Table of contents

Table of contents

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

```
                                        .GLOBL  DSKBUF,PROBUF,FNDHRB,HANXMR
                                ;
                                ;  Following global only needed for the Pro distribution
                                ;  creation program - MAKPRO and installation program - INSTSX
                                ;
                                .IF     NE,PROASM
                                ;
                                ;** Assemble this code if we are generating for a Pro
                                ;
                                        .GLOBL  PROSIZ,PROINI,PROLIN,PROHAN,PRONOP
                                        .GLOBL  PIHAN,PIDPTR,PIDRIV
                                .IFF    ;NE,PROASM
                                ;
                                ;** Assemble this code if we are not generating for a Pro
                                ;
                                        .GLOBL  TSXPRO
000000                          TSXPRO  =       0
                                ;
                                ;** End of conditional Pro code
                                ;
                                .ENDC   ;NE,PROASM

                                ;
                                ;  Global references
                                ;
                                        .GLOBL  HANDSK,MAXDEV,NDVRCB,HANRCB,HANRCO
                                        .GLOBL  LXCL,VSYDMP,STKLVL,INTSSZ,INDFIL,NXIVMH,EXCBUF
                                        .GLOBL  VNUIP,NSIP,INSTBL,INSTBN,II$$SZ,DCCSIZ,VNUMDC,NUMCDB
                                        .GLOBL  NSCP,SCPFHD,SP$$SZ,CSHDEV,CSHDVN,VMXCSH,CD$$SZ
                                        .GLOBL  LSTPL,LMXNUM,MXCSR,MXVEC,RSR,INVEC,VHIMEM,CXTPAG
                                        .GLOBL  LSWPBK,LSTSL,SWDBLK,SWPCHN,NUMDEV,CS$NMX,SCHED
                                        .GLOBL  H.DSTS,DVSTAT,HANENT,H.GEN,FORK,INTEN,PNAME
                                        .GLOBL  $SXON,LSW10,LHIRBB,LHIRBE,LHIRBP,LHIRBG,LHIRBA
                                        .GLOBL  LHIRBS,LHIRBC,VNCSLO,VNCXOF,VNCXON,SDDVU,VMSCHR,MAXSLO
                                        .GLOBL  MPAR0,MPAR16,PARENL,MPARFL,TSEMT,VDBFLG,DX$EBA
                                        .GLOBL  H.SIZ,HANSIZ,H.DVSZ,DEVSIZ,LOMAP,MMENBL,UPAR7
                                        .GLOBL  PSW,HIMAP,FSTDL,LSTDL,LINBUF,LINSIZ,NUMCCB,TK1SEC
                                        .GLOBL  FRKINI,FRKGEN,NUMFRK,FQ$$SZ,H.CSR,H.INS,VSWPSL,DMYDEV
                                        .GLOBL  LINEND,LOTBUF,LOTSIZ,LOTEND,KMNTOP,KMNHI,NSL,NDL
                                        .GLOBL  DX$MPH,DX$NHM,DX$IBH,HANPAR,HANXIT,MAPPAR,LINSPC
                                        .GLOBL  KMNPGS,KMNSTK,KMNSTR,KMNCHN,SROMMR,KPAR0,PROFLG
                                        .GLOBL  EMMAP,IOMAP,SR3MMR,IOPAGE,MAPSIZ,SR3FLG,NSPLDV
                                        .GLOBL  UPAR0,KPDR0,UPDR0,KPAR7,BASMAP,PTWRD,PTBYT,LOKMEM
                                        .GLOBL  GTBYT,MPPHY,RELOC,BRKPT,TSGEN,TSEXEC,VSWPFL
                                        .GLOBL  CW$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$CL,CLSTS
                        .GLOBL  FREPQS,IOQSIZ,SYUNIT,UMSYTP,DI$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$$SZ,PMSIZE,PMPAR
                        .GLOBL  NUMDCD,MEM256,LOKCSH,DC$$SZ
                        .GLOBL  JCXPQS,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.
                ;
```

```
                                STA     =       40              ;PROGRAM START ADDRESS
                                STK     =       42              ;INITIAL STACK POINTER
                                RSZ     =       52              ;ROOT SIZE
                                OSZ     =       56              ;OVERLAY SIZE
                                RID     =       60              ;REL FILE ID
                                RBD     =       62              ;DISPLACEMENT TO 1ST REL BLOCK
                                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
030066'                         WRKBUF: .WORD   INITOP
004000                          WRKSIZ: .WORD   2048.
052077                          R50MSG: .RAD50  /MSG/
110466                          R50WIN: .RAD50  /WIN/
046543                          R50LOK: .RAD50  /LOK/
103112                          R50USR: .RAD50  /USR/
012700                          R50CSH: .RAD50  /CSH/
077167                          R50TIO: .RAD50  /TIO/
100040                          R50TT:  .RAD50  /TT /
075250                          R50SY:  .RAD50  /SY /
045640                          R50LD:  .RAD50  /LD /
062550                          R50PI:  .RAD50  /PI /
012240                          R50CL:  .RAD50  /CL /
012276                          R50CL0: .RAD50  /CL0/
012305                          R50CL7: .RAD50  /CL7/
013630                          R50C1:  .RAD50  /C1/
013666                          R50C10: .RAD50  /C10/
```

```
                                R50C17: .RAD50  /C17/
                                R50VM:  .RAD50  /VM /
                                R50LS:  .RAD50  /LS /
                                R50ODT: .RAD50  /ODT/
000224  100040  015270  075250 SKPDEV: .RAD50  /TT DK SY CL C1 PI    /
000232  012240  013630  062550
000240  000000
000242     000     110         GTLIN:  .BYTE   0,110
000244  075250  114730  000000 HANNAM: .RAD50  /SY XXX   TSX/
000252  100020
000254  075250  075273  057164 ODTBLK: .RAD50  /SY SYSODTREL/
000262  070524
000264  075250  035164  000000 INDNAM: .RAD50  /SY IND   SAV/
000272  073376
000274  000000                 RLBF:   .WORD   0
000276  000000                 RLBFND: .WORD   0
000300  000000                 ODTSTA: .WORD   0
000302  000000                 MEMLIM: .WORD   0
000304  000000                 HGENFL: .WORD   0
                               ;
                               ;  Initialization configuration word
                               ;
000306  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
                               ;
000310  075250  062550  000000 PISRT:  .RAD50  /SY PI    TSX/
000316  100020
000320  000000  000000  000000         .WORD   0,0,0
                               ;
                               ;  Byte data cells
                               ;
000326     000                 PPTERM: .BYTE   0               ;1 if printer port is T/S terminal
                                       .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 = Minimum acceptable RT-11 version.
                                ;   Arg 3 = Minimum acceptable RT-11 update within the version.
                                ;
                                        .MACRO  HANVER  DEVID,RTVERS,RTUPDT
                                        .BYTE   DEVID           ;ID code for device type
                                        .BYTE   RTVERS          ;Minimum RT-11 version
                                        .BYTE   RTUPDT          ;Minimum update level within version
                                        .ENDM   HANVER
                                ;
                                ;  Define offsets into handler version table
                                ;
000000                          HV$ID   =       0               ;Handler identification code
000001                          HV$VER  =       1               ;Minimum RT-11 version
000002                          HV$UPD  =       2               ;Minimum update level within version
000003                          HV$$SZ  =       3               ;Size of handler version table entry
                                ;
                                ;  Define minimum versions for various handlers
                                ;  ********************************************************
                                ;  *** Note RT-11 version update numbering system changed ***
                                ;  *** at 5.2 so that the update number for 5.2 is lower  ***
                                ;  *** than for 5.1C. See the CLVX entries below for      ***
                                ;  *** version and update correlations.                   ***
                                ;  ********************************************************
000330                          HVTBL:
000330                                  HANVER  DI$DU,5.,0.     ;DU - 5/0 (5.0)
000333                                  HANVER  DI$XL,5.,2.     ;XL - 5/6 (5.1B) ;Fixed for 5.2 SCB
000336                                  HANVER  DI$MU,5.,4.     ;MU - 5/4 (5.4)
000341                          HVEND:
                                        .EVEN
```

```
                                ;-----------------------------------------------------------------------
                                ;  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
                                ;
000342                          DVFLBS:
000342                                  DEFFLG  <CR>,<DX$MPH>
000346                                  DEFFLG  <CT>,<DX$MPH>
000352                                  DEFFLG  <DB>,<DX$DMA!DX$MPH>
000356                                  DEFFLG  <DD>,<DX$NHM>
000362                                  DEFFLG  <DL>,<DX$DMA!DX$MPH!DX$IBH>
000366                                  DEFFLG  <DM>,<DX$DMA!DX$NHM>
000372                                  DEFFLG  <DP>,<DX$DMA>
000376                                  DEFFLG  <DS>,<DX$DMA>
000402                                  DEFFLG  <DT>,<DX$DMA>
000406                                  DEFFLG  <DU>,<DX$DMA!DX$NHM!DX$NST>
000412                                  DEFFLG  <DW>,<DX$MPH>
000416                                  DEFFLG  <DX>,<DX$MPH>
000422                                  DEFFLG  <DY>,<DX$DMA!DX$NHM>
000426                                  DEFFLG  <DZ>,<DX$MPH>
000432                                  DEFFLG  <FW>,<DX$DMA>
000436                                  DEFFLG  <LP>,<DX$MPH>
000442                                  DEFFLG  <LS>,<DX$MPH>
000446                                  DEFFLG  <MM>,<DX$DMA!DX$MPH!DX$IBH>
000452                                  DEFFLG  <MS>,<DX$DMA!DX$MPH!DX$IBH>
000456                                  DEFFLG  <MT>,<DX$DMA!DX$MPH!DX$IBH>
000462                                  DEFFLG  <MU>,<DX$DMA!DX$NHM!DX$IBH!DX$NST>
000466                                  DEFFLG  <NL>,<DX$MPH>
000472                                  DEFFLG  <PC>,<DX$MPH>
000476                                  DEFFLG  <RF>,<DX$DMA>
000502                                  DEFFLG  <RK>,<DX$DMA!DX$MPH>
000506                                  DEFFLG  <VM>,<DX$EBA!DX$NCA!DX$NHM>
000512                                  DEFFLG  <XC>,<DX$MPH>
000516                                  DEFFLG  <XL>,<DX$MPH>
000522                          DVFLND:

                                ;-----------------------------------------------------------------------
                                ;  The following table specifies which version number is to be
                                ;  returned by CL in response to the XL/CL .SPFUN used by
                                ;  VTCOM to match it to the handler.
                                ;  The macro has 3 arguments:
                                ;    1. Minimum RT-11 version number where this CL version should be used.
                                ;    2. Minimum RT-11 update number where this CL version should be used.
                                ;    3. CL version number that starts with specified RT-11 version.
                                ;
                                    .MACRO  CLVX    RTV,RTU,CLV
                                    .BYTE   RTU,RTV
                                    .WORD   CLV
                                    .ENDM   CLVX
                                ;
```

```
                                ;  Define CL versions based on RT-11 versions
                                ;
000522                          CLVTBL:
000522                                  CLVX    5,1,16.         ;Version 5.1
000526                                  CLVX    5,6,16.         ;Version 5.1B
000532                                  CLVX    5,35,16.        ;Version 5.1
000536                                  CLVX    5,44,16.        ;Version 5.1C
000542                                  CLVX    5,2,17.         ;Version 5.2
000546                                  CLVX    5,3,17.         ;Version 5.3
000552                                  CLVX    5,4,18.         ;Version 5.4
000556                          CLVEND:
                                ;
                                ;-------------------------------------------------------------------
                                ;  RT-11 v5.4 changed the structure of the LD translation tables.
                                ;  The following table is used to determine which table format
                                ;  to return to LD spfun 372.
                                ;
                                ;  Use original format for the following versions:
                                ;
000556                          LD1TBL:
000556  005     000                     .BYTE   5,0             ;Version 5.0
000560  005     001                     .BYTE   5,1             ;Version 5.1
000562  005     006                     .BYTE   5,6             ;Version 5.1B
000564  005     035                     .BYTE   5,35            ;Version 5.1
000566  005     044                     .BYTE   5,44            ;Version 5.1C
000570  005     002                     .BYTE   5,2             ;Version 5.2
000572  005     003                     .BYTE   5,3             ;Version 5.3
000574  000000                          .WORD   0
                                ;
                                ;-------------------------------------------------------------------
                                ;  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$$SZ  =       6               ;Size of each overlay entry
                                ;
        000031                  MAXOVL  =       25.             ;Maximum number of system overlays
                                ;
000576                          OSTABL: .BLKB   OS$$SZ*MAXOVL   ;Reserve room for table
001024                          OSEND:                          ;-Define end of table
001024  000576'                 OSLAST: .WORD   OSTABL          ;Pointer past last used entry in table
                                ;
                                ;  Table of system overlays that must be loaded over TSINIT
                                ;
001026  012700                  LOWOVL: .RAD50  /CSH/           ;TSCASH
001030  077167                          .RAD50  /TIO/           ;TSTIO
001032  046543                          .RAD50  /LOK/           ;TSLOCK
001034                          LOWEND:                         ;End of table
```

```
                                ;-------------------------------------------------------------------------
                                ;  Text messages
                                ;
                                        .NLIST  BEX
001034  077     124     123     TSXHD:  .ASCII  /?TSX-F-/<200>
001044  111     156     166     BADLIN: .ASCII  'Invalid status register address for T/S line: '<200>
001123  111     156     166     BDVMSG: .ASCII  'Invalid interrupt vector address for T/S line: '<200>
001203  114     151     156     BDLMSG: .ASCII  /Line # = /<200>
001215  000                     CRLF:   .BYTE   0
001216  124     123     130     REQMIS: .ASCII  /TSX generation did not include device /<200>
001265  103     141     156     BADOPN: .ASCIZ  /Cannot open program swap file/
001323  103     141     156     RSFERR: .ASCIZ  /Cannot open PLAS region swap file/
001365  116     165     155     CONSPC: .ASCII  /Number of contiguous blocks needed = /<200>
001433  103     141     156     BDSPOP: .ASCII  /Cannot open spooled device: /<200>
001470  111     156     163     BOSF:   .ASCIZ  /Insufficient disk space for spool file/
001537  103     141     156     NOKMON: .ASCIZ  /Cannot find "SY:TSKMON.SAV" file/
001600  103     141     156     NOCCL:  .ASCIZ  /Cannot find "SY:CCL.SAV" file/
001636  103     141     156     CFHMSG: .ASCII  /Cannot find device handler file: /<200>
001700  105     162     162     ERHMSG: .ASCII  /Error reading device handler file: /<200>
001744  111     156     166     ERHNDV: .ASCII  /Invalid RT-11 version for device handler:/<200>
002016  124     123     130     TSXRUN: .ASCIZ  /TSX is already running/
002045  110     141     156     HSGER:  .ASCII  /Handler not generated with extended memory support: /<200>
002132  103     157     155     NOCLOK: .ASCIZ  /Computer line time clock (50 or 60 Hz) is not working/
002220  123     171     163     NXMMSG: .ASCIZ  /System is not equipped with memory management hardware/
002307  123     171     163     NEXMSG: .ASCIZ  /System is not equipped with extended memory management hardware/
002407  103     141     156     NOODT:  .ASCIZ  /Cannot locate "SY:SYSODT.REL" file/
002452  115     141     160     HN2BIG: .ASCII  /Mapped handler is larger than 8KB: /<200>
002516  105     162     162     ODTRDM: .ASCIZ  /Error on read of SYSODT rel file/
002557  110     141     156     NOSYDV: .ASCIZ  /Handler for SY device was not loaded/
002624  107     145     156     TOOBIG: .ASCIZ  /Generated TSX system is too large/
002666  122     145     144     REDUCE: .ASCII  /Reduce size of TSGEN by /<200>
002717  056     040     142     BYTES:  .ASCIZ  /. bytes/
002727  111     156     163     PHSOVF: .ASCIZ  /Insufficient total physical memory for generated system/
003017  103     141     156     COSRT:  .ASCII  /Cannot open shared run-time file: /<200>
003062  103     141     156     SVERR:  .ASCIZ  /Cannot locate "SY:TSX.SAV"/
003115  111     156     163     TSXSIZ: .ASCIZ  /Insufficient memory to load all mapped system regions/
003203  105     162     162     RDERR:  .ASCIZ  /Error reading "SY:TSX.SAV"/
003236  111     156     163     SRTOVF: .ASCIZ  /Insufficient memory to load all shared run-time systems/
003326  111     156     163     CSHOVF: .ASCIZ  /Insufficient memory space for data cache/
003377  103     141     156     INDOPN: .ASCIZ  /Cannot open TSXIND file/
003427  103     141     156     UCLOPN: .ASCIZ  /Cannot open TSXUCL data file/
003464  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.
