     R0              ;CLEAR HIGH-ORDER VALUE
025310  071027  000050                  DIV     #50,R0          ;DIVIDE R0-R1 BY 50
025314  116101  003347'                 MOVB    R50CHR(R1),R1   ;CONVERT REMAINDER TO ASCII CHARACTER
025320  010146                          MOV     R1,-(SP)        ;STACK THE CHARACTER
025322  010001                          MOV     R0,R1           ;GET QUOTIENT
025324  077210                          SOB     R2,1$           ;BR IF MORE CHARS TO CONVERT
                                ;
                                ;  Finished conversion.  Print the result.
                                ;
025326  012702  000003                  MOV     #3,R2           ;GET # CHARS TO PRINT
025332  012600                  2$:     MOV     (SP)+,R0        ;GET NEXT CHARACTER
025334                                  .TTYOUT                 ;PRINT THE CHARACTER
025340  077204                          SOB     R2,2$           ;LOOP IF MORE CHARS TO PRINT
                                ;
                                ;  Finished
                                ;
025342  012602                          MOV     (SP)+,R2
025344  012601                          MOV     (SP)+,R1
025346  000207                          RETURN
                                ;-------------------------------------------------------------------------
                                ;  Define top of TSINIT
                                ;
025350                          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
                                ;
025350  177740                  EDARGB: .WORD   -32.                     ;# OF BYTES TO BE DECRYPTED (EDMTH3)
025352  025660'                 EDADDR: .WORD   DSKBUF                  ;POINTER TO BUFFER TO BE DECRYPTED
                                ;
                                ;  Recover license number and decrypt disk image of Pro ID to intermed. state
                                ;
025354  010146                  INSCHK: MOV     R1,-(SP)                ;SAVE REGISTERS
025356  010246                          MOV     R2,-(SP)
025360  010346                          MOV     R3,-(SP)
025362  010446                          MOV     R4,-(SP)
025364  010546                          MOV     R5,-(SP)
025366  012700                          MOV     (PC)+,R0                ;DECRYPT LICENSE NUMBER
025370  073472                          .RAD50  /SCB/                   ;WITH THIS CODE
025372  074037  025720'                 XOR     R0,LICNUM               ;BY XORING IT
025376  013737  025720' 000000G         MOV     LICNUM,TSXSIT           ;MOVE LICENCE NUMBER TO TSGEN CELL
025404  012700  025350'                 MOV     #EDARGB,R0              ;POINT TO ENC/DEC ARG BLOCK (PRESET)
025410  004737  026046'                 CALL    EDMTH3                  ;DECRYPT TO INTERMED STATE
                                ;
                                ;  Copy Pro ID ROM low bytes into memory, and encrypt 1 step
                                ;
                        173600  IDADDR  = 173600          ;ADDRESS OF START OF PRO 350 ID ROM
025414  012701  173600                  MOV     #IDADDR,R1              ;GET POINTER TO PRO ID ROM
025420  012702  025722'                 MOV     #ROMBUF,R2              ;POINTER TO COPY OF HARDWARE ID
025424  010237  025352'                 MOV     R2,EDADDR               ;SAVE ADDRESS FOR ENCRYPTION
025430  005437  025350'                 NEG     EDARGB                  ;MAKE +32. FOR ENCRYPTION
025434  013700  025350'                 MOV     EDARGB,R0               ;ALSO USE AS LOOP COUNTER
025440  112122                  3$:     MOVB    (R1)+,(R2)+             ;GET NEXT LOW BYTE
025442  005201                          INC     R1                      ;SKIP ID ROM HIGH BYTES
025444  077003                          SOB     R0,3$                   ;REPEAT THROUGH 32 BYTE ROM
025446  012700  025350'                 MOV     #EDARGB,R0              ;POINT TO ENCRYPTION ARG BLOCK
025452  004737  025762'                 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.