                                ;
003534                          TAKOVR:
                                ;
                                ;  Read in system overlays that go over TSINIT
                                ;
003534  004737  004374'                 CALL    INIOVL          ;Read overlays over TSINIT
                                ;
                                ;  Set pointer to monitor offset vector
                                ;
003540  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.
                                ;
003546  012777  123456  000000G         MOV     #123456,@INTSND ;Say stack has not overflowed
003554  105037  000000G                 CLRB    STKLVL          ;Say we are already on interrupt stack
                                ;
                                ;  If we are running on a Professional, disable its interrupts
                                ;
                                .IF     NE,PROASM
                                        TSTB    PROFLG          ;Are we running on a PRO?
                                        BEQ     7$              ;Br if not on a PRO
                                        CALL    PRONOP          ;Disable its interrupts
                                        BR      5$              ;Ignore unexpected interrupts on PRO
                                .ENDC   ;NE,PROASM
                                ;
                                ;  Set up vectors to catch unexpected interrupts
                                ;  Note: We encode the interrupt vector address in the PS --
                                ;  the low-order two bits of the address are dropped (they are
                                ;  always zero) and the remainder of the address is encoded in the
                                ;  PS fields priority (high-order 3 bits) and n-z-v-c (low-order 4 bits).
                                ;
003560  012702  000000G         7$:     MOV     #UEXINT,R2      ;SEND UNEXPECTED INTERRUPTS TO THIS ROUTINE
003564  012700  000044                 MOV     #44,R0          ;120 ENCODED IN PS FIELDS
003570  105737  000000G                 TSTB    VUXIFL          ;ARE WE TO IGNORE UNEXPECTED INTERRUPTS?
003574  001004                         BNE     10$             ;BR IF NOT
003576  012702  000000G         5$:     MOV     #UEXRTN,R2      ;ROUTINE TO GO TO TO IGNORE INTERRUPT
003602  012700  000340                 MOV     #340,R0         ;SET PRIO=7 IN PS
003606  012701  000120         10$:    MOV     #120,R1         ;INIT ALL VECTORS STARTING AT 120
003612  010221                 1$:     MOV     R2,(R1)+        ;SET PC FOR INTERRUPT
003614  010021                         MOV     R0,(R1)+        ;SET PS FOR INTERRUPT (ENCODED ADDRESS VALUE)
003616  105737  000000G         6$:     TSTB    VUXIFL          ;ARE WE TO IGNORE UNEXPECTED INTS?
003622  001411                         BEQ     2$              ;BR IF YES
003624  105737  000000G                 TSTB    PROFLG          ;IS THIS A PRO?
003630  001006                         BNE     2$              ;BR IF YES
003632  005200                         INC     R0              ;ADVANCE ENCODED ADDRESS
003634  032700  000020                 BIT     #20,R0          ;DID WE CARRY INTO "T"-FIELD?
003640  001402                         BEQ     2$              ;BR IF NOT
```

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

```
     58 003642  062700  000020                ADD     #20,R0          ;FORCE CARRY OUT OF T-FIELD AND INTO PRIO FIELD
     59 003646  020127  000420          2$:   CMP     R1,#420         ;DONE ALL INTERRUPT VECTORS OF INTEREST?
     60 003652  103757                        BLO     1$              ;BR IF NOT
     61 003654  010237  000060                MOV     R2,@#60         ;CATCH CONSOLE TERMINAL VECTOR TOO
     62 003660  012737  000014  000062        MOV     #14,@#62        ;ENCODED 60
     63 003666  010237  000064                MOV     R2,@#64
     64 003672  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 003700  012700  000340          11$:  MOV     #340,R0         ;PRIORITY 7 PS
     71 003704  012737  000000G 000000G       MOV     #CLKRTI,@#CLKVEC;Send clock interrupt to RTI instruct for now
     72 003712  010037  000002G               MOV     R0,@#CLKVEC+2
     73                                ;
     74                                ;  Take over traps, EMT, BPT, etc.
     75                                ;
     76 003716  005001                        CLR     R1              ;Start at location 0
     77 003720  012721  000137                MOV     #137,(R1)+      ;[JMP @#JMP0] ==> 0
     78 003724  012721  000000G               MOV     #JMP0,(R1)+     ;CATCH JUMPS TO LOCATION 0
     79 003730  012721  000000G               MOV     #TRP4,(R1)+     ;TRAP 4
     80 003734  005021                        CLR     (R1)+
     81 003736  012721  000000G               MOV     #TRP10,(R1)+    ;TRAP 10
     82 003742  005021                        CLR     (R1)+
     83 003744  012721  000000G               MOV     #TRP14,(R1)+    ;TRAP 14 (BREAKPOINTS)
     84 003750  010021                        MOV     R0,(R1)+
     85 003752  012721  000000G               MOV     #TRP20,(R1)+    ;IOT TRAP
     86 003756  005021                        CLR     (R1)+
     87 003760  012721  000000G               MOV     #TRP24,(R1)+    ;POWER FAIL
     88 003764  010021                        MOV     R0,(R1)+
     89 003766  012721  000000G               MOV     #EMTENT,(R1)+   ;EMT
     90 003772  005021                        CLR     (R1)+
     91 003774  012721  000000G               MOV     #TRP34,(R1)+    ;TRAP
     92 004000  005021                        CLR     (R1)+
     93 004002  012737  000000G 000114        MOV     #MEMPAR,@#114   ;MEMORY PARITY TRAP
     94 004010  010037  000116                MOV     R0,@#116
     95 004014  012737  000000G 000244        MOV     #FPTRAP,@#244   ;TRAP 244 -- FLOATING POINT TRAP
     96 004022  010037  000246                MOV     R0,@#246        ;Enter FPU trap at priority 7
     97 004026  012737  000000G 000250        MOV     #TRP250,@#250   ;TRAP 250 -- MEMORY MANAGEMENT TRAP
     98 004034  005037  000252                CLR     @#252
     99                                ;
    100                                ;  Initialize the system mapped region.
    101                                ;
    102 004040  010546                        MOV     R5,-(SP)        ;SAVE THE CURRENT CONTENTS OF R5
    103 004042  012705  000006                MOV     #6,R5           ;INITIALIZE TO THE FIRST REGION
    104 004046  004737  000000G               CALL    MAPSYS          ;CALL THE SYSTEM MAPPING ROUTINE
    105 004052  012605                        MOV     (SP)+,R5        ;RESTORE THE PREVIOUS CONTENTS OF R5
    106                               ;
    107                               ;  Set up Unibus mapping if needed
    108                               ;
    109 004054  004737  004732'               CALL    SETUMP          ;SET UP UNIBUS MAPPING
    110                               ;
    111                               ;  Initialize time-sharing lines.
    112                               ;
    113 004060  004737  005350'               CALL    LININI          ;INIT LINES & SET UP VECTORS
    114                               ;
```

```
                                        ;  Enable memory management
                                        ;  (The kernel-mode mapping registers are already set up)
                                        ;
004064  052737  000000G 000000G         BIS     #MMENBL,@#SROMMR;Turn on memory management
004072  105737  000000G                 TSTB    SR3FLG          ;Does machine have memory management reg 3?
004076  001415                         BEQ     4$              ;Br if register does not exist (no ext. mem.)
004100  023727  000000G 010000         CMP     PHYMEM,#4096.   ;Does machine have at least 256Kb phys memory?
004106  103411                         BLO     4$              ;Br if not
004110  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;Set extended memory on
004116  105737  000000G                 TSTB    MEM256          ;Will TSX-Plus use at least 256Kb?
004122  001403                         BEQ     4$              ;Br if not
004124  052737  000000G 000000G         BIS     #IOMAP,@#SR3MMR ;Turn on 22-bit memory management for I/O
                                        ;
                                        ;  Initialize the memory allocation table
                                        ;
004132  013737  000000G 000000G 4$:     MOV     MAPPAR,@#KPAR5  ;Map to memory allocation table
004140  013702  000000G                 MOV     LOMAP,R2        ;Point to 1st user-page entry
004144  105022                 8$:     CLRB    (R2)+           ;Say page is free
004146  020237  000000G                 CMP     R2,HIMAP        ;Done all user pages?
004152  103774                         BLO     8$              ;Loop if not
004154  112712  000000G                 MOVB    #MA$SYS,(R2)    ;Set flag marking start of system pages
                                        ;
                                        ;  Set up I/O device interrupt vectors.
                                        ;
004160  004737  005022'                 CALL    DEVVEC          ;SET UP DEVICE INTERRUPT VECTORS
                                        ;
                                        ;  If we are running on a Professional, initiailize the PI handler
                                        ;
                                .IF     NE,PROASM
                                        TSTB    PROFLG          ;Are we running on a Professional?
                                        BEQ     3$              ;Br if not
                                        CALL    PROHAN          ;Initialize the PI handler
                                        CALL    PIDVEN          ;Make device table entry for PI
                                .ENDC   ;NE,PROASM
                                        ;
                                        ;  Initialize interrupt stack area
                                        ;
004164  013702  000000G         3$:     MOV     INTSND,R2       ;Point to base of stack area
004170  012700  123456                  MOV     #123456,R0      ;Get initialization value
004174  010022                  12$:    MOV     R0,(R2)+        ;Initialize the interrupt stack area
004176  020237  000000G                 CMP     R2,INTSTK       ;Finished?
004202  103774                         BLO     12$             ;Loop if not
004204  112737  177777  000000G         MOVB    #-1,STKLVL      ;Say we are not running on interrupt stack
                                        ;
                                        ; Enter TSEXEC to complete initialization
                                        ;
004212  000137  000000G                 JMP     INIJMP          ;ENTER INITIALIZATION ROUTINE IN TSEXEC
                                        ;
                                        ;  Abort the initialization
                                        ;
004216  013737  000042' 000000G INISTP: MOV     CLK100,@#CLKVEC ;Restore RT-11 clock vector
004224  013737  000044' 000004          MOV     RTTRP4,@#4      ;Restore trap 4 vector
004232  013737  000046' 000000G         MOV     RTMNVC,@#RMON   ;Restore RT-11 monitor pointer
004240                          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.
                                ;
004242  010146                  LODINI: MOV     R1,-(SP)
004244  010346                          MOV     R3,-(SP)
004246  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get pointer to linker-built overlay entry
                                ;
004250  016201  000004                  MOV     OS$OVL(R2),R1   ;Get pointer to linker-built table
                                ;
                                ;  Determine how much code to read from the segment
                                ;
004254  016204  000000                  MOV     OS$SIZ(R2),R4   ;Get # 64-byte blks allocated for segment
004260  072427  000005                  ASH     #5,R4           ;Convert to # words
004264  020461  000000G                 CMP     R4,O.SIZ(R1)    ;Compare with original segment code size
004270  101402                          BLOS    1$              ;Br if segment was truncated by init
004272  016104  000000G                 MOV     O.SIZ(R1),R4    ;Get code size
                                ;
                                ;  Read the segment into memory
                                ;
004276  010503                  1$:     MOV     R5,R3           ;Get 64-byte block #
004300  072327  000006                  ASH     #6,R3           ;Convert to byte address
004304                                  .READW  #AREA,#17,R3,R4,O.BLK(R1)
004340  103406                          BCS     10$             ;Br if error on read
                                ;
                                ;  Store the physical address of the segment into the overlay descriptor
                                ;
004342  010561  000000G                 MOV     R5,O.PAR(R1)    ;Remember physical address of segment
                                ;
                                ;  Finished
                                ;
004346  012604                          MOV     (SP)+,R4
004350  012603                          MOV     (SP)+,R3
004352  012601                          MOV     (SP)+,R1
004354  000207                          RETURN
                                ;
                                ;  Error on read
                                ;
004356                          10$:    .PRINT  #TSXHD
004364                                  .PRINT  #RDERR
004372                                  .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.
                                ;
004374  010246                  INIOVL: MOV     R2,-(SP)
004376  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize pointer to start of memory area for overlays
                                ;
004400  013705  000140'                 MOV     OVLBAS,R5       ;Start of area for overlays
004404  072527  177772                  ASH     #-6,R5          ;Convert to 64-byte #
004410  042705  176000                  BIC     #176000,R5      ;Clear possible propagated sign bits
                                ;
                                ;  Begin loop to load each overlay that goes over TSINIT
                                ;
004414  012702  000576'                 MOV     #OSTABL,R2      ;Point to 1st overlay segment entry
004420  005762  000002          1$:     TST     OS$FLG(R2)      ;Does this segment go over TSINIT?
004424  001406                          BEQ     2$              ;Br if not
004426  004737  004604'                 CALL    KEYSEG          ;Remember base of some segments
004432  004737  004242'                 CALL    LODINI          ;Load the segment
004436  066205  000000                  ADD     OS$SIZ(R2),R5   ;Advance memory pointer
004442  062702  000006          2$:     ADD     #OS$$SZ,R2      ;Point to entry for next segment
004446  020237  001024'                 CMP     R2,OSLAST       ;Finished all segments?
004452  103762                          BLO     1$              ;Loop if not
                                ;
                                ;  Initialize entry point vector for TSTIOX segment
                                ;
004454  013702  000000G                 MOV     TIOBAS,R2       ;Get addr of base of TSTIOX
004460  072227  000006                  ASH     #6,R2           ;Convert to byte address
004464  012705  000000G                 MOV     #TIOVEC,R5      ;Point to entry point vector
004470  004737  004546'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Initialize entry point vector for TSCASH segment
                                ;
004474  013702  000000G                 MOV     CSHBAS,R2       ;Get addr of base of TSCASH
004500  001406                          BEQ     3$              ;Br if TSCASH not loaded
004502  072227  000006                  ASH     #6,R2           ;Convert to byte address
004506  012705  000000G                 MOV     #CSHVEC,R5      ;Point to entry point vector
004512  004737  004546'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Initialize entry point vector for TSLOCK segment
                                ;
004516  013702  000000G         3$:     MOV     LOKBAS,R2       ;Get addr of base of TSLOCK
004522  001406                          BEQ     9$              ;Br if TSLOCK not loaded
004524  072227  000006                  ASH     #6,R2           ;Convert to byte address
004530  012705  000000G                 MOV     #LOKVEC,R5      ;Point to entry point vector
004534  004737  004546'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Finished
                                ;
004540  012605                  9$:     MOV     (SP)+,R5
004542  012602                          MOV     (SP)+,R2
004544  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)
                                ;
004546  010246                  ENTVEC: MOV     R2,-(SP)
004550  010446                          MOV     R4,-(SP)
004552  010546                          MOV     R5,-(SP)
004554  010204                          MOV     R2,R4           ;Get addr of base of segment
004556  062704  000004                  ADD     #4,R4           ;Point to start of vector in segment
004562  005722                          TST     (R2)+           ;Get value to use to relocate offsets
004564  012415                  1$:     MOV     (R4)+,(R5)      ;Get offset to entry point within segment
004566  060225                          ADD     R2,(R5)+        ;Convert to absolute address
004570  005715                          TST     (R5)            ;Any more words to initialize?
004572  001774                          BEQ     1$              ;Br if yes
                                ;
                                ;  Finished
                                ;
004574  012605                          MOV     (SP)+,R5
004576  012604                          MOV     (SP)+,R4
004600  012602                          MOV     (SP)+,R2
004602  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.
                                ;
004604  010446                  KEYSEG: MOV     R4,-(SP)
                                ;
                                ;  Get the name of the segment out of the linker-built segment block
                                ;
004606  016200  000004                  MOV     OS$OVL(R2),R0   ;Point to linker-built entry
004612  016004  000000G                 MOV     O.ADR(R0),R4    ;Get the name of the segment
                                ;
                                ;  See if this is a segment whose address we want to remember
                                ;
004616  020437  000156'                 CMP     R4,R50MSG       ;Is this the TSMSG segment?
004622  001003                          BNE     1$              ;Br if not
004624  010537  000000G                 MOV     R5,MSGBAS       ;Remember base of TSMSG segment
004630  000436                          BR      8$
004632  020437  000160'          1$:     CMP     R4,R50WIN       ;Is this the TSWIN segment?
004636  001003                          BNE     3$              ;Br if not
004640  010537  000000G                 MOV     R5,WINBAS       ;Remember base of TSWIN segment
004644  000430                          BR      8$
004646  020437  000164'          3$:     CMP     R4,R50USR       ;Is this the TSUSR segment?
004652  001003                          BNE     4$              ;Br if not
004654  010537  000000G                 MOV     R5,USRBAS       ;Remember base of TSUSR segment
004660  000422                          BR      8$
004662  020437  000162'          4$:     CMP     R4,R50LOK       ;Is this the TSLOCK segment?
004666  001003                          BNE     5$              ;Br if not
004670  010537  000000G                 MOV     R5,LOKBAS       ;Remember base of TSLOCK segment
004674  000414                          BR      8$
004676  020437  000166'          5$:     CMP     R4,R50CSH       ;Is this the TSCASH segment?
004702  001003                          BNE     6$              ;Br if not
004704  010537  000000G                 MOV     R5,CSHBAS       ;Remember base of TSCASH segment
004710  000406                          BR      8$
004712  020437  000170'          6$:     CMP     R4,R50TIO       ;Is this the TSTIOX segment?
004716  001003                          BNE     8$              ;Br if not
004720  010537  000000G                 MOV     R5,TIOBAS       ;Remember base of module
004724  000400                          BR      8$
                                ;
                                ;  Finished
                                ;
004726  012604                  8$:     MOV     (SP)+,R4
004730  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.
                                ;
004732  010246                 SETUMP: MOV     R2,-(SP)
004734  010346                         MOV     R3,-(SP)
004736  013746  000004                 MOV     @#4,-(SP)       ;SAVE TRAP VECTOR
004742  012737  005010' 000004         MOV     #9$,@#4         ;CATCH TRAPS
                                ;
                                ;  See if this is a type of maching that needs unibus mapping
                                ;
004750  105737  000000G                 TSTB    UBUSMP          ;Is UNIBUS mapping needed?
004754  001415                         BEQ     9$              ;Br if not
                                ;
                                ;  Unibus mapping is needed
                                ;  Load unibus map registers # 0-4 to point to low 48Kb of memory.
                                ;
004756  012705  000000G          2$:    MOV     #UMRADR,R5      ;POINT TO CONTROL REGISTER FOR UNIBUS MAP 0
004762  005004                         CLR     R4              ;START MAPPING TO BOTTOM OF MEMORY
004764  012700  000005                 MOV     #5,R0           ;LOAD 5 MAP REGISTERS
004770  010425                 1$:     MOV     R4,(R5)+        ;SET LOW-ORDER VALUE IN MAP REGISTER
004772  005025                         CLR     (R5)+           ;CLEAR HIGH-ORDER VALUE IN MAP REGISTER
004774  062704  020000                 ADD     #8192.,R4       ;ADVANCE MEMORY ADDRESS
005000  077005                         SOB     R0,1$           ;LOOP IF MORE MAP REGISTERS TO LOAD
                                ;
                                ;  Turn on Unibus mapping
                                ;
005002  052737  000000G 000000G         BIS     #IOMAP,@#SR3MMR ;ENABLE UNIBUS MAPPING
                                ;
                                ;  Finished
                                ;
005010  012637  000004          9$:     MOV     (SP)+,@#4       ;RESTORE TRAP VECTOR
005014  012605                         MOV     (SP)+,R5
005016  012604                         MOV     (SP)+,R4
005020  000207                         RETURN
                                        .IFF    ;NE,<PROASM-1>  ;Following code for Pro-only assembly
                                ;
                                ;  Define dummy SETUMP routine for Pro
                                ;
                                SETUMP: RETURN
                                        .ENDC   ;NE,<PROASM-1>
```

```
                                        .SBTTL  DEVVEC -- Set up device vectors
                                ;-----------------------------------------------------------------------
                                ;  DEVVEC is called to set up device interrupt vectors for handlers
                                ;  that have been loaded.
                                ;
005022  010146                  DEVVEC: MOV     R1,-(SP)
005024  010346                          MOV     R3,-(SP)
005026  010546                          MOV     R5,-(SP)
005030  013746  000000G                 MOV     @#KPAR5,-(SP)   ;Save PAR 5 mapping
                                ;
                                ;  Begin loop to set up vectors for each device
                                ;
005034  012701  000002                  MOV     #2,R1           ;Get index # of 1st device after TT
005040  016103  000000G         1$:     MOV     HANENT(R1),R3   ;Get handler entry point address
005044  020327  000006                  CMP     R3,#6           ;Is this a real device?
005050  101436                          BLOS    6$              ;Br if not
                                ;
                                ;  See if we need to map PAR 5 to this handler
                                ;
005052  016100  000000G                 MOV     HANPAR(R1),R0   ;Get PAR 5 base for this handler
005056  001402                          BEQ     2$              ;Br if this is not a mapped handler
005060  010037  000000G                 MOV     R0,@#KPAR5      ;Map PAR 5 to the handler
                                ;
                                ;  Clear CQE and LQE in handler header
                                ;
005064  005023                  2$:     CLR     (R3)+           ;Clear LQE (4th word in handler)
005066  005013                          CLR     (R3)            ;Clear CQE (5th word in handler)
005070  162703  000010                  SUB     #10,R3          ;Point to 1st word of handler
                                ;
                                ;  Set up interrupt vectors for this handler
                                ;
005074  005005                          CLR     R5              ;Assume vector base address is 0
005076  005713                          TST     (R3)            ;Any vectors to set up?
005100  001422                          BEQ     6$              ;Br if no vectors to set up
005102  002403                          BLT     5$              ;Br if multiple-vector
005104  004737  005174'                 CALL    SETVEC          ;Set up the vector
005110  000416                          BR      6$              ;Finished
                                ;
                                ;  Multiple vectors.
                                ;
005112  012300                  5$:     MOV     (R3)+,R0        ;Get offset to list of vector info
005114  006300                          ASL     R0              ;Get byte offset to vector list
005116  060003                          ADD     R0,R3           ;Get absolute address of vector table
005120  005713                          TST     (R3)            ;Is this a PRO device with floating vectors?
005122  002005                          BGE     7$              ;Br if not
005124  005723                          TST     (R3)+           ;Point to word with device ID
005126  012346                          MOV     (R3)+,-(SP)     ;Get device ID
005130  004777  000000G                 CALL    @RPRVEC         ;Get base vector location for device
005134  012605                          MOV     (SP)+,R5        ;This is base of vector locations
005136  004737  005174'         7$:     CALL    SETVEC          ;Set up the vector
005142  005713                          TST     (R3)            ;Any more vectors to set up?
005144  003374                          BGT     7$              ;Br if yes
                                ;
                                ;  See if there are more devices to set up.
                                ;
005146  062701  000002          6$:     ADD     #2,R1           ;Advance device table index
005152  020137  000000G                 CMP     R1,NUMDEV       ;More to do?
```

```
     58 005156  101730                          BLOS    1$              ;Br if yes
     59                                ;
     60                                ;  Finished
     61                                ;
     62 005160  012637  000000G                 MOV     (SP)+,@#KPAR5
     63 005164  012605                          MOV     (SP)+,R5
     64 005166  012603                          MOV     (SP)+,R3
     65 005170  012601                          MOV     (SP)+,R1
     66 005172  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
                                ;
005174  010446                  SETVEC: MOV     R4,-(SP)
                                ;
                                ;  See if this is a mapped handler
                                ;
005176  005761  000000G                 TST     HANPAR(R1)      ;Is this a mapped handler
005202  001010                          BNE     1$              ;Br if yes
                                ;
                                ;  This is an unmapped handler.
                                ;  Vector interrupts directly to the handler.
                                ;
005204  012300                          MOV     (R3)+,R0        ;Get address of interrupt vector
005206  060500                          ADD     R5,R0           ;Add base address to vector location
005210  010310                          MOV     R3,(R0)         ;Store address of cell in handler
005212  062320                          ADD     (R3)+,(R0)+     ;Add offset to interrupt entry point
005214  012310                          MOV     (R3)+,(R0)      ;Set PS for interrupt
005216  052710  000340                  BIS     #340,(R0)       ;Make sure priority = 7
005222  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.
                                ;
005224  013704  000122'         1$:     MOV     XMVBAS,R4       ;Point to area where we store interrupt rtn
005230  012300                          MOV     (R3)+,R0        ;Get address of interrupt vector
005232  060500                          ADD     R5,R0           ;Add base address to interrupt location
005234  010420                          MOV     R4,(R0)+        ;Direct interrupt to our routine
005236  012724  013746                  MOV     #013746,(R4)+   ; [ MOV @#KPAR5,-(SP) ]
005242  012724  000000G                 MOV     #KPAR5,(R4)+
005246  012724  012737                  MOV     #012737,(R4)+   ; [ MOV #par5val,@#KPAR5 ]
005252  016124  000000G                 MOV     HANPAR(R1),(R4)+
005256  012724  000000G                 MOV     #KPAR5,(R4)+
005262  012724  012746                  MOV     #012746,(R4)+   ; [ MOV #340,-(SP) ]
005266  012724  000340                  MOV     #340,(R4)+
```

```
      58 005272  012724  012746                    MOV     #012746,(R4)+   ; [ MOV #HANXIT,-(SP) ]
      59 005276  012724  0000000                   MOV     #HANXIT,(R4)+
      60 005302  012724  000257                    MOV     #000257,(R4)+   ; [ CCC - Clear all condition codes ]
      61 005306  016314  000002                    MOV     2(R3),(R4)      ; [ SEx - Set condition codes specified in PS]
      62 005312  042714  177760                    BIC     #^C17,(R4)
      63 005316  052724  000260                    BIS     #260,(R4)+
      64 005322  012724  000137                    MOV     #000137,(R4)+   ; [ JMP @#handler_entry ]
      65 005326  010314                            MOV     R3,(R4)         ; Store address of int entry point
      66 005330  062324                            ADD     (R3)+,(R4)+
      67 005332  012310                            MOV     (R3)+,(R0)      ;Set PS for interrupt entry
      68 005334  052710  000340                    BIS     #340,(R0)       ;Make sure priority = 7
      69                                ;
      70                                ;  Save address beyond end of compiled interrupt catcher routine
      71                                ;
      72 005340  010437  000122'                   MOV     R4,XMVBAS       ;Save address beyond end of routine
      73                                ;
      74                                ;  Finished
      75                                ;
      76 005344  012604                 9$:       MOV     (SP)+,R4
      77 005346  000207                            RETURN
```

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

```
                                        .IF     NE,<PROASM-1>   ;If assembling for PDP-11
                                        .SBTTL  LININI -- Initialize time-sharing lines
                                ;-----------------------------------------------------------------------
                                ;  LININI is called to initialize the time-sharing lines.
                                ;  This consists of setting up interrupt vectors and setting control
                                ;  flags in the status registers.
                                ;
005350  010146                  LININI: MOV     R1,-(SP)
005352  010246                          MOV     R2,-(SP)
005354  010346                          MOV     R3,-(SP)
005356  010446                          MOV     R4,-(SP)
005360  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set up interrupt vectors for DL11 lines
                                ;
005362  012701  000002                  MOV     #2,R1           ;Index for 1st line
005366  012704  000340                  MOV     #340,R4         ;Priority 7 PS
005372  032761  0000000G 000000G 1$:    BIT     #$DEAD,LSW3(R1) ;Is this line uninstalled?
005400  001027                          BNE     8$              ;Br if yes
005402  032761  0000000G 000000G        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
005410  001423                          BEQ     8$              ;Br if not
005412  026127  0000000G 000000G        CMP     LCDTYP(R1),#CDX$DL ;Is this a DL11 line?
005420  001017                          BNE     8$              ;Br if not
                                ;
                                ;  DL-11 line
                                ;
005422  016105  0000000G                MOV     INVEC(R1),R5    ;GET ADDRESS OF INPUT VECTOR
005426  012702  0000000G                MOV     #INRECV,R2      ;END OF RECEIVING VECTOR
005432  012703  1777720G                MOV     #<OTRECV-6>,R3  ;START OF INPUT INTERRUPT ENTRY POINTS
005436  010100                          MOV     R1,R0           ;GET LINE NUMBER
005440  006300                          ASL     R0              ;4 BYTES PER INPUT INTERRUPT ENTRY POINT
005442  160002                          SUB     R0,R2           ;GET ADDRESS OF INPUT INTERRUPT ENTRY POINT
005444  060100                          ADD     R1,R0           ;6 BYTES PER OUTPUT INTERRUPT ENTRY POINT
005446  060003                          ADD     R0,R3           ;GET ADDRESS OF OUTPUT INTERRUPT ENTRY POINT
005450  010225                          MOV     R2,(R5)+        ;SET PC FOR INPUT INTERRUPT ENTRY POINT
005452  010425                          MOV     R4,(R5)+        ;SET PS FOR INPUT INTERRUPT
005454  010325                          MOV     R3,(R5)+        ;SET PC FOR OUTPUT INTERRUPT
005456  010425                          MOV     R4,(R5)+        ;SET PS FOR OUTPUT INTERRUPT
                                ;
                                ;  Try next line
                                ;
005460  062701  000002          8$:     ADD     #2,R1           ;ADVANCE LINE INDEX NUMBER
005464  020127  0000000G                CMP     R1,#LSTHL       ;MORE TO DO?