                                ;
025456  012701  025722'                 MOV     #ROMBUF,R1              ;POINT TO HARDWARE COPY OF ID
025462  012702  025660'                 MOV     #DSKBUF,R2              ;POINT TO DISK COPY OF ID
025466  012704  000000G                 MOV     #SCHED,R4               ;POINT TO CODE TO BE CORRECTED
025472  013703  025350'                 MOV     EDARGB,R3               ;INIT LOOP COUNTER
025476  004737  025652'                 CALL    GETLIC                  ;USE LIC # AS SEED FOR EDPRNW, IN R0
025502  122122                  4$:     CMPB    (R1)+,(R2)+             ;VERIFY ID'S ARE THE SAME
025504  001014                          BNE     5$                      ;ABORT IF NO MATCH ON ANY BYTE
025506  004737  026250'                 CALL    EDPRNW                  ;RANDOMIZE R0 FOR XOR (LIC# INIT SEED)
025512  011405                          MOV     @R4,R5                  ;GET ENCRYPTED CODE
025514  074005                          XOR     R0,R5                   ;RESTORE FUNCTIONAL CODE
025516  010524                          MOV     R5,(R4)+                ;PUT DECRYPTED CODE BACK IN MEMORY
```

```
     58 025520  077310                          SOB     R3,4$                   ;REPEAT THROUGH ID TESTS
     59 025522  012605                          MOV     (SP)+,R5                ;RESTORE REGISTERS
     60 025524  012604                          MOV     (SP)+,R4
     61 025526  012603                          MOV     (SP)+,R3
     62 025530  012602                          MOV     (SP)+,R2
     63 025532  012601                          MOV     (SP)+,R1
     64 025534  000207                          RETURN                          ;ID CHECKS AND CODE DECRYPTED
     65                                 ;
     66 025536                          5$:     .PRINT  #TSXHD                  ;?TSX-F
     67 025544                                  .PRINT  #NOTLIC                 ;NOT LICENSED FOR THIS MACHINE
     68 025552  000137  004134'                 JMP     INISTP                  ;ID'S DON'T MATCH, ABORT INIT.
     69                                 ;
     70                                         .NLIST  BEX
     71 025556     124     150     151 NOTLIC: .ASCIZ  /This copy of TSX-Plus not licensed for use on this machine./
     72                                         .LIST   BEX
     73                                         .EVEN
     74                                 ;
     75                                 ;  Subroutine to recover incremental license number.  Assume it has been
     76                                 ;  decrypted already by XORing with .RAD50 /SCB/.
     77                                 ;
     78 025652  013700  025720'         GETLIC: MOV     LICNUM,R0               ;RETRIEVE DECRYPTED LIC # INTO R0
     79 025656  000207                          RETURN
     80                                 ;
     81                                 ;  Reserve room for both disk and hardware copies of the Pro ID number
     82                                 ;  and for the incremental license number
     83                                 ;
     84 025660                          DSKBUF: .BLKB   32.                     ;DISK IMAGE OF PRO ID
     85 025720  000000                  LICNUM: .WORD   0                       ;INCREMENTAL LICENSE NUMBER
     86 025722                          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
                                ;
025762                          EDMTH2:
025762  010146                          MOV     R1,-(SP)        ;Save registers
025764  010246                          MOV     R2,-(SP)
025766  010346                          MOV     R3,-(SP)
                                ;
                                ;  Fetch byte count, buffer pointer and initialize PRN seed
                                ;
025770  012003                          MOV     (R0)+,R3        ;Fetch byte count of input buffer
025772  011001                          MOV     (R0),R1         ;Fetch pointer to input buffer
025774  004737  025652'                 CALL    GETLIC          ;Use license number as initial PRN seed
026000  010002                          MOV     R0,R2           ;Copy license number to form repeat count
026002  042702  177770                  BIC     #^C7,R2         ;No more than 8 repeats
026006  005202                          INC     R2              ;Make sure there is at least one
026010  004737  026250'         2$:     CALL    EDPRNW          ;Get a new PRN
026014  077203                          SOB     R2,2$           ;Advance the seed between 1 and 8 times
                                ;
                                ;  Now sweep the buffer, XORing each byte with the high PRN byte
                                ;
026016  004737  026250'         1$:     CALL    EDPRNW          ;With seed in R0, get next random number
026022  111102                          MOVB    (R1),R2         ;Get next input byte
```

```
     58 026024  000300                    SWAB    R0              ;Reverse PRN high and low bytes
     59 026026  074002                    XOR     R0,R2           ;Encrypt the byte
     60 026030  000300                    SWAB    R0              ;Restore PRN high and low bytes for next seed
     61 026032  110221                    MOVB    R2,(R1)+        ;Save encrypted bytes back into input buffer
     62 026034  077310                    SOB     R3,1$           ;Repeat for entire input buffer
     63                                   ;
     64 026036  012603                    MOV     (SP)+,R3        ;Restore registers
     65 026040  012602                    MOV     (SP)+,R2
     66 026042  012601                    MOV     (SP)+,R1
     67 026044  000207                    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
                                ;
026046  010146          EDMTH3: MOV     R1,-(SP)                ;Save registers
026050  010246                  MOV     R2,-(SP)
026052  010346                  MOV     R3,-(SP)
026054  010446                  MOV     R4,-(SP)
026056  010546                  MOV     R5,-(SP)
026060  012003                  MOV     (R0)+,R3                ;Save the string length
026062  011046                  MOV     @R0,-(SP)               ;And save the input buffer pointer
                                ;  Initialize prn generator of desired length
026064  010300                  MOV     R3,R0                   ;Recover string length
026066  002002                  BGE     1$                      ;Branch if encryption
026070  005400                  NEG     R0                      ;If decryption, get real repeat
026072  005203                  INC     R3                      ;If neg, correct for ASR round down
026074  004737  026300'  1$:     CALL    INPRNM                  ;Set up for desired repeat length
                                ;  Start encryption loop through string
026100  006203                  ASR     R3                      ;Repeat for 1/2 the string length
026102  004737  025652'          CALL    GETLIC                  ;Get lic. num. for initial seed in R0
026106  004737  026324'          CALL    EDPRNM                  ;Seed PRN generator (-adjacent pairs)
026112  005703          2$:     TST     R3                      ;Less than 2 bytes left?