005470  101740                          BLOS    1$              ;BR IF YES
                                ;
                                ;  Initialize multiplexers.
                                ;
005472  012701  0000000G        SETMUX: MOV     #LSTMX,R1       ;Get last mux index #
005476  001423                          BEQ     SETLIN          ;Br if there are no mux lines
005500  026127  0000000G 000000G 3$:    CMP     MXTYPE(R1),#CDX$DZ ;Is this a DZ11, DH11, or DHV11?
005506  001412                          BEQ     1$              ;Br if DZ11
005510  026127  0000000G 000000G        CMP     MXTYPE(R1),#CDX$VH ;Is this a DHV11?
005516  001003                          BNE     2$              ;Br if not
005520  004737  006270'                  CALL    VHINIT          ;Initialize a DHV11
005524  000405                          BR      4$
005526  004737  006220'          2$:     CALL    DHINIT          ;Initialize a DH11
005532  000402                          BR      4$
```

```
                                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 005726  012605                        MOV     (SP)+,R5
    116 005730  012604                        MOV     (SP)+,R4
    117 005732  012603                        MOV     (SP)+,R3
    118 005734  012602                        MOV     (SP)+,R2
    119 005736  012601                        MOV     (SP)+,R1
    120 005740  000207                        RETURN
```

```
                                        .SBTTL  DHLPRM -- Set line parameters for a DH11 line
                                ;-----------------------------------------------------------------------
                                ;  DHLPRM is called to set up the line parameters for a DH11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;
005742                          DHLPRM:
                                ;
                                ;  Enable DM11 for this line
                                ;
005742  016100  000000G               MOV     LMXNUM(R1),R0   ;Get mux index number
005746  005760  000000G               TST     DM$CSR(R0)      ;Does this DH11 have DM11 modem control?
005752  001411                        BEQ     2$              ;Br if not
005754  142770  000000G 000000G       BICB    #MF$LIN,@DM$CSR(R0) ;Clear line # field in DM11 CSR
005762  156170  000000G 000000G       BISB    LMXLN(R1),@DM$CSR(R0);Select line of interest
005770  012770  000000G 000000G       MOV     #MF$LE,@DM$LSR(R0);Enable the line
                                ;
                                ;  Finished
                                ;
005776  000207                  2$:   RETURN
```

```
                                        .SBTTL  VHLPRM -- Set line parameter values for DHV11 line
                                ;-----------------------------------------------------------------------
                                ;  Set the line parameter values for a DHV11 line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;
006000                          VHLPRM:
                                ;
                                ;  Enable the line
                                ;
006000  016100  000000G                 MOV     LMXNUM(R1),R0   ;Get mux index number
006004  042770  000000G 000000G         BIC     #VF$LIN,@VH$CSR(R0) ;Clear line # field in mux CSR
006012  156170  000000G 000000G         BISB    LMXLN(R1),@VH$CSR(R0) ;Set our line #
006020  012770  000000G 000000G         MOV     #<VF$RE>,@VH$LCR(R0) ;Enable the line
                                ;
                                ;  Finished
                                ;
006026  000207                          RETURN
```

```
                                        .SBTTL  DZINIT -- Initialize a DZ11 multiplexer
                                ;-------------------------------------------------------------------------
                                ;  DZINIT is called to initialize a DZ11 multiplexer.
                                ;
                                ;  Inputs:
                                ;   R1 = Mux index number.
                                ;
006030                          DZINIT:
                                ;
                                ;  See if this DZ11 is installed
                                ;
006030  005761  000000G                 TST     MXCSR(R1)       ;Is this DZ-11 installed?
006034  001416                          BEQ     4$              ;Br if not
                                ;
                                ;  Set up interrupt vector connections for this MUX
                                ;
006036  004737  006074'                 CALL    MUXVEC          ;Set up interrupt vectors for this DZ11
                                ;
                                ;  Start up the mux operation
                                ;
006042  052771  000000G 000000G         BIS     #ZCLR,@MXCSR(R1);Do master clear on DZ-11
006050  032771  000000G 000000G 1$:      BIT     #ZCLR,@MXCSR(R1);Wait for clear to finish
006056  001374                          BNE     1$
006060  017100  000000G           2$:   MOV     @MXRBUF(R1),R0  ;Clear silo
006064  100775                          BMI     2$
006066  105071  000000G                 CLRB    @MXDTR(R1)      ;Disable all data sets
                                ;
                                ;  Finished
                                ;
006072  000207                  4$:     RETURN
```

```
                                        .SBTTL  MUXVEC -- Set up interrupt vectors for a multiplexer
                                ;-----------------------------------------------------------------------
                                ;  MUXVEC is called to set up the interrupt vector connections for
                                ;  a DZ11, DH11, or DHV11 multiplexer.
                                ;
                                ;  Inputs:
                                ;   R1 = Mux index number.
                                ;
006074  010246                  MUXVEC: MOV     R2,-(SP)
006076  010346                          MOV     R3,-(SP)
006100  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set interrupt vector for mux
                                ;
006102  016105  000000G                 MOV     MXVEC(R1),R5    ;Get address of input interrupt vector
006106  012702  000000G                 MOV     #INMXV,R2       ;End of receiving vector
006112  012703  177772G                 MOV     #<OTMXV-6>,R3   ;Output interrupt table
006116  010100                          MOV     R1,R0           ;Get mux index number
006120  006300                          ASL     R0              ;4 bytes per line in input int table
006122  160002                          SUB     R0,R2           ;Get address of input entry point
006124  060100                          ADD     R1,R0           ;6 bytes per mux in output entry point table
006126  060003                          ADD     R0,R3           ;Get address of output int entry point
006130  010225                          MOV     R2,(R5)+        ;Set PC for input interrupt
006132  012725  000340                  MOV     #340,(R5)+      ;Set PS for output interrupt
006136  010325                          MOV     R3,(R5)+        ;Set PC for output interrupt
006140  012715  000340                  MOV     #340,(R5)       ;Set PS for output interrupt
                                ;
                                ;  Now store an instruction sequence of the form:
                                ;
                                ;       JSR     R5,@#interrupt_routine
                                ;       .WORD   mux_index
                                ;
                                ;  to catch mux output interrupts and vector them to the interrupt routine.
                                ;
006144  012723  004537                  MOV     #004537,(R3)+   ;JSR R5,@#x
006150  012700  000000G                 MOV     #DZOINT,R0      ;Assume this is a DZ11
006154  026127  000000G 000000G         CMP     MXTYPE(R1),#CDX$DZ ;Is this a DZ11?
006162  001410                          BEQ     1$              ;Br if yes
006164  012700  000000G                 MOV     #VHOINT,R0      ;Assume this is a DHV11
006170  026127  000000G 000000G         CMP     MXTYPE(R1),#CDX$VH ;Is this a DHV11?
006176  001402                          BEQ     1$              ;Br if yes
006200  012700  000000G                 MOV     #DHOINT,R0      ;Get interrupt routine for DH11's
006204  010023                  1$:     MOV     R0,(R3)+        ;Store address of interrupt routine
006206  010113                          MOV     R1,(R3)         ;Store mux index number
                                ;
                                ;  Finished
                                ;
006210  012605                  9$:     MOV     (SP)+,R5
006212  012603                          MOV     (SP)+,R3
006214  012602                          MOV     (SP)+,R2
006216  000207                          RETURN
```

```
                                        .SBTTL  DHINIT -- Initialize a DH11 multiplexer
                                ;-----------------------------------------------------------------------
                                ;  DHINIT is called to initialize a DH11 multiplexer
                                ;
                                ;  Inputs:
                                ;   R1 = Mux index number
                                ;
006220                          DHINIT:
                                ;
                                ;  See if this DH11 is installed
                                ;
006220  005761  000000G                 TST     MH$SCR(R1)      ;Is this DH11 installed?
006224  001420                          BEQ     9$              ;Br if not
                                ;
                                ;  Connect interrupt vector to DH11
                                ;
006226  004737  006074'                 CALL    MUXVEC          ;Set up interrupt vectors for DH11
                                ;
                                ;  Clear the multiplexer
                                ;
006232  012771  000000G 000000G        MOV     #HF$MC,@MH$SCR(R1) ;Set the master-clear flag
006240  032771  000000G 000000G 1$:     BIT     #HF$MC,@MH$SCR(R1) ;Wait for the master clear to be completed
006246  001374                          BNE     1$
                                ;
                                ;  Clear the DM11 scanner
                                ;
006250  016100  000000G                 MOV     DM$CSR(R1),R0   ;Is there an associated DM11?
006254  001404                          BEQ     3$              ;Br if not
006256  012710  000000G                 MOV     #MF$CS,(R0)     ;Clear the scanner
006262  052710  000000G                 BIS     #MF$CM,(R0)     ;Clear the multiplexer
006266                          3$:
                                ;
                                ;  Finished
                                ;
006266                          9$:
006266  000207                          RETURN
```

```
                                        .SBTTL  VHINIT -- Initialize a DHV11 multiplexer
                                ;-----------------------------------------------------------------------
                                ;  Perform initialization for a DHV11 mux.
                                ;
                                ;  Inputs:
                                ;   R1 = Mux index number.
                                ;
006270                          VHINIT:
                                ;
                                ;  See if this DHV11 is installed
                                ;
006270  005761  000000G                 TST     VH$CSR(R1)      ;Is this DHV11 installed?
006274  001414                          BEQ     9$              ;Br if not
                                ;
                                ;  Connect interrupt vector to DHV11
                                ;
006276  004737  006074'                 CALL    MUXVEC          ;Set up interrupt vectors
                                ;
                                ;  Clear the multiplexer
                                ;
006302  012771  000000G 000000G         MOV     #VF$MR,@VH$CSR(R1) ;Reset the multiplexer
006310  032771  000000G 000000G 1$:     BIT     #VF$MR,@VH$CSR(R1) ;Wait for reset to finish
006316  001374                          BNE     1$
                                ;
                                ;  Clean out the FIFO buffer in the mux
                                ;
006320  017100  000000G         2$:     MOV     @MXRBUF(R1),R0  ;Get contents of receiver buffer register
006324  002775                          BLT     2$              ;Loop until RBUF empty
                                ;
                                ;  Finished
                                ;
006326  000207                  9$:     RETURN
```

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

```
                                        .SBTTL  LINTYP -- Determine the type of a line
                                ;-----------------------------------------------------------------------
                                ;  LINTYP is called to determine if the current line is a time-sharing line
                                ;  a CL line, or a non-hardware connected line.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number
                                ;
006330  020127  000000G         LINTYP: CMP     R1,#LSTPL       ;Is this a time-sharing line?
006334  101422                          BLOS    1$              ;Br if yes
006336  020127  000000G                 CMP     R1,#FSTIOL      ;Is this a CL line?
006342  103004                          BHIS    2$              ;Br if yes
006344  012761  177777  000000G         MOV     #-1,LCLUNT(R1)  ;Say line not in use as a CL line
006352  000432                          BR      9$
                                ;
                                ;  This is a CL line
                                ;
006354  016100  000000G         2$:     MOV     LCLUNT(R1),R0   ;Get the CL unit index number
006360  010160  000000G                 MOV     R1,CL$LIX(R0)   ;Say which line is assoc with this CL unit
006364  012761  000000G 000000G         MOV     #CLOTIR,LOUTIR(R1) ;Set terminal output interrupt routine
006372  012761  000000G 000000G         MOV     #CLINCP,LINIR(R1)  ;Set terminal input interrupt routine
006400  000414                          BR      8$
                                ;
                                ;  This is a time-sharing line
                                ;
006402  012761  177777  000000G 1$:     MOV     #-1,LCLUNT(R1)  ;Say line is not in use as a CL unit
006410  012761  177777  000000G         MOV     #-1,LXCL(R1)    ;Line is not cross-connected to CL unit
006416  012761  000000G 000000G         MOV     #NEDCHR,LOUTIR(R1) ;Set terminal output interrupt routine
006424  012761  000000G 000000G         MOV     #TTINCP,LINIR(R1)  ;Set terminal input interrupt routine
006432  052761  000000G 000000G 8$:     BIS     #$HARD,LSW3(R1) ;This line is connected to hardware
                                ;
                                ;  Finished
                                ;
006440  000207                  9$:     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.
                                ;
006442                          INITGO:
                                ;
                                ;  Save some RT-11 pointers in case we abort the initialization
                                ;
006442  013737  000004  000044'         MOV     @#4,RTTRP4      ;Save trap 4 vector
006450  013737  0000000 000046'         MOV     @#RMON,RTMNVC   ;RT-11 monitor pointer
006456  013737  0000000 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.
                                ;
006464                                  .GTIM   #AREA,#SYTIMH   ;Get the current time of day
                                ;
                                ;  Trap ^C for later test so we can restore clock vector
                                ;
006504                                  .SCCA   #AREA,#CCAFLG   ;Catch control-C
                                ;
                                ;  Check for TSGEN size overflow
                                ;
006524  012700  0000000                 MOV     #GENTOP,R0      ;Get top of TSGEN
006530  162700  037776                  SUB     #<40000-2>,R0   ;Will TSKMON have problems?
006534  003422                          BLE     15$             ;Continue if not
006536  010046                          MOV     R0,-(SP)        ;Save overflow size
006540                                  .PRINT  #TSXHD          ;Print error message
006546                                  .PRINT  #TOOBIG
006554                                  .PRINT  #REDUCE
006562  012600                          MOV     (SP)+,R0        ;Recover amount of overflow
006564  004737  027746'                 CALL    PRTDEC
006570                                  .PRINT  #BYTES
006576  000137  004216'                 JMP     INISTP
                                ;
                                ;  Initialize the system stack (below 1000)
                                ;
006602  012701  0000000 15$:    MOV     #SSEND,R1       ;Point to bottom of stack
006606  012700  123456          MOV     #123456,R0      ;Get initialization value
006612  010021          13$:    MOV     R0,(R1)+        ;Initialize the stack
006614  020127  0000000         CMP     R1,#SS          ;Reached top of the stack area?
006620  103774                  BLO     13$             ;Loop if not
006622  010106                  MOV     R1,SP           ;Run on system stack
                                ;
                                ;  Make sure we are not already running under TSX.
                                ;
006624                          .SERR                   ;DON'T DIE ON ERRORS
006632  012700  000242'         MOV     #GTLIN,R0       ;TSX EMT TO GET LINE NUMBER
006636  104375                  EMT     375             ;TRY A TSX EMT
006640  103410                  BCS     1$              ;BR IF NOT UNDER TSX
006642                          .PRINT  #TSXHD          ;ALREADY UNDER TSX
006650                          .PRINT  #TSXRUN
006656  000137  004216'         JMP     INISTP
006662                  1$:     .HERR                   ;RENABLE FATAL ERRORS
```

```
                                        ;
                                        ;  Make sure this machine has memory management facilities.
                                        ;
006670                                          .TRPSET #AREA,#NOXM     ;CATCH TRAPS
006710  005737  000000G                         TST     @#SROMMR        ;TRY TO ACCESS MEMORY MANAGEMENT REGISTER
006714                                          .TRPSET #AREA,#0        ;Release trap control
                                        ;
                                        ;  Request all available memory from RT-11.
                                        ;
006732                                          .SETTOP #-2             ;REQUEST ALL AVAILABLE MEMORY
006740  010037  000132'                         MOV     R0,TOPMEM       ;REMEMBER WHERE TOP OF MEMORY IS
006744  020027  000000G                         CMP     R0,#VPAR5       ;TSX CANNOT EXTEND ABOVE PAR5 BASE ADDRESS
006750  101402                                  BLOS    3$              ;BR IF RT-11 IS BELOW THAT
006752  012700  000000G                         MOV     #VPAR5,R0       ;SET PAR5 BASE AS UPPER LIMIT ON TSX SIZE
006756  010037  000302'                 3$:     MOV     R0,MEMLIM       ;TSX MAY NOT EXCEED THIS UPPER LIMIT
                                        ;
                                        ;  Lock USR in memory for speed
                                        ;  (Set USR to swap over TSEMT to get out of the way)
                                        ;
006762  012705  177776'                         MOV     #TSINIT-2,R5    ;GET THE BASE OF TSINIT
006766                                          .GVAL   #AREA,#374      ;GET SIZE OF RT-11 USR MODULE
007006  160005                                  SUB     R0,R5           ;ALLOCATE SPACE BELOW TSINIT FOR USR
007010  010537  000046                         MOV     R5,@#46         ;SET USR TO SWAP OVER TSEMT
007014                                  5$:     .LOCK                   ;LOCK USR IN MEMORY
                                        ;
                                        ;  Determine if we are to run system with the system debugger
                                        ;
007016  032737  000000G 000000G                 BIT     #CHAIN,@#JSWLOC ;WERE WE CHAINED TO?
007024  001406                                  BEQ     10$             ;BR IF NOT
007026  023737  000510  000222'                 CMP     @#510,R50ODT    ;SHOULD WE RUN UNDER ODT?
007034  001002                                  BNE     10$             ;BR IF NOT
007036  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.
                                        ;
007042                                  10$:
                                        .IF     NE,PROCID               ;** Do if assembling for pro only **
                                                CALL    INSCHK          ;PERFORM VERIFICATION AND DECRYPTION FOR PRO
                                        .ENDC   ;NE,PROCID
                                        ;
                                        ;  Allocate non-initialized buffer space over TSINIT.
                                        ;
007042  012705  000000'                         MOV     #TSINIT,R5      ;Allocate buffer space over TSINIT
007046  004737  013006'                         CALL    ALOCBF          ;Do allocation
007052  004737  013452'                         CALL    ALCSLO          ;Allocate silo buffers for lines
007056  020527  006442'                         CMP     R5,#INITGO      ;Are we beyond code that takes over control?
007062  103002                                  BHIS    12$             ;Br if yes
007064  012705  006442'                         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
007070  010537  000000G                 12$:    MOV     R5,INTSND       ;Ptr to base of interrupt stack
007074  062705  000002                         ADD     #2,R5           ;Always leave last word of stack for flag val
```

```
                                        MOV     @#RMON,R1           ;Get pointer to RT-11 RMON base
                                        BIT     #CW$PRO,370(R1)     ;Are we running on a PRO?
                                        BEQ     11$                 ;Br if not
                                        INCB    PROFLG              ;Set flag saying this is a PRO-350
                                        MOV     R5,PROBUF           ;Save pointer to buffer area
                                        ADD     #PIINSZ,R5          ;Allocate space for buffer
                                        BR      14$
                                11$:    ADD     #INTSSZ,R5          ;Allocate space for interrupt stack
                                14$:    MOV     R5,INTSTK           ;Address of top of interrupt stack
                                ;
                                ;  Allocate space for those overlays that go over TSINIT
                                ;
                                        CALL    OVLPOS              ;Determine how much space to alloc for overlay
                                ;
                                ;  Note: from this point onward we are carrying the address of the
                                ;  base of the free memory area in R5.
                                ;
                                        CMP     R5,#INITOP          ;Have we allocated up to top of TSINIT?
                                        BHIS    4$                  ;Br if yes
                                        MOV     #INITOP,R5          ;Advance to top of TSINIT
                                ;
                                ;  Allocate a 2048. byte work buffer
                                ;
                                4$:     CALL    ALCWRK              ;Allocate work buffer
                                ;
                                ;  Allocate empty Region Control Blocks for use by handlers
                                ;
                                        CALL    ALCHRB
                                ;
                                ;  If we were started in debug mode, load ODT.
                                ;
                                        .IF     EQ,PROCID           ;Don't allow ODT for production PRO version
                                        TST     ODTFLG              ;Are we to load system debugger?
                                        BEQ     2$                  ;Br if not
                                        CALL    GETODT              ;Load ODT and start it
                                        .ENDC   ;EQ,PROCID
                                ;
                                ;  Initialize memory management registers for 1-to-1 mapping but
                                ;  leave memory management turned off
                                ;
                                2$:     CALL    MEMINI              ;Initialize memory management
                                ;
                                ;  Extract information from RT-11 configuration and sysgen words.
                                ;
                                        MOV     @#RMON,R1           ;GET POINTER TO BASE OF RMON
                                        MOV     300(R1),R2          ;GET RT-11 CONFIGURATION WORD
                                        BIC     #CW$GDH+CW$BTH+CW$LGS,R2 ;RESET A FEW FLAGS
                                        BIS     #CW$FB+CW$FGJ+CW$USR+CW$XM,R2 ;SET A FEW FLAGS
                                        MOV     R2,CONFIG           ;INITIALIZE OUR CONFIGURATION WORD
                                ;  Now get extended configutation word.
                                        MOV     370(R1),CONFG2      ;EXTENDED CONFIGURATION WORD
                                        BIS     #CW$ESP,CONFG2      ;SET EXIT NO SWAP FLAG
                                        CMPB    VBUSTP,#QBUS        ;Is this a Q-bus machine?
                                        BNE     25$                 ;Br if not
                                        BIS     #CW$QBS,CONFG2      ;Set QBUS flag
                                ;  And sysgen option word.
                                25$:    MOV     372(R1),R2
```

```
    172 007270  042702  000000C                 BIC     #SG$ELG+SG$PAR+SG$MTS,R2
    173 007274  052702  000000C                 BIS     #SG$MMU+SG$MTM+SG$IOT+SG$TSX,R2
    174 007300  010237  000000G                 MOV     R2,SYSGEN       ;INITIALIZE OUR SYSGEN WORD
    175                             ;  Get system version number
    176 007304                              .GVAL   #AREA,#276      ;GET RT-11 SYSTEM VERSION NUMBER
    177 007324  010037  000000G                 MOV     R0,SYSVER       ;SET AS TSX-PLUS VERSION NUMBER
```

```
                                        ;
                                        ;  Set up a few clock constants based on clock frequency.
                                        ;  See if we have a 50 or 60 Hz clock
                                        ;
007330  032737  000000G 000000G INICLK: BIT     #CW$50H,CONFIG  ;50 or 60 Hz clock?
007336  001017                        BNE     2$              ;Br if 50 Hz
                                        ;
                                        ;  60 Hz clock
                                        ;
007340  012737  000074  000000G         MOV     #60.,TK1SEC     ;Clock ticks per 1 second
007346  012737  000036  000000G         MOV     #30.,TK5VAL     ;Clock ticks per 0.5 seconds
007354  012737  000264  000000G         MOV     #180.,TK3SVL    ;Clock ticks per 3 seconds
007362  012737  000006  000000G         MOV     #6.,TK1VAL      ;Clock ticks per 0.1 seconds
007370  012700  001130                  MOV     #600.,R0        ;Get # clock ticks per 10 seconds
007374  000416                        BR      8$
                                        ;
                                        ;  50 Hz clock
                                        ;
007376  012737  000062  000000G 2$:     MOV     #50.,TK1SEC     ;Clock ticks per 1 second
007404  012737  000031  000000G         MOV     #25.,TK5VAL     ;Clock ticks per 0.5 seconds
007412  012737  000226  000000G         MOV     #150.,TK3SVL    ;Clock ticks per 3 seconds
007420  012737  000005  000000G         MOV     #5.,TK1VAL      ;Clock ticks per 0.1 seconds
007426  012700  000764                  MOV     #500.,R0        ;Get # clock ticks per 10 seconds
                                        ;
                                        ;  Set number of clock ticks per day
                                        ;
007432  012702  020700          8$:     MOV     #8640.,R2       ;(# seconds per day) / 10.
007436  070200                        MUL     R0,R2           ;Get # clock ticks per day
007440  010237  000000G                 MOV     R2,DATIMH       ;High-order value
007444  010337  000000G                 MOV     R3,DATIML       ;Low-order value
                                        ;
                                        ;  Do a fast check to make sure specified T/S line addresses are ok.
                                        ;
007450  004737  010376'         CKLIN:  CALL    LINCHK          ;CHECK T/S LINE ADDRESSES
                                        ;
                                        ;  Do PRO-350 system initialization
                                        ;
                                .IF     NE,PROASM
                                        BIT     #CW$PRO,CONFG2  ;Are we running on a PRO-350?
                                        BEQ     INIDEV          ;Br if not
                                        CALL    PROINI          ;Do PRO-350 initialization
                                        MOV     #PIDRIV,PIDPTR  ;Set up pointer to clock driven PI routine
                                .ENDC   ;NE,PROASM
                                        ;
                                        ;  Make entry in device handler table for TT device.
                                        ;
007454  013737  000172' 000000G INIDEV: MOV     R50TT,PNAME     ;PERMANENT NAME "TT"
007462  012737  000000G 000000G         MOV     #DI$TT,DVSTAT   ;SET DEVICE STATUS FLAGS FOR TT
007470  005037  000000G                 CLR     DVFLAG
007474  005037  000000G                 CLR     DEVSIZ
007500  012737  000002  000000G         MOV     #2,HANENT       ;SET UP HANENT SO HANDLER LOOKS RESIDENT
007506  005037  000000G                 CLR     NUMDEV          ;IT IS DEVICE # 0
                                        ;
                                        ;  Make device table entry for LD (logical disk) device
                                        ;
007512  012737  177777  000000G         MOV     #-1,LDDEVX      ;ASSUME LD SUPPORT NOT WANTED
007520  105737  000000G                 TSTB    VLDSYS          ;IS LD SUPPORT GENNED IN?
```

```
                007524  001427                          BEQ     6$                      ;BR IF NOT
                007526  062737  000002  000000G         ADD     #2,NUMDEV               ;ONE MORE DEVICE
                007534  013701  000000G                 MOV     NUMDEV,R1               ;GET DEVICE TABLE INDEX
                007540  010137  000000G                 MOV     R1,LDDEVX               ;REMEMBER INDEX NUMBER FOR LD DEVICE
                007544  013761  000176' 000000G         MOV     R50LD,PNAME(R1)         ;SET DEVICE NAME ("LD")
                007552  012761  000000C 000000G         MOV     #<DS$DIR+DS$SFN+DS$VSZ+DI$LD>,DVSTAT(R1);SET DEV STATUS FLAGS
                007560  012761  000000G 000000G         MOV     #DX$EBA,DVFLAG(R1);Say buffers must be on even byte boundaries
                007566  005061  000000G                 CLR     DEVSIZ(R1)
                007572  012761  000002  000000G         MOV     #2,HANENT(R1)           ;SAY HANDLER IS RESIDENT
                007600  004737  015312'                 CALL    LDINIT                  ;DETERMINE LD TRANSLATION TABLE FORMAT
                                                ;
                                                ;  Make device table entry for CL (communications line) device
                                                ;
                007604  005727  000000G         6$:     TST     #CLTOTL                 ;Are there any communications lines?
                007610  001402                          BEQ     8$                      ;Br if not
                007612  004737  014572'                 CALL    CLINIT                  ;Initialize CL handler
                                                ;
                                                ;  Disable clock interrupts.
                                                ;
                007616  012737  000002  000000  8$:     MOV     #2,@#0                  ;LOAD RTI IN LOCATION 0
                007624  005037  000000G                 CLR     @#CLKVEC                ;ATTACH CLOCK INTERRUPT TO 0
                007630  032737  000000G 000000G         BIT     #CW$PRO,CONFG2          ;ARE WE RUNNING ON A PRO?
                007636  001402                          BEQ     1$                      ;BR IF NOT
                007640  005037  000230                  CLR     @#230                   ;380 CLOCK INTERRUPT VECTOR
                                                ;
                                                ;  Set up memory parity control
                                                ;
                007644  004737  017700'         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
                                        ;
007720  004737  015372'                       CALL    INDINI          ;INITIALIZE FOR IND PROGRAM
                                        ;
                                        ;  Initialize the TSXUCL data file
                                        ;
007724  004737  016134'                       CALL    UCLINI
                                        ;
                                        ;  Set name of device that UCL program is to be run from
                                        ;
007730  013737  000000G 000000G               MOV     SYNAME,UCLNAM   ;SET DEVICE NAME FOR UCL PROGRAM
                                        ;
                                        ;  Initialize spooling system
                                        ;
007736  004737  011656'                       CALL    SPLINI          ;INITIALIZE SPOOLING SYSTEM
                                        ;
                                        ;  Open system swap file
                                        ;
007742  105737  000000G                       TSTB    VSWPFL          ;IS JOB SWAPPING ALLOWED?
007746  001402                               BEQ     3$              ;BR IF NOT
007750  004737  010720'                       CALL    OPNSWP          ;OPEN THE SYSTEM SWAP FILE
                                        ;
                                        ;  Open swap file used for PLAS regions
                                        ;
007754  004737  011352'                3$:    CALL    OPNRSF          ;Open PLAS region swap file
                                        ;
                                        ;  Set up information about which devices need to have their I/O mapped
                                        ;
007760  004737  023160'                       CALL    SETMIO          ;Set up information about mapped I/O
                                        ;
                                        ;  We are finished allocating low-memory buffer space.
                                        ;
007764  010500                               MOV     R5,R0           ;ENSURE WE DON'T OVERFLOW 40KB
007766  004737  027622'                       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.
                                        ;
007772  010537  000000G                       MOV     R5,UMSYTP       ;SAVE ADDRESS OF NON-EXTENDED SYSTEM TOP
007776  062705  000077                       ADD     #77,R5          ;BOUND UP TO 64-BYTE BOUNDARY
010002  072527  177772                       ASH     #-6,R5          ;CONVERT TO 64-BYTE BLOCK #
010006  042705  176000                       BIC     #176000,R5      ;KILL SIGN EXTENSION
                                        ;
                                        ;  Allocate buffer space that is not contrained to 40Kb TSX-Plus region.
                                        ;
010012  004737  013672'                       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.
                                        ;
010016  010504                               MOV     R5,R4           ;Save the base of free memory in R4
010020  010437  000136'                       MOV     R4,FMEMLO       ;Save pointer above top of alloc low memory
010024  013705  000134'                       MOV     FMEMHI,R5       ;Get 64-byte block # of free high memory
                                        ;
                                        ;  Load any mapped system code
                                        ;
```

```
    172 010030  004737  024012'                 CALL    GETMAP          ;LOAD USR, EMT, MSG, LOCK, SPOOL, etc.
    173                                 ;
    174                                 ;  Load any shared run-time systems
    175                                 ;
    176 010034  005727  000000G                 TST     #NUMRDB         ;Do we need to load any shared run-times?
    177 010040  001415                          BEQ     4$              ;Br if not
    178 010042  012701  000000G                 MOV     #RDB,R1         ;Point to 1st run-time descriptor block
    179 010046  010502                          MOV     R5,R2           ;Save initial memory pointer
    180 010050  004737  025456'         5$:     CALL    GETSRT          ;Load a shared run-time system
    181 010054  062701  000000G                 ADD     #RT$$SZ,R1      ;Point to next shared run-time descriptor
    182 010060  020127  000000G                 CMP     R1,#RDBEND      ;Are there more to load?