026114  001446                  BEQ     9$                      ;Quit if so (odd len -> 1 byte unch.)
026116  005004                  CLR     R4                      ;Clean out shifting registers
026120  005005                  CLR     R5
026122  011601                  MOV     @SP,R1                  ;Retrieve buffer pointer
026124  010102                  MOV     R1,R2                   ;And second copy
                                ;  Select first random byte
026126  004737  026324'          CALL    EDPRNM                  ;Randomize in range 0 - <strlen-1>
026132  060001                  ADD     R0,R1                   ;Point to first random byte of pair
                                ;  Select second random byte
026134  004737  026324'          CALL    EDPRNM                  ;Randomize again
026140  060002                  ADD     R0,R2                   ;Point to next random byte
                                ;  Use part of PRNM as semi-random shift amount
026142  010046                  MOV     R0,-(SP)                ;Save EDPRNM seed for later
026144  042700  177771          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
                ;
026250          EDPRNW:
026250  010446          MOV     R4,-(SP)        ;Save registers
026252  010546          MOV     R5,-(SP)
026254  010004          MOV     R0,R4           ;Get seed to be multiplied
026256  012700          MOV     (PC)+,R0        ;Fetch multiplier
026260  104375  EDPRNA: .WORD   104375          ;Multiplier, can be replaced
026262  070400          MUL     R0,R4           ;Multiply by A
026264  062705          ADD     (PC)+,R5        ;Add C
026266  012705  EDPRNC: .WORD   012705          ;Additive, can be replaced
026270  010500          MOV     R5,R0           ;Return result mod 65536. as PRN
026272  012605          MOV     (SP)+,R5        ;Restore registers
026274  012604          MOV     (SP)+,R4
026276  000207          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
                                ;
026300                          INPRNM:
026300  012737  000040  026366' MOV     #32.,EDMODM     ;Get repeat length to cover Pro ID
026306  012737  000005  026354' MOV     #5,EDMULA       ;Use first valid A
026314  012737  000003  026360' MOV     #3,EDADDC       ;And first valid C
026322  000207                  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
                                ;
026324                          EDPRNM:
026324  010446                         MOV     R4,-(SP)        ;Save R4 and R5
026326  010546                         MOV     R5,-(SP)
026330  020037  026366'                CMP     R0,EDMODM       ;Is seed in range 0 to EDMODM?
026334  103405                         BLO     1$              ;Branch and proceed if so
026336  010005                         MOV     R0,R5           ;Set up to divide it by EDMODM
026340  005004                         CLR     R4              ;Set up for divide
026342  071437  026366'                DIV     EDMODM,R4       ;Divide it
026346  010500                         MOV     R5,R0           ;And use remainder as seed
026350  010004                 1$:     MOV     R0,R4           ;Get current seed ready to be multiplied
026352  070427                         MUL     (PC)+,R4        ;Multiply by chosen A
026354  061125                 EDMULA: .WORD   25173.          ;Replace at run-time with 5
026356  062705                         ADD     (PC)+,R5        ;Add in C
026360  033031                 EDADDC: .WORD   13849.          ;Replace at run-time with 3
026362  005004                         CLR     R4              ;Clear high word for division
026364  071427                         DIV     (PC)+,R4        ;Perform mod M
026366  000400                 EDMODM: .WORD   256.            ;Replace at run-time with 32.
026370  010500                         MOV     R5,R0           ;Return remainder
026372  012605                         MOV     (SP)+,R5        ;Restore R4 and R5
026374  012604                         MOV     (SP)+,R4
026376  000207                         RETURN
                                ;
                                .IFF    ;NE,PROCID              ;Assemble if protection code not included
                                DSKBUF:                         ;Define dummy DSKBUF global symbol
                                .ENDC   ;NE,PROCID
                                ;
                                ;  Address of real top of TSINIT, including PRO init code
                                ;
                                PROITP:
                                        .CSECT  TSXEND
000001                                  .END
```

Symbol table

```
AHEND = ****** G
ALBFX   012536R     002 CL$EPN= ****** G
ALCHRB  007562R     002 CL$EPP= ****** G
ALCOVL  022130R     002 CL$EPS= ****** G
ALCSLO  012316R     002 CL$LEN= ****** G
ALCWRK  007526R     002 CL$LIX= ****** G
ALOCBF  011652R     002 CL$OPT= ****** G
AREA    000000R     002 CL$ORA= ****** G
AUTHAN= ****** G        CL$ORB= ****** G
BADDEV  010170R     002 CL$ORE= ****** G
BADLIN  000764R     002 CL$ORG= ****** G
BADOPN  001137R     002 CL$ORP= ****** G
BASMAP= ****** G        CL$ORS= ****** G
BDLMSG  001055R     002 CL$STA= ****** G
BDSPOP  001305R     002 CL$VER= 000002
BDVMSG  001017R     002 CONFG2= ****** G
BOSF    001342R     002 CONFIG= ****** G
BRKPT = ****** G        CONSPC  001237R
BYTES   002602R     002 COSRT   002702R
CA$BLK= ****** G        CO$DEF= ****** G
CA$DVU= ****** G        CO$FF = ****** G
CA$HBL= ****** G        CO$TAB= ****** G
CA$HFL= ****** G        CO$8BT= ****** G
CA$HSH= ****** G        CRLF    001067R
CA$UBL= ****** G        CSHALC= ****** G
CA$UFL= ****** G        CSHBAS= ****** G
CA$WCT= ****** G        CSHBFP= ****** G
CCAFLG  000040R     002 CSHBUF  024104R
CCATST  025144R     002 CSHDEV= ****** G
CCBHD = ****** G        CSHDVN= ****** G
CCLNAM  000102R     002 CSHOVF  003211R
CCLSAV= ****** G        CSHSIZ= ****** G
CC$$SZ= ****** G        CSHVEC= ****** G
CDX$DH= ****** G        CS$ENT= ****** G
CDX$DL= ****** G        CS$NMX= ****** G
CDX$DZ= ****** G        CS$OPN= ****** G
CDX$VH= ****** G        CURDEV  000144R
CD$$SZ= ****** G        CURNAM  000146R
CFHMSG  001510R     002 CVTDVU  011476R
CHAIN = ****** G        CW$BTH= ****** G
CHKCLD  011372R     002 CW$ESP= ****** G
CHKMEM  025104R     002 CW$FB = ****** G
CHNSIZ= ****** G        CW$FGJ= ****** G
CKLIN   006644R     002 CW$GDH= ****** G
CLDEVX= ****** G        CW$LGS= ****** G
CLEOFS= ****** G        CW$PRO= ****** G
CLHEAD= ****** G        CW$QBS= ****** G
CLINCP= ****** G        CW$USR= ****** G
CLINIT  013444R     002 CW$XM = ****** G
CLKRTI= ****** G        CW$5OH= ****** G
CLKVEC= ****** G        CXTALC  015646R
CLK100  000042R     002 CXTBAS= ****** G
CLORSZ= ****** G        CXTBUF= ****** G
CLOTIR= ****** G        CXTPAG= ****** G
CLSIZE= ****** G        CXTPDR= ****** G
CLSTS = ****** G        CXTRMN= ****** G
CLTOTL= ****** G        CXTSIZ= ****** G
CLVERS= ****** G        CXTWDS= ****** G
C.CSW = ****** G        EDADDR  025352R     002 HANXMR  021370RG    002
C.DEVQ= ****** G        EDARGB  025350R     002 HF$LIN= ****** G
C.SBLK= ****** G        EDMODM  026366R     002 HF$MC = ****** G
C1DEVX= ****** G        EDMTH2  025762R     002 HF$RIE= ****** G
DATIMH= ****** G        EDMTH3  026046R     002 HF$TIE= ****** G
DATIML= ****** G        EDMULA  026354R     002 HF$TSB= ****** G
DCCSIZ= ****** G        EDPRNA  026260R     002 HGENFL  000240R     002