    183 010064  103771                          BLO     5$              ;Br if yes
    184 010066  160502                          SUB     R5,R2           ;Compute amt of space used by run-times
    185 010070  010237  000000G                 MOV     R2,SRTSIZ       ;Save total run-time size
    186 010074                          4$:
    187                                 .IF     NE,PROASM
    188                                 ;
    189                                 ;  If we are running on a Pro, load the PI handler like a shared run-time
    190                                 ;
    191                                         TSTB    PROFLG          ;Are we running on a Pro?
    192                                         BEQ     10$             ;Br if not
    193                                         MOV     #PISRT,R1       ;Point to dummy shared run-time block for PI
    194                                         MOV     R5,R2           ;Save current memory pointer
    195                                         CALL    GETSRT          ;Load PI handler like a shared run-time
    196                                         SUB     R5,R2           ;Calculate amt of space used by PI handler
    197                                         ADD     R2,MHNSIZ       ;Count in mapped-handler size
    198                                 .ENDC   ;NE,PROASM
    199                                 ;
    200                                 ;  Load any mapped handlers
    201                                 ;
    202 010074  004737  021274'         10$:    CALL    GETHNH          ;Load mapped handlers
    203                                 ;
    204                                 ;  Allocate space for data cache buffers and control tables
    205                                 ;
    206 010100  004737  026070'                 CALL    CSHBUF          ;Allocate space for data cache
    207                                 ;
    208                                 ;  We have finished allocating all of the memory used by the system.
    209                                 ;  Allocate and initialize a memory map table that will be used to
    210                                 ;  show which pages are available for user jobs.
    211                                 ;
    212 010104  004737  017370'                 CALL    MAPALC          ;Allocate memory map table
    213                                 ;
    214                                 ;  Set up info about maximum memory space available to jobs
    215                                 ;
    216 010110  004737  017554'                 CALL    SETJSZ          ;SET JOB SIZE INFO
    217                                 ;
    218                                 ;  Set up date and time
    219                                 ;
    220 010114  013702  000000G                 MOV     SYTIML,R2       ;Save time we got at start of init
    221 010120                                  .GTIM   #AREA,#SYTIMH   ;SET TIME OF DAY
    222 010140                                  .DATE                   ;GET DATE
    223 010146  010037  000000G                 MOV     R0,SYSDAT       ;SET SYSTEM DATE
    224 010152  020237  000000G                 CMP     R2,SYTIML       ;Make sure some time has elapsed
    225 010156  001010                          BNE     11$             ;Br if clock is running
    226 010160                                  .PRINT  #TSXHD          ;Print error message heading
    227 010166                                  .PRINT  #NOCLOK         ;Print clock-not-working message
    228 010174  000137  004216'                 JMP     INISTP          ;Abort initialization
```

```
                                        ;
                                        ;  Unlock the USR so that TSEMT will be swapped back in.
                                        ;
010200                                  11$:    .UNLOCK                 ;RELEASE USR
                                        ;
                                        ;  Read back into memory that part of the resident portion of TSX
                                        ;  that we overlayed with our work buffer.
                                        ;
010202  013702  000154'                         MOV     WRKSIZ,R2       ;Get size of work buffer
010206  006202                                  ASR     R2              ;Convert to # words
010210  013703  000152'                         MOV     WRKBUF,R3       ;Get address of work buffer area
010214  000241                                  CLC                     ;Convert to block # in TSX.SAV file
010216  006003                                  ROR     R3
010220  000303                                  SWAB    R3
010222                                          .READW  #AREA,#17,WRKBUF,R2,R3 ;Read back TSX over work buffer
                                        ;
                                        ;  See if user requested control-C abort
                                        ;
010256  004737  027662'                         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
                                        ;
010262  000137  003534'                         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).
                                ;
010266  010246                  ALCWRK: MOV     R2,-(SP)
                                ;
                                ;  Get address of start of area where buffer can go and then bound
                                ;  up to a block boundary.
                                ;
010270  012702  000000G                 MOV     #EXCBUF,R2      ;Get address of base of buffer area
010274  062702  000777                  ADD     #777,R2         ;Bound up to block boundary
010300  042702  000777                  BIC     #777,R2         ;Set to block boundary
010304  010237  000152'                 MOV     R2,WRKBUF
010310  012737  004000  000154'         MOV     #2048.,WRKSIZ
                                ;
                                ;  Finished
                                ;
010316  012602                          MOV     (SP)+,R2
010320  000207                          RETURN
```

```
                                        .SBTTL  ALCHRB -- Allocate Region Control Blocks for handlers
                                ;-----------------------------------------------------------------------
                                ;  This routine allocates and initializes empty Region Control Blocks for
                                ;  use by device handlers.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to start of memory area where RCB's are to be built.
                                ;
                                ;  Outputs:
                                ;   R5 = Pointer past end of RCB area.
                                ;
010322  010246                  ALCHRB: MOV     R2,-(SP)
                                ;
                                ;  Get count of # RCB's to build
                                ;
010324  013700  000000G                 MOV     NDVRCB,R0       ;Get # RCB's to build for handlers
                                ;
                                ;  Store pointer to start of RCB area and store -1 at beginning of area
                                ;
010330  010537  000000G                 MOV     R5,HANRCB       ;Start of RCB area
010334  010537  000000G                 MOV     R5,HANRCO       ;Store offset relative to MONVEC
010340  162737  000000G 000000G        SUB     #MONVEC,HANRCO  ;Convert address to offset
010346  012725  177777                  MOV     #-1,(R5)+       ;Store -1 at start of area
                                ;
                                ;  Allocate and initialize to zero the RCB's
                                ;
010352  012702  000005          1$:     MOV     #5.,R2          ;Each RCB has 5 words
010356  005025                  2$:     CLR     (R5)+           ;Zero the RCB
010360  077202                          SOB     R2,2$
                                ;
                                ;  See if there are more RCB's to build
                                ;
010362  005300                          DEC     R0              ;More RCB's to initialize?
010364  003372                          BGT     1$              ;Loop if yes
                                ;
                                ;  Store -1 at end of RCB area
                                ;
010366  012725  177777                  MOV     #-1,(R5)+       ;Mark end of RCB list
                                ;
                                ;  Finished
                                ;
010372  012602                          MOV     (SP)+,R2
010374  000207                          RETURN
```

```
                                        .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 $DEAD flag for the line if INIABT=0.
                                ;
010376  010146                 LINCHK: MOV     R1,-(SP)
010400  010246                         MOV     R2,-(SP)
010402  010346                         MOV     R3,-(SP)
010404  010446                         MOV     R4,-(SP)
010406  013746  000004                 MOV     @#4,-(SP)       ;SAVE ORIGINAL TRAP VECTOR
                                ;
                                ;  Take over trap control
                                ;
010412  012737  010612' 000004         MOV     #6$,@#4         ;CATCH TRAPS
                                ;
                                ;  Loop through the test for each line.
                                ;
010420  012701  000000G                MOV     #LSTLIN,R1      ;NUMBER OF LAST LINE
                                ;
                                ;  Determine if this is a primary line or an I/O line and set the
                                ;  addresses of the interrupt service routines.
                                ;
010424  004737  006330'         1$:     CALL    LINTYP          ;Determine the type of this line
010430  032761  000000G 000000G        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
010436  001440                         BEQ     31$             ;Br if not
010440  016103  000000G                MOV     LMXNUM(R1),R3   ;IS THIS A DL-11 OR MULTIPLEXER LINE?
010444  001411                         BEQ     2$              ;BR IF DL-11
010446  016302  000000G                MOV     MXCSR(R3),R2    ;GET DZ11 OR DH11 STATUS REGISTER ADDRESS
010452  001403                         BEQ     11$             ;BR IF ALREADY MARKED AS DEAD
010454  016304  000000G                MOV     MXVEC(R3),R4    ;GET MUX INTERRUPT VECTOR ADDRESS
010460  000407                         BR      3$
010462  004737  010564'         11$:    CALL    4$              ;MARK LINE AS DEAD
010466  000424                         BR      31$             ;CONTINUE CHECKING TERMINALS
010470  016102  000000G         2$:     MOV     RSR(R1),R2      ;GET DL-11 STATUS REGISTER ADDRESS
010474  016104  000000G                MOV     INVEC(R1),R4    ;GET DL-11 INTERRUPT VECTOR ADDRESS
                                ;  Check validity of status register address.
010500  020227  160000         3$:     CMP     R2,#160000      ;IS IT IN I/O PAGE?
010504  103445                         BLO     LINTRP          ;ERROR IF NOT
010506  032702  000007                 BIT     #7,R2           ;IS IT ON 8-BYTE BOUNDARY?
010512  001042                         BNE     LINTRP          ;ERROR IF NOT
010514  005712                         TST     @R2             ;TRY TO ACCESS IT AND SEE IF WE TRAP
                                ;  Check validty of interrupt vector address.
010516  020427  000060                 CMP     R4,#60          ;CAN'T BE BELOW 60
010522  103445                         BLO     BADVEC
010524  020427  000500                 CMP     R4,#500         ;OR ABOVE 500
010530  103042                         BHIS    BADVEC
010532  032704  000007                 BIT     #7,R4           ;MUST BE ON 8-BYTE BOUNDARY
010536  001037                         BNE     BADVEC
                                ;  This line looks good.  Check next.
010540  162701  000002         31$:    SUB     #2,R1           ;MORE TO CHECK?
010544  003327                         BGT     1$              ;BR IF YES
                                ;
                                ;  Finished -- all lines look ok.
                                ;
```

```
     58 010546  012637  000004                    MOV     (SP)+,@#4       ;RESTORE TRAP VECTOR
     59 010552  012604                            MOV     (SP)+,R4
     60 010554  012603                            MOV     (SP)+,R3
     61 010556  012602                            MOV     (SP)+,R2
     62 010560  012601                            MOV     (SP)+,R1
     63 010562  000207                            RETURN
     64                                 ;
     65                                 ;  See if we should abort or just mark the line as dead.
     66                                 ;
     67 010564  105737  000000G         4$:       TSTB    VINABT          ;DOES HE WANT TO ABORT?
     68 010570  001013                            BNE     LINTRP          ;YES
     69 010572  052761  000000G 000000G           BIS     #$DEAD,LSW3(R1) ;MARK LINE AS DEAD
     70 010600  005703                            TST     R3              ;IS THIS A DL11 OR A MUX LINE?
     71 010602  001402                            BEQ     5$              ;BR IF DL11
     72 010604  005063  000000G                   CLR     MXCSR(R3)       ;MARK DZ OR DH AS DEAD
     73 010610  000207                  5$:       RETURN
     74
     75                                 ;
     76                                 ;  Trap occured while trying to access status register.
     77                                 ;
     78 010612  004737  010564'         6$:       CALL    4$              ;REPORT ERROR OR MARK AS DEAD LINE
     79 010616  000002                            RTI                     ;RETURN TO LINE CHECKING
     80                                 ;
     81                                 ;  Error: Invalid status register address.
     82                                 ;  R1 = Line number,  R2 = status register address
     83                                 ;
     84 010620                          LINTRP:   .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     85 010626                                    .PRINT  #BADLIN
     86 010634  000407                            BR      ERP
     87                                 ;
     88                                 ;  Error: Invalid interrupt vector address.
     89                                 ;  R1 = Line number,  R4 = interrupt vector address
     90                                 ;
     91 010636                          BADVEC:   .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     92 010644                                    .PRINT  #BDVMSG
     93 010652  010402                            MOV     R4,R2           ;GET VECTOR ADDRESS TO R2
     94 010654  010200                  ERP:      MOV     R2,R0           ;GET ADDRESS TO R0
     95 010656  004737  027676'                   CALL    PRTOCT          ;PRINT OCTAL VALUE
     96 010662                                    .PRINT  #CRLF
     97 010670                                    .PRINT  #BDLMSG         ;LINE # =
     98 010676  010100                            MOV     R1,R0           ;GET LINE NUMBER
     99 010700  006200                            ASR     R0              ;# 1
    100 010702  004737  027746'                   CALL    PRTDEC          ;PRINT LINE NUMBER
    101 010706                                    .PRINT  #CRLF
    102 010714  000137  004216'                   JMP     INISTP          ;ABORT INITIALIZATION
    103                                           .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.
                                ;
010720  010146                  OPNSWP: MOV     R1,-(SP)
010722  010246                          MOV     R2,-(SP)
010724  010346                          MOV     R3,-(SP)
                                ;
                                ;  Load RT-11 handler for swap device.
                                ;
010726  013700  000000G                 MOV     SWDBLK,R0       ;Get name of device
010732  004737  027506'                 CALL    RTFTCH          ;Fetch the RT-11 handler
010736  103546                          BCS     11$             ;Br if invalid device
                                ;
                                ;  Compute the maximum number of slots in swap file that we could need
                                ;
010740  012703  000000C                 MOV     #NSL+NDL,R3     ;Get # virtual lines and detached jobs
010744  012701  000000G                 MOV     #LSTPL,R1       ;Get index to last primary line
010750  032761  000000G 000000G 1$:     BIT     #$DEAD,LSW3(R1) ;Is this line installed?
010756  001001                          BNE     5$              ;Br if not
010760  005203                          INC     R3              ;Count another primary line
010762  162701  000002          5$:     SUB     #2,R1           ;Get next line index
010766  003370                          BGT     1$              ;Loop if more lines to check
010770  020337  000000G                 CMP     R3,VSWPSL       ;Compare with # slots specified
010774  002002                          BGE     6$              ;Br if VSWPSL value is ok
010776  010337  000000G                 MOV     R3,VSWPSL       ;Reduce number of slots in swap file
                                ;
                                ;  Determine how many blocks are needed for each slot in swap file.
                                ;
011002  013703  000000G         6$:     MOV     VHIMEM,R3       ;GET # BLOCKS NEEDED FOR LARGEST JOB SIZE
011006  006303                          ASL     R3
011010  063703  000000G                 ADD     CXTPAG,R3       ;ADD # BLOCKS NEEDED FOR JOB CONTEXT AREA
011014  010337  000000G                 MOV     R3,SLTSIZ       ;Save size of swap file slot
                                ;
                                ;  Compute the total number of blocks needed for the swap file.
                                ;
011020  070337  000000G                 MUL     VSWPSL,R3       ;Multiply by # slots in swap file
                                ;
                                ;  R3 now contains total number of blocks needed in swap file.
                                ;  See if swap file already exists on disk.
                                ;
011024                          4$:     .LOOKUP #AREA,#1,#SWDBLK;DOES SWAP FILE EXIST NOW?
011044  103415                          BCS     2$              ;BR IF NOT
                                ;  Swap file exists.  See if it is the right size.
011046  020003                          CMP     R0,R3           ;IS SWAP FILE THE RIGHT SIZE?
011050  001453                          BEQ     3$              ;BR IF YES
                                ;  Old swap file is not of correct size.
                                ;  Delete it and open a new swap file.
011052                                  .CLOSE  #1
```

```
 011060                                         .DELETE #AREA,#1,#SWDBLK;DELETE THE OLD SWAP FILE
                                        ;
                                        ;  Create a new swap file.