DC$$SZ= ****** G        EDPRNC  026266R     002 HIMAP = ****** G
DEVSIZ= ****** G        EDPRNM  026324R     002 HLERR   020472R     002
DEVVEC  004652R     002 EDPRNW  026250R     002 HMAP    000126R     002
DFJMEM= ****** G    002 EMMAP = ****** G        HN2BIG  002335R     002
DHBFSZ= ****** G        EMTENT= ****** G        HSGER   002027R     002
DHINIT  005260R     002 ENTVEC  004464R     002 HVEND   000336R     002
DHLPRM  005260R     002 ERHINS  001721R     002 HVTBL   000322R     002
DHOINT= ****** G    002 ERHMSG  001552R     002 HV$DMY= 000001
DI$CL = ****** G        ERHNDV  001616R     002 HV$ID = 000000
DI$DU = ****** G        ERRLOG= ****** G        HV$VER= 000002
DI$LD = ****** G        EXCBUF= ****** G        HV$$SZ= 000004
DI$MU = ****** G        EXTLSI= 000001          H.CSR = ****** G
DI$PI = ****** G        FC$LBN= ****** G        H.DSTS= ****** G
DI$TT = ****** G        FC$$SZ= ****** G        H.DVSZ= ****** G
DI$XL = ****** G        FETDEV  000130R     002 H.GEN = ****** G
DMYDEV= ****** G        FF$$SZ= ****** G        H.INS = ****** G
DM$CSR= ****** G        FILBLK  000142R     002 H.SIZ = ****** G
DM$LSR= ****** G        FMEMHI  000134R     002 ICONFG  000242R     002
DOHNLC  020542R     002 FMEMLO  000136R     002 IDADDR= 173600
DSKBUF  025660RG    002 FNDHRB  021364RG    002 IDSFLG= ****** G
DSTBLK  000112R     002 FORCEO  011574R     002 II$$SZ= ****** G
DS$ABT= ****** G    002 FORK  = ****** G        ILSW2 = ****** G
DS$DIR= ****** G        FPTRAP= ****** G        INDDBL= ****** G
DS$NRD= ****** G        FQ$$SZ= ****** G        INDDBS= ****** G
DS$SFN= ****** G        FREIOQ= ****** G        INDFIL= ****** G
DS$VSZ= ****** G        FREPGS= ****** G        INDINI  014110R     002
DTLX  = ****** G        FRKGEN= ****** G        INDNAM  000220R     002
DTYPE = ****** G        FRKINI= ****** G        INDOPN  003262R     002
DVFLAG= ****** G        FSTDL = ****** G        INDSAV= ****** G
DVFLBS  000336R     002 FSTIOL= ****** G        INDTSV= ****** G
DVFLND  000516R     002 FW$$SZ= ****** G        INICLK  006524R     002
DVSTAT= ****** G    002 GENTOP= ****** G        INIDEV  006672R     002
DV$FLG= 000002          GETHNH  017634R     002 INIJMP= ****** G
DV$NAM= 000000          GETHNL  016320R     002 INIOVL  004312R     002
DV$$SZ= 000004          GETLIC  025652R     002 INISTP  004134R     002
DW$$SZ= ****** G        GETMAP  022010R     002 INITGO  005452RG    002
DX$DMA= ****** G        GETOVL  023116R     002 INITOP  025350RG    002
DX$EBA= ****** G        GETSRT  023472R     002 INMXV = ****** G
DX$IBH= ****** G        GTBYT = ****** G        INPRNM  026300R     002
DX$MAP= ****** G        GTLIN   000176R     002 INRECV= ****** G
DX$MPH= ****** G        HANDSK= ****** G        INSCHK  025354R     002
DX$NCA= ****** G        HANENT= ****** G        INSCK1  016602R     002
DX$NHM= ****** G        HANMAP  021242R     002 INSCK2  016760R     002
DX$NMT= ****** G        HANNAM  000200R     002 INSTBL= ****** G
DX$NRD= ****** G        HANPAR= ****** G        INSTBN= ****** G
DX$NST= ****** G        HANRCB= ****** G        INTEN = ****** G
DZINIT  005260R         HANRCO= ****** G        INTMX1= ****** G
DZOINT= ****** G    002 HANSIZ= ****** G        INTSND= ****** G
EDADDC  026360R         HANUMP  021316R     002 INTSSZ= ****** G
                    002 HANXIT= ****** G        INTSTK= ****** G
```

Symbol table

```
INVEC = ****** G        LOKVEC= ****** G        MPAR16= ****** G        ODTSTA  000234R     002 PMPAR = ****** G
IOMAP = ****** G        LOMAP = ****** G        MPIVSZ= 000032          ODTTOP  000036R     002 PNAME = ****** G
IOPAGE= ****** G        LOTBUF= ****** G        MPPHY = ****** G        ODTTRP= ****** G        PPTERM  000262RG    002