                                        ;
 011100                                 2$:     .ENTER  #AREA,#1,#SWDBLK,R3 ;CREATE A NEW SWAP FILE
 011124  103443                                 BCS     9$                  ;BR IF SOME ERROR ON OPEN
                                        ;
                                        ;  Swap file has been created.
                                        ;  Write to last block to reserve full space in file then close
                                        ;  and reopen the channel using a .lookup.
                                        ;
 011126  005303                                 DEC     R3                  ;GET # OF LAST BLOCK IN FILE
 011130                                         .WRITW  #AREA,#1,#TSINIT,#256.,R3 ;WRITE TO LAST BLOCK IN FILE
 011166  005203                                 INC     R3                  ;GET BACK # BLOCKS IN FILE
 011170                                         .CLOSE  #1                  ;CLOSE FILE WE CREATED
 011176  000712                                 BR      4$                  ;NOW GO REOPEN USING A .LOOKUP
                                        ;
                                        ;  Swap file has been successfully opened using a .lookup.
                                        ;  Now copy channel status to TSX swap channel.
                                        ;
 011200  012700  000000G                3$:     MOV     #SWPCHN,R0          ;POINT TO SWAP CHANNEL BLOCK
 011204  013702  000000G                        MOV     SWDBLK,R2           ;GET DEVICE NAME
 011210  004737  027126'                        CALL    SETCHN              ;SET UP SWAP CHANNEL INFO
                                        ;
                                        ;  Release the RT-11 device handler
                                        ;
 011214                                         .RELEAS #SWDBLK             ;Release RT-11 device handler
                                        ;
                                        ;  Finished
                                        ;
 011224  012603                                 MOV     (SP)+,R3
 011226  012602                                 MOV     (SP)+,R2
 011230  012601                                 MOV     (SP)+,R1
 011232  000207                                 RETURN
                                        ;
                                        ;  Error: Cannot open swap file
                                        ;
 011234                                 9$:     .PRINT  #TSXHD              ;PRINT ERROR MESSAGE
 011242                                         .PRINT  #BADOPN
 011250  004737  011276'                        CALL    SPNEED              ;Print info about number of blocks needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
 011254  010001                         11$:    MOV     R0,R1               ;Save device name
 011256                                         .PRINT  #TSXHD              ;Print error message
 011264                                         .PRINT  #BADOPN
 011272  004737  011324'                        CALL    BADDEV              ;Print invalid device specification
                                        ;
                                        ;  Error: Number of contiguous blocks required.
                                        ;
 011276                                 SPNEED: .PRINT  #CONSPC             ;Print contiquous blocks needed
 011304  010300                                 MOV     R3,R0               ;GET # BLOCKS NEEDED FOR FILE
 011306  004737  027746'                        CALL    PRTDEC              ;DISPLAY # BLOCKS NEEDED
 011312                                         .PRINT  #CRLF
 011320  000137  004216'                        JMP     INISTP              ;ABORT INITIALIZATION
                                        ;
                                        ;  Bad file specification.
```

```
    115                                         ;
    116 011324                                  BADDEV: .PRINT  #CFHMSG         ;Print invalid device specification
    117 011332  010100                                  MOV     R1,R0           ;Get the rad50 device name
    118 011334  004737  030012'                         CALL    PRTR50          ;Print rad50 device name
    119 011340                                          .PRINT  #CRLF           ;Print carriage return/line feed
    120 011346  000137  004216'                         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.
                                ;
011352  010346                  OPNRSF: MOV     R3,-(SP)
                                ;
                                ;  Return if this is a non-swapping system or if region swap file is
                                ;  not wanted.
                                ;
011354  105737  000000G                 TSTB    VSWPFL          ;Is this a non-swapping system?
011360  001513                          BEQ     9$              ;Br if yes
011362  005737  000000G                 TST     VPLAS           ;Is a PLAS swap file wanted?
011366  001510                          BEQ     9$              ;Br if not
                                ;
                                ;  Load RT-11 device handler for swap device
                                ;
011370  013700  000000G                 MOV     RSFBLK,R0       ;Get name of device
011374  004737  027506'                 CALL    RTFTCH          ;Try to fetch the RT-11 device handler
011400  103515                          BCS     11$             ;Br if error on handler fetch
011402  013703  000000G                 MOV     VPLAS,R3        ;Get # blocks in PLAS swap file
                                ;
                                ;  See if PLAS swap file already exists on disk
                                ;
011406                          4$:     .LOOKUP #AREA,#1,#RSFBLK ;Try to find existing PLAS swap file
011426  103416                          BCS     2$              ;Br if file does not now exist
                                ;
                                ;  PLAS swap file exists.
                                ;  See if it is the correct size.
                                ;
011430  020037  000000G                 CMP     R0,VPLAS        ;Is swap file of the correct size?
011434  001453                          BEQ     3$              ;Br if yes
                                ;
                                ;  Old PLAS swap file is not of correct size.
                                ;  Delete it and open a new swap file.
                                ;
011436                                  .CLOSE  #1              ;Close and delete the old file
011444                                  .DELETE #AREA,#1,#RSFBLK;Delete the old file
                                ;
                                ;  Create new swap file
                                ;
011464                          2$:     .ENTER  #AREA,#1,#RSFBLK,VPLAS ;Create a new PLAS swap file
011512  103440                          BCS     10$             ;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.
                                ;
011514  005303                          DEC     R3              ;Get # of last block in file
011516                                  .WRITW  #AREA,#1,#TSINIT,#256.,R3
011554                                  .CLOSE  #1
011562  000711                          BR      4$              ;Go back and lookup file
```

```
                                ;
                                ;  Swap file has been successfully opened using lookup.
                                ;  Copy channel status to TSX channel block.
                                ;
011564  012700  000000G         3$:     MOV     #SEGCHN,R0      ;Point to TSX PLAS swap channel
011570  013703  000000G                 MOV     RSFBLK,R3       ;Get device name
011574  004737  027126'                 CALL    SETCHN          ;Set up TSX channel block
                                ;
                                ;  Release the RT-11 device handler
                                ;
011600                                  .RELEAS #RSFBLK         ;Release RT-11 device handler
                                ;
                                ;  Finished
                                ;
011610  012603                  9$:     MOV     (SP)+,R3
011612  000207                          RETURN
                                ;
                                ;  Error -- Cannot open PLAS swap file
                                ;
011614                          10$:    .PRINT  #TSXHD          ;Print error prefix
011622                                  .PRINT  #RSFERR         ;Print error message
011630  004737  011276'                 CALL    SPNEED          ;Print information about amt of space needed
                                ;
                                ;  Error: Invalid device specification.
                                ;
011634  010001                  11$:    MOV     R0,R1           ;Save device name
011636                                  .PRINT  #TSXHD          ;Print error message
011644                                  .PRINT  #RSFERR
011652  004737  011324'                 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.
                                ;
011656  005727  000000G         SPLINI: TST     #SPLND          ;Are there any spooled devices?
011662  001401                          BEQ     13$             ;Br if not
011664  000401                          BR      10$             ;Initialize the spooled devices
011666  000207                  13$:    RETURN
                                ;
                                ;  There are some spooled devices
                                ;
011670  010146                  10$:    MOV     R1,-(SP)
011672  010246                          MOV     R2,-(SP)
011674  010346                          MOV     R3,-(SP)
011676  010446                          MOV     R4,-(SP)
011700  010546                          MOV     R5,-(SP)
                                ;
                                ;  Open each spooled device
                                ;
011702  105037  000000G                 CLRB    NSPLDV          ;INIT COUNT OF # ACTUAL SPOOLED DEVICES
011706  012701  000000G                 MOV     #SDCB,R1        ;POINT TO 1ST SDCB
011712  012703  000000G                 MOV     #SPLDEV,R3      ;POINT TO TABLE OF RAD50 DEV NAMES
011716  012704  000000G                 MOV     #SPLANM,R4      ;POINT TO TABLE OF ASCII DEV NAMES
011722  004737  012730'         2$:     CALL    FORCEO          ;FORCE UNIDENTIFIED UNIT #S TO 0
011726  011302                          MOV     (R3),R2         ;GET RAD50 NAME OF SPOOLED DEVICE
011730  010261  000000G                 MOV     R2,SDNAME(R1)   ;SET NAME IN SDCB
011734  010100                          MOV     R1,R0           ;GET ADDRESS OF SDCB
011736  062700  000000G                 ADD     #SDANAM,R0      ;POINT TO CELL FOR ASCII NAME
011742  112420                          MOVB    (R4)+,(R0)+     ;MOVE IN ASCII DEVICE NAME
011744  112420                          MOVB    (R4)+,(R0)+
011746  112420                          MOVB    (R4)+,(R0)+
011750  020227  000000G                 CMP     R2,#DMYDEV      ;Is this a dummy entry for later patching?
011754  001451                          BEQ     1$              ;Br if yes -- Ignore it
011756  010200                          MOV     R2,R0           ;Get name to R0
011760  004737  012632'                 CALL    CVTDVU          ;Convert name to device # and unit #
011764  010061  000000G                 MOV     R0,SDDVU(R1)    ;Store device # and unit # in SDCB
011770  010200                          MOV     R2,R0           ;Get device name
011772  004737  012526'                 CALL    CHKCLD          ;See if this is a CL device?
011776  103406                          BCS     14$             ;Br if not
012000  004737  012442'                 CALL    SPLCLD          ;Set up for spooling to CL device
012004  103414                          BCS     3$              ;Br if invalid unit
012006  105237  000000G                 INCB    NSPLDV          ;Count # of actual spooled devices
012012  000432                          BR      1$              ;Process next device
012014  010100                  14$:    MOV     R1,R0           ;Get address of SDCB
012016  062700  000000G                 ADD     #SDCHAN,R0      ;Point to channel block within SDCB
012022  004737  027040'                 CALL    OPNCHN          ;Set TSX-Plus channel block open to device
012026  103403                          BCS     3$              ;Br if did not recognize device
012030  105237  000000G                 INCB    NSPLDV          ;Count # actual spooled devices
012034  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.
                                ;
012036  012761  000000G 000000G 3$:     MOV     #DMYDEV,SDNAME(R1);SAY THIS DEVICE IS NOT SPOOLED
```

```
     58 012044  105737  000000G                 TSTB    VINABT          ;ABORT OR CONTINUE ON ERRORS?
     59 012050  001413                          BEQ     1$              ;BR TO IGNORE DEVICE AND CONTINUE INIT
     60 012052                                  .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     61 012060                                  .PRINT  #BDSPOP
     62 012066  010200                          MOV     R2,R0           ;GET RAD50 DEVICE NAME
     63 012070  004737  030012'                 CALL    PRTR50          ;PRINT DEVICE NAME
     64 012074  000137  004216'                 JMP     INISTP          ;ABORT INITIALIZATION
     65                                 ;
     66                                 ;  Process next spooled device
     67                                 ;
     68 012100  062701  000000G         1$:     ADD     #SDCBSZ,R1      ;POINT TO NEXT SDCB
     69 012104  005723                          TST     (R3)+           ;POINT TO NEXT DEVICE NAME
     70 012106  020327  000000G                 CMP     R3,#SPLDVN      ;OPENED ALL SPOOLED DEVICES?
     71 012112  103703                          BLO     2$              ;BR IF MORE TO DO
     72                                 ;
     73                                 ;  Open the spool file
     74                                 ;
     75 012114  105737  000000G                 TSTB    NSPLDV          ;ARE THERE ANY ACTUAL SPOOLED DEVICES?
     76 012120  001521                          BEQ     12$             ;BR IF THERE ARE NO ACTUAL SPOOLED DEVICES
     77 012122  013700  000000G                 MOV     SPLBLK,R0       ;Get name of device for spool file
     78 012126  004737  027506'                 CALL    RTFTCH          ;Fetch the RT-11 device handler
     79 012132  103532                          BCS     11$             ;Br if cannot fetch handler
     80 012134  013702  000000G                 MOV     NSPLBL,R2       ;GET # BLOCKS TO ALLOCATE FOR FILE
     81 012140  005202                          INC     R2              ;Add 1 extra block
     82                                 ;
     83                                 ;  See if spool file already exists
     84                                 ;
     85 012142                          5$:     .LOOKUP #AREA,#1,#SPLBLK;SEE IF SPOOL FILE ALREADY EXISTS
     86 012162  103415                          BCS     6$              ;BR IF IT DOES NOT EXIST
     87 012164  020002                          CMP     R0,R2           ;IS IT THE RIGHT SIZE?
     88 012166  001453                          BEQ     7$              ;BR IF YES
     89 012170                                  .CLOSE  #1              ;IT EXISTS BUT IS OF WRONG SIZE
     90 012176                                  .DELETE #AREA,#1,#SPLBLK;DELETE CURRENT FILE AND OPEN NEW ONE
     91                                 ;
     92                                 ;  Open new spool file
     93                                 ;
     94 012216                          6$:     .ENTER  #AREA,#1,#SPLBLK,R2;CREATE A NEW SPOOL FILE
     95 012242  103456                          BCS     8$              ;BR IF ERROR ON ENTER
     96                                 ;  Write to last block in file to reserve full file space
     97 012244  010203                          MOV     R2,R3           ;Get # of blocks in file
     98 012246  005303                          DEC     R3              ;Get # of last block in file
     99 012250                                  .WRITW  #AREA,#1,#TSINIT,#256.,R3
    100                                 ;  Now close and reopen using a lookup
    101 012306                                  .CLOSE  #1              ;CLOSE SPOOL FILE
    102 012314  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 012316  012700  000000G         7$:     MOV     #SPLCHN,R0      ;SAVE CHANNEL STATUS HERE
    108 012322  013702  000000G                 MOV     SPLBLK,R2       ;GET DEVICE NAME
    109 012326  004737  027126'                 CALL    SETCHN          ;SAVE CHANNEL STATUS
    110 012332                                  .RELEAS #SPLBLK         ;Release the RT-11 device handler
    111                                 ;
    112                                 ;  Set number of free public blocks in spool file
    113                                 ;
    114 012342  113703  000000G                 MOVB    NSPLDV,R3       ;Get # spooled devices
```

```
    115 012346  070327  000000G                 MUL     #PVSPBL,R3      ;Times number of private blocks per dev
    116 012352  005403                          NEG     R3
    117 012354  063703  000000G                 ADD     NSPLBL,R3               ;Get # public spool blocks
    118 012360  010337  000000G                 MOV     R3,NFRESB       ;This is number of public free spool blocks
    119                             ;
    120                             ;  Finished
    121                             ;
    122 012364  012605              12$:        MOV     (SP)+,R5
    123 012366  012604                          MOV     (SP)+,R4
    124 012370  012603                          MOV     (SP)+,R3
    125 012372  012602                          MOV     (SP)+,R2
    126 012374  012601                          MOV     (SP)+,R1
    127 012376  000207              9$:         RETURN
    128                             ;
    129                             ;  Error: Cannot open spool file.
    130                             ;
    131 012400                      8$:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
    132 012406                                  .PRINT  #BOSF
    133 012414  000137  004216'                 JMP     INISTP          ;ABORT INITIALIZATION
    134                             ;
    135                             ;  Error: Invalid device specification.