IOQSIZ= ****** G        LOTEND= ****** G        MR$$SZ= ****** G        OORCLO  022710R     002 PROASM= 000001 G
IOTIMR= ****** G        LOTSIZ= ****** G        MSGBAS= ****** G        OORCSH  022730R     002 PROBUF  000150RG    002
JCXPGS= ****** G        LOUTIR= ****** G        MTSXDV  024544R     002 OORDBG  022720R     002 PROCID= 000001
JMP0  = ****** G        LOWEND  000754R     002 MUXVEC  005260R     002 OORDMP  022740R     002 PROFLG= ****** G
JM$$SZ= ****** G        LOWOVL  000746R     002 MU$TXT= ****** G        OORLOK  022522R     002 PROHAN= ****** G
JSWLOC= ****** G        LSTDL = ****** G        MVSIZ = ****** G        OORMIO  022700R     002 PROINI= ****** G
KEYSEG  004522R     002 LSTHL = ****** G        MW$LNK= ****** G        OORMSG  022544R     002 PROITP  026400RG    002
KMNBAS= ****** G        LSTLIN= ****** G        MW$$SZ= ****** G        OORPLS  022620R     002 PROLIN= ****** G
KMNCHN= ****** G        LSTMX = ****** G        MXCSR = ****** G        OORSLE  022652R     002 PRONOP= ****** G
KMNHI = ****** G        LSTPL = ****** G        MXDTR = ****** G        OORSPL  022452R     002 PROSIZ= ****** G
KMNNAM  000072R     002 LSTSL = ****** G        MXJADR= ****** G        OORSP2  022512R     002 PRTDEC  025230R     002
KMNPGS= ****** G        LSWPBK= ****** G        MXJMEM= ****** G        OORSWP  022642R     002 PRTOCT  025160R     002
KMNSTK= ****** G        LSW10 = ****** G        MXLNT = ****** G        OORUSR  022436R     002 PRTR50  025274R     002
KMNSTR= ****** G        LSW3  = ****** G        MXLPR = ****** G        OORWIN  022662R     002 PSW   = ****** G
KMNTOP= ****** G        LSW5  = ****** G        MXRBUF= ****** G        OOXFIN  023010R     002 PTBYT = ****** G
KPAR0 = ****** G        LSW6  = ****** G        MXTYPE= ****** G        OOXNO   022750R     002 PTWRD = ****** G
KPAR5 = ****** G        LTTPAR= ****** G        MXVEC = ****** G        OOXYES  022756R     002 PVSPBL= ****** G
KPAR6 = ****** G        LXCL  = ****** G        NDL   = ****** G        OPNCHN  024322R     002 QBUS  = ****** G
KPAR7 = ****** G        MAPALC  016006R     002 NDVRCB= ****** G        OPNKMN  013142R     002 RBD   = 000062
KPDR0 = ****** G        MAPPAR= ****** G        NEDCHR= ****** G        OPNRSF  010216R     002 RBR   = ****** G
LCDTYP= ****** G        MAPSIZ= ****** G        NEXMSG  002223R     002 OPNSWP  007564R     002 RC$$SZ= ****** G
LCLUNT= ****** G        MAPSYS= ****** G        NFRESB= ****** G        OPTOVL  022274R     002 RDB   = ****** G
LDDEVX= ****** G        MAXDEV= ****** G        NFSBLK  000020R     002 OSEND   000744R     002 RDBEND= ****** G
LDHAND  016430R     002 MAXOVL= 000031          NIOL  = ****** G        OSLAST  000744R     002 RDERR   003066R     002
LDHB1B= ****** G        MAXSLO= ****** G        NLINES= ****** G        OSTABL  000516R     002 RDINT = ****** G
LDHB1P= ****** G        MA$SYS= ****** G        NMSNMB= ****** G        OSZ   = 000056          REDUCE  002551R     002
LDHB2B= ****** G        MB$$SZ= ****** G        NMXHAN  000124R     002 OS$FLG= 000002          RELOC = ****** G
LDHNHI  017672R     002 MEMINI  015230R     002 NOCCL   001452R     002 OS$OVL= 000004          REQMIS  001070R     002
LDHNLO  017504R     002 MEMLIM  000236R     002 NOCLOK  002077R     002 OS$SIZ= 000000          RID   = 000060
LDREAD  021072R     002 MEMPAR= ****** G        NOCSRR  001670R     002 OS$$SZ= 000006          RLBF    000230R     002
LDVERS= ****** G        MEMTST  015334R     002 NOKMON  001411R     002 OTMXV = ****** G        RLBFND  000232R     002
LHIRBA= ****** G        MEM256= ****** G        NOODT   002272R     002 OTRECV= ****** G        RMNPDR= ****** G
LHIRBB= ****** G        MF$CM = ****** G        NOSYDV  002442R     002 OVLBAS  000140R     002 RMON  = ****** G
LHIRBC= ****** G        MF$CS = ****** G        NOTLIC  025556R     002 OVLBLD  021610R     002 ROMBUF  025722R     002
LHIRBE= ****** G        MF$LE = ****** G        NOXM    015626R     002 OVLEND  022436R     002 RPRVEC= ****** G
LHIRBG= ****** G        MF$LIN= ****** G        NSCP  = ****** G        OVLLST  022346R     002 RSFBLK= ****** G
LHIRBP= ****** G        MHNSIZ= ****** G        NSIP  = ****** G        OVLPOS  021400R     002 RSFERR  001175R     002
LHIRBS= ****** G        MH$LPR= ****** G        NSL   = ****** G        OVLTRY  023020R     002 RSR   = ****** G
LICNUM  025720R     002 MH$PBR= ****** G        NSPLBL= ****** G        OVRADD= ****** G        RSZ   = 000052
LINBUF= ****** G        MH$SCR= ****** G        NSPLDV= ****** G        O. ADR = ****** G       RTFTCH  024770R     002
LINCHK  005260R     002 MINCTR= ****** G        NSPLFL= ****** G        O. BLK = ****** G       RTMNVC  000046R     002
LINEND= ****** G        MIOBHD= ****** G        NUMCCB= ****** G        O. PAR = ****** G       RTNKM   015616R     002
LININI  005262R     002 MIODBG= ****** G        NUMCDB= ****** G        O. SIZ = ****** G       RTTRP4  000044R     002
LINIR = ****** G        MIOFLG= ****** G        NUMDCD= ****** G        PARENL= ****** G        RTVDEF  000314R     002
LINSIZ= ****** G        MIONWB= ****** G        NUMDEV= ****** G        PARSET  016316R     002 RTVEND  000322R     002
LINSPC= ****** G        MIOWHD= ****** G        NUMFRK= ****** G        PHSOVF  002612R     002 RTVER   000264R     002
LINTYP  005340R     002 MI$LNK= ****** G        NUMIOQ= ****** G        PHYMEM= ****** G        RTVPTR  000156R     002
LMXLN = ****** G        MI$SBP= ****** G        NUMRDB= ****** G        PIDPTR= ****** G        RTV$SZ= 000003
LMXNUM= ****** G        MI$$SZ= ****** G        NXIVMH= ****** G        PIDRIV= ****** G        RT$BAS= ****** G
LODINI  004160R     002 MMENBL= ****** G        NXMMSG  002165R     002 PIDVEN  005200R     002 RT$NAM= ****** G
LODOVL  023162R     002 MONFQH= ****** G        ODTBAS= 001000          PIHAN = ****** G        RT$SKP= ****** G
LOKBAS= ****** G        MONVEC= ****** G        ODTBLK  000210R     002 PIINSZ= 001274          RT$TOP= ****** G
LOKCSH= ****** G        MPARFL= ****** G        ODTFLG  000034R     002 PISRT   000244RG    002 RT$UPD= 000001
LOKMEM= ****** G        MPAR0 = ****** G        ODTRDM  002401R     002 PMCELS= ****** G        RT$VER= 000000
```

Symbol table

```
RT$$SZ= ****** G
RT40    000264R
RT50    000267R
RT51    000272R
RT51B   000300R
RT51C   000303R
RT51X   000275R
RT52    000306R
RT53    000311R
RT54    000314R
RT55    000317R
R50CHR  003347R
R50CL = 012240
R50CL0= 012276
R50CL7= 012305
R50CSH= 012700
R50C1 = 013630
R50C10= 013666
R50C17= 013675
R50LD = 045640
R50LOK= 046543
R50LS = 046770
R50MSG= 052077
R50ODT= 057164
R50PI = 062550
R50SY = 075250
R50TIO= 077167
R50TT = 100040
R50USR= 103112
R50VM = 105610
R50WIN= 110466
SAVBLK  000050R
SB$$SZ= ****** G
SCHED = ****** G
SCPFHD= ****** G
SDANAM= ****** G
SDCB  = ****** G
SDCBSZ= ****** G
SDCHAN= ****** G
SDDVU = ****** G
SDNAME= ****** G
SEGCHN= ****** G
SETCHN  024410R
SETJSZ  016172R
SETMIO  021376R
    SETSY   024600R
002 SETUMP  004650R
002 SETVEC  005024R
002 SFCB  = ****** G
002 SFCBFH= ****** G
002 SFCBND= ****** G
002 SFCBSZ= ****** G
002 SG$ELG= ****** G
002 SG$IOT= ****** G
002 SG$MMU= ****** G
002 SG$MTM= ****** G
002 SG$MTS= ****** G
    SG$PAR= ****** G
    SG$TSX= ****** G
    SHRRCB= ****** G
    SHRRCN= ****** G
    SIZERR  025124R
    SKPDEV  000160R
    SLTSIZ= ****** G
    SMRSIZ= ****** G
    SNMSHD= ****** G
    SPLANM= ****** G
    SPLBLK= ****** G
    SPLCHN= ****** G
    SPLCLD  011306R
    SPLDEV= ****** G
    SPLDVN= ****** G
    SPLINI  010522R
    SPLNB = ****** G
    SPLND = ****** G
    SPNEED  010142R
002 SP$$SZ= ****** G
    SRTOVF  003121R
    SRTSIZ= ****** G
    SROMMR= ****** G
    SR3FLG= ****** G
    SR3MMR= ****** G
    SS    = ****** G
    SSEND = ****** G
    STA   = 000040
    STDVTB  017374R
    STHNPV  020400R
002 STK   = 000042
002 STKLVL= ****** G
002 SVERR   002745R
002 SWDBLK= ****** G
002 SWPCHN= ****** G
002 SWPJOB= ****** G
    SWPPOS= ****** G
    SYINDX= ****** G
    SYNAME= ****** G
    SYSDAT= ****** G
    SYSGEN= ****** G
    SYSMAP= ****** G
    SYSUPD= ****** G
    SYSVER= ****** G
    SYTIMH= ****** G
    SYTIML= ****** G
    SYUNIT= ****** G
    TAKOVR  003420R
    TIOBAS= ****** G
002 TIOVEC= ****** G
002 TK1SEC= ****** G
    TK1VAL= ****** G
    TK3SVL= ****** G
    TK5VAL= ****** G
    TOOBIG  002507R
    TOPMEM  000132R
    TRCSET  015606R
002 TRP10 = ****** G
    TRP14 = ****** G
    TRP20 = ****** G
002 TRP24 = ****** G
    TRP250= ****** G
    TRP34 = ****** G
002 TRP4  = ****** G
    TSEMT = ****** G
002 TSEXEC= ****** G
    TSGEN = ****** G
    TSINIT  000000RG
    TSR   = ****** G
    TSTWRD= 000110
    TSXHD   000754R
    TSXRUN  002000R
    TSXSAV  000062R
002 TSXSIT= ****** G
002 TSXSIZ  003000R
    TTINCP= ****** G
    UBUSMP= ****** G
002
    UCLBLK= ****** G
    UCLDAT= ****** G
    UCLINI  014652R
    UCLNAM= ****** G
    UCLOPN  003312R
    UC$$SZ= ****** G
    UDDRO = ****** G
    UEXINT= ****** G
    UEXRTN= ****** G
    UK$$SZ= ****** G
    UMODE = ****** G
    UMRADR= ****** G
    UMSYTP= ****** G
    UNIBUS= ****** G
002 UPAR0 = ****** G
    UPAR6 = ****** G
    UPAR7 = ****** G
    UPDR0 = ****** G
    UPDR6 = ****** G
    UPMODE= ****** G
    USRBAS= ****** G
002 US$$SZ= ****** G
002 VBUSTP= ****** G
002 VCSHNB= ****** G
    VDBFLG= ****** G
    VDFMEM= ****** G
    VF$LIN= ****** G
    VF$MR = ****** G
    VF$RE = ****** G
    VF$RIE= ****** G
    VF$SC = ****** G
    VF$TIE= ****** G
    VHIMEM= ****** G
    VHINIT  005260R
002 VHLPRM  005260R
    VHOINT= ****** G
    VH$CSR= ****** G
002 VH$LCR= ****** G
002 VH$LPR= ****** G
002 VINABT= ****** G
    VLDSYS= ****** G
002 VMAXMC= ****** G
    VMIOBF= ****** G
    VMIOSZ= ****** G
    VMLBLK= ****** G
    VMSCHR= ****** G
002 VMXCSH= ****** G
    VMXMON= ****** G
002 VMXMRB= ****** G
    VMXMSG= ****** G
    VMXSF = ****** G
    VMXSFC= ****** G
    VMXWIN= ****** G
    VNCSLO= ****** G
    VNCXOF= ****** G
    VNCXON= ****** G
    VNFCSH= ****** G
    VNGR  = ****** G
    VNUIP = ****** G
    VNUMDC= ****** G
    VPAR5 = ****** G
    VPAR6 = ****** G
    VPLAS = ****** G
    VPMSIZ= ****** G
    VSLEDT= ****** G
    VSWPFL= ****** G
    VSWPSL= ****** G
    VSYDMP= ****** G
    VUCLMC= ****** G
    VUXIFL= ****** G
    VU$CL = ****** G
    WINBAS= ****** G
    WRKBUF  000152R     002
    WRKSIZ  000154R     002
    XMVBAS  000122R     002
    ZCLR  = ****** G
    $DEAD = ****** G
002 $FORM = ****** G
002 $HARD = ****** G
    $MEMSZ= ****** G
    $NOIN = ****** G
    $OVRH = ****** G
    $PHONE= ****** G
    $SXON = ****** G
    $TAB  = ****** G
    $8BIT = ****** G
    . . . V1 = 000003
    . . . V2 = 000027

. ABS.   000000    000   (RW, I, GBL, ABS, OVR)
         000000    001   (RW, I, LCL, REL, CON)
TSINIT   026400    002   (RW, I, GBL, REL, OVR)
TSXEND   000000    003   (RW, I, GBL, REL, OVR)
Errors detected:   0

*** Assembler statistics


Work  file  reads:  0
Work  file  writes:  0
```

Symbol table

Size of work file: 12558 Words (50 Pages)
Size of core pool: 18432 Words (72 Pages)
Operating system: RT-11

Elapsed time: 00:02:30.52
,LP:PROINI=DK:PROASM,TSINIT