    136                             ;
    137 012420  010001              11$:        MOV     R0,R1           ;Save device name
    138 012422                                  .PRINT  #TSXHD          ;Print error message
    139 012430                                  .PRINT  #BOSF
    140 012436  004737  011324'                 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
                                ;
012442  010546                  SPLCLD: MOV     R5,-(SP)
                                ;
                                ;  Make sure CL unit number is valie
                                ;
012444  010005                          MOV     R0,R5           ;Get CL unit number
012446  020527  000000G                 CMP     R5,#CLTOTL      ;Is this a valid unit #
012452  103022                          BHIS    8$              ;Br if invalid
                                ;
                                ;  Set up channel control block in SDCB
                                ;
012454  005061  000000C                 CLR     SDCHAN+C.SBLK(R1);Starting block # = 0
012460  020527  000007                  CMP     R5,#7           ;Is this a CL or C1 unit?
012464  101405                          BLOS    1$              ;Br if CL unit
012466  162705  000010                  SUB     #8.,R5          ;Remove C1 unit # bias
012472  013700  000000G                 MOV     C1DEVX,R0       ;Get C1 device index number
012476  000402                          BR      2$
012500  013700  000000G         1$:     MOV     CLDEVX,R0       ;Get CL device index number
012504  010061  000000C         2$:     MOV     R0,SDCHAN+C.CSW(R1);Set device index number
012510  110561  000000C                 MOVB    R5,SDCHAN+C.DEVQ(R1);Set unit #
                                ;
                                ;  We successfully set up a CL unit
                                ;
012514  000241                          CLC                     ;Signal success on error
012516  000401                          BR      9$
                                ;
                                ;  We cannot open this CL unit
                                ;
012520  000261                  8$:     SEC                     ;Signal error
                                ;
                                ;  Finished
                                ;
012522  012605                  9$:     MOV     (SP)+,R5
012524  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)
                                ;
012526                          CHKCLD:
                                ;
                                ;  See if this is a CL unit
                                ;
012526  020037  000202'                 CMP     R0,R50CL        ;Is name "CL"?
012532  001411                          BEQ     1$              ;Br if yes
012534  020037  000204'                 CMP     R0,R50CL0       ;Is name in the range CL0 to CL7?
012540  103432                          BLO     8$              ;Br if not
012542  020037  000206'                 CMP     R0,R50CL7
012546  101005                          BHI     2$
012550  163700  000204'                 SUB     R50CL0,R0       ;Get CL unit number
012554  000422                          BR      7$
012556  005000                  1$:     CLR     R0              ;CL = CL0
012560  000420                          BR      7$
                                ;
                                ;  See if this is a C1 unit
                                ;
012562  020037  000210'          2$:     CMP     R0,R50C1        ;Is name "C1"?
012566  001413                          BEQ     3$              ;Br if yes
012570  020037  000212'                 CMP     R0,R50C10       ;Is name in the range C10 to C17?
012574  103414                          BLO     8$              ;Br if not
012576  020037  000214'                 CMP     R0,R50C17
012602  101011                          BHI     8$
012604  163700  000212'                 SUB     R50C10,R0       ;Get unit number
012610  062700  000010                  ADD     #8.,R0          ;Bias by 8 for C1 units
012614  000402                          BR      7$
012616  012700  000010          3$:     MOV     #8.,R0          ;C1 = CL8
                                ;
                                ;  This is a CL or C1 unit
                                ;
012622  000241                  7$:     CLC                     ;Signal success on return
012624  000401                          BR      9$
                                ;
                                ;  This is not a CL or C1 unit
                                ;
012626  000261                  8$:     SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
012630  000207                  9$:     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).
                                ;
012632  010246                  CVTDVU: MOV     R2,-(SP)
012634  010346                          MOV     R3,-(SP)
                                ;
                                ;  Split the unit number off of the full device name
                                ;
012636  010003                          MOV     R0,R3           ;Get full device name
012640  005002                          CLR     R2              ;Set up for divide
012642  071227  000050                  DIV     #50,R2          ;Split name and unit (R0=name, R1=unit)
012646  005703                          TST     R3              ;Was a unit number specified?
012650  001402                          BEQ     1$              ;Br if not
012652  162703  000036                  SUB     #36,R3          ;Convert unit number to binary value
012656  010300                  1$:     MOV     R3,R0           ;Get unit number
012660  000300                          SWAB    R0              ;Position to high-order byte
                                ;
                                ;  Look up the device name to get the device index
                                ;
012662  070227  000050                  MUL     #50,R2          ;Now get the device name without unit number
012666  013702  000000G                 MOV     NUMDEV,R2       ;Get index number of last device
012672  020362  000000G         2$:     CMP     R3,PNAME(R2)    ;Search for device in name table
012676  001407                          BEQ     3$              ;Br if found it
012700  162702  000002                  SUB     #2,R2           ;Try next device
012704  002372                          BGE     2$              ;Loop if more to check
                                ;
                                ;  Error, cannot find device name in tables
                                ;
012706  012700  177777                  MOV     #-1,R0          ;Set device # = unit # = -1
012712  000261                          SEC                     ;Signal error on return
012714  000402                          BR      9$
                                ;
                                ;  Found the device in the tables
                                ;
012716  050200                  3$:     BIS     R2,R0           ;Combine device # and unit #
012720  000241                          CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
012722  012603                  9$:     MOV     (SP)+,R3
012724  012602                          MOV     (SP)+,R2
012726  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
                                ;
012730  010346                  FORCE0: MOV     R3,-(SP)
012732  010446                          MOV     R4,-(SP)
012734  010546                          MOV     R5,-(SP)
012736  011305                          MOV     (R3),R5         ;MOVE CURRENT DEV NAME TO R5
012740  005004                          CLR     R4              ;SET UP FOR DIVIDE
012742  071427  000050                  DIV     #50,R4          ;SEPARATE INTO NAME AND UNIT
012746  005705                          TST     R5              ;WAS 3RD CHAR BLANK?
012750  001012                          BNE     9$              ;RETURN IF NOT
012752  010405                          MOV     R4,R5           ;GET HIGH 2 CHARS
012754  005004                          CLR     R4              ;SET UP FOR ANOTHER DIVIDE
012756  071427  000050                  DIV     #50,R4          ;SEPARATE 1 & 2 CHARS
012762  005704                          TST     R4              ;WAS CHAR 1 BLANK?
012764  001404                          BEQ     9$              ;EMPTY OR INVALID DEV NAME!
012766  005705                          TST     R5              ;WAS CHAR 2 BLANK?
012770  001402                          BEQ     9$              ;1-CHAR DEV NAME SHOULD BE INVALID???
012772  062713  000036                  ADD     #^R  0,(R3)     ;FORCE TO UNIT 0
012776  012605                  9$:     MOV     (SP)+,R5
013000  012604                          MOV     (SP)+,R4
013002  012603                          MOV     (SP)+,R3
013004  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.
                                ;
013006  010146                  ALOCBF: MOV     R1,-(SP)
                                ;
                                ;  Assign space for I/O queue elements.
                                ;
013010  010537  000000G                 MOV     R5,FREIOQ       ;START OF I/O QUEUE SPACE
013014  062705  000000C                 ADD     #IOQSIZ*NUMIOQ,R5;RESERVE SPACE FOR I/O QUEUE ELEMENTS
                                ;
                                ;  Assign space for shared PLAS region control blocks
                                ;
013020  010537  000000G                 MOV     R5,SHRRCB       ;Start of area for RCB's
013024  013701  000000G                 MOV     VNGR,R1         ;Get number of RCB's wanted
013030  020137  000000G                 CMP     R1,VMXWIN       ;Must have one for each display window
013034  103002                          BHIS    13$             ;Br if ok
013036  013701  000000G                 MOV     VMXWIN,R1       ;Force one for each window
013042  070127  000000G         13$:    MUL     #RC$$SZ,R1      ;Multiply by size of each block
013046  060105                          ADD     R1,R5           ;Allocate space for RCB's
013050  010537  000000G                 MOV     R5,SHRRCN       ;Address of end of region
                                ;
                                ;  Assign space for fork blocks
                                ;
013054  012700  000000C                 MOV     #<NUMFRK-FRKGEN>,R0 ;Get # fork blocks to allocate
013060  003404                          BLE     11$             ;Br if none to allocate
013062  010537  000000G                 MOV     R5,FRKINI       ;Set pointer to start of area
013066  062705  000000C                 ADD     #<<NUMFRK-FRKGEN>*FQ$$SZ>,R5 ;Reserve space for fork blocks
                                ;
                                ;  Assign space for job monitoring control blocks
                                ;
013072  013701  000000G         11$:    MOV     VMXMON,R1       ;Any job monitoring blocks wanted?
013076  001405                          BEQ     10$             ;Br if not
013100  010537  000000G                 MOV     R5,MONFQH       ;Start of job monitoring control blocks
013104  070127  000000G                 MUL     #JM$$SZ,R1      ;Compute space needed for control blocks
013110  060105                          ADD     R1,R5           ;Allocate the space
                                ;
                                ;  Allocate space for system message buffers.
                                ;
013112  010537  000000G         10$:    MOV     R5,SNMSHD       ;HEAD OF SYSTEM MESSAGE BUFFER AREA
013116  062705  000000C                 ADD     #<NMSNMB*SB$$SZ>,R5;RESERVE ROOM FOR MESSAGE BUFFERS
                                ;
                                ;  Allocate space for INSTALLed program table
                                ;
013122  010537  000000G                 MOV     R5,INSTBL       ;Base of table
013126  013701  000000G                 MOV     VNUIP,R1        ;# slots for user installed programs
013132  062701  000000G                 ADD     #NSIP,R1        ;Add # slots for system programs
013136  070127  000000G                 MUL     #II$$SZ,R1      ;Multiply by size of each slot
```

```
                                        ADD     R1,R5           ;Allocate space for table
013144  010537  000000G                MOV     R5,INSTBN       ;Pointer past end of INSTALL table
                                ;
                                ;  Allocate space for device mount entries
                                ;
013150  010537  000000G                 MOV     R5,CSHDEV       ;Point to start of area
013154  012701  000000G                 MOV     #CD$$SZ,R1      ;Get size of each entry
013160  070137  000000G                 MUL     VMXCSH,R1       ;Multiply by number of entries
013164  060105                          ADD     R1,R5           ;Reserve space for table
013166  010537  000000G                 MOV     R5,CSHDVN       ;Save pointer past end of table
                                ;
                                ;  Allocate space for data cache control blocks
                                ;
013172  005737  000000G                 TST     CSHALC          ;Is data caching wanted?
013176  001404                          BEQ     12$             ;Br if not
013200  010537  000000G                 MOV     R5,CCBHD        ;Head of free list area
013204  062705  000000C                 ADD     #NUMCCB*CC$$SZ,R5 ;Allocate space for control blocks
                                ;
                                ;  Allocate space for spool file control blocks
                                ;
013210  013701  000000G         12$:    MOV     NSPLFL,R1       ;Get # spool file control blocks needed
013214  001407                          BEQ     1$              ;Br if none needed
013216  070127  000000G                 MUL     #SFCBSZ,R1      ;Compute space needed by control blocks
013222  010537  000000G                 MOV     R5,SFCB         ;Base of control block area
013226  060105                          ADD     R1,R5           ;Allocate space for control blocks
013230  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.
                                ;
013234  012701  000002          1$:     MOV     #2,R1           ;START WITH FIRST MUX
013240  020127  000000G         2$:     CMP     R1,#LSTMX       ;HAVE WE DONE ALL MUX'S?
013244  101007                          BHI     5$              ;BR IF YES
013246  010561  000000G                 MOV     R5,MXLNT(R1)    ;SET ADDRESS OF START OF VECTOR
013252  062705  000020                  ADD     #16.,R5         ;RESERVE SPACE FOR VECTOR
013256  062701  000002                  ADD     #2,R1           ;ADVANCE TO NEXT MUX
013262  000766                          BR      2$
                                ;
                                ;  Allocate buffers to hold characters for DMA transfers to DH11 multiplexers
                                ;
013264  012701  000002          5$:     MOV     #2,R1           ;Start with first line
013270  020127  000000G         6$:     CMP     R1,#LSTPL       ;Is this a primary time-sharing line?
013274  101403                          BLOS    3$              ;Br if yes
013276  020127  000000G                 CMP     R1,#FSTIOL      ;Is this a CL line?
013302  103422                          BLO     7$              ;Br if not
013304  026127  000000G 000000G 3$:     CMP     LCDTYP(R1),#CDX$DH ;Is this line connected to a DH11?
013312  001404                          BEQ     8$              ;Br if yes
013314  026127  000000G 000000G         CMP     LCDTYP(R1),#CDX$VH ;Is this line connected to a DHV11?
013322  001012                          BNE     7$              ;Br if not
013324  010561  000000G         8$:     MOV     R5,LDHB1B(R1)   ;Set address of start of buffer 1
013330  010561  000000G                 MOV     R5,LDHB1P(R1)   ;Initialize pointer into buffer 1
013334  062705  000000G                 ADD     #DHBFSZ,R5      ;Reserve space for buffer
013340  010561  000000G                 MOV     R5,LDHB2B(R1)   ;Set address of start of buffer 2
013344  062705  000000G                 ADD     #DHBFSZ,R5      ;Reserve space for buffer
013350  062701  000002          7$:     ADD     #2,R1           ;Get # of next line
013354  020127  000000G                 CMP     R1,#LSTHL       ;Have we checked all lines?
```

```
    115 013360  101743                          BLOS    6$              ;Br if not
    116 013362  005205                          INC     R5              ;Bound up to next word
    117 013364  042705  000001                  BIC     #1,R5
    118                                 ;
    119                                 ;  Allocate space for tables that keep track of space in swap file
    120                                 ;
    121 013370  105737  000000G                 TSTB    VSWPFL          ;Is this a swapping system?
    122 013374  001415                          BEQ     14$             ;Br if not
    123 013376  013700  000000G                 MOV     VSWPSL,R0       ;Get # slots in swap file
    124 013402  006300                          ASL     R0              ;Allocate 2 bytes per slot
    125 013404  010537  000000G                 MOV     R5,SWPPOS       ;Start of table with starting block #'s
    126 013410  060005                          ADD     R0,R5           ;Allocate space
    127 013412  010537  000000G                 MOV     R5,SWPJOB       ;Start of table with job #'s
    128 013416  060005                          ADD     R0,R5           ;Allocate space
    129 013420  010537  000000G                 MOV     R5,SCPFHD       ;Pointer to area with command packets
    130 013424  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 013430  010537  000000G         14$:    MOV     R5,CXTBUF       ;Set address of buffer
    135 013434  062705  001400                  ADD     #1400,R5        ;Reserve space for the buffer
    136                                 ;
    137                                 ;  Make sure TSX is not too big.
    138                                 ;
    139 013440  010500                          MOV     R5,R0           ;GET CURRENT MEMORY ADDRESS
    140 013442  004737  027622'                 CALL    CHKMEM          ;CHECK FOR SPACE OVERFLOW
    141                                 ;
    142                                 ;  Finished
    143                                 ;
    144 013446  012601                          MOV     (SP)+,R1
    145 013450  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.
                                ;
013452  010146                  ALCSLO: MOV     R1,-(SP)
013454  010246                          MOV     R2,-(SP)
                                ;
                                ;  Begin loop to check each line
                                ;
013456  012701  000000G                 MOV     #LSTHL,R1       ;Get index to last hardware line
                                ;
                                ;  Only allocate silo buffers for real lines
                                ;
013462  012702  000040          1$:     MOV     #32.,R2         ;Set minimum size for time-sharing lines
013466  020127  000000G                 CMP     R1,#LSTPL       ;Is this a primary line?
013472  101405                          BLOS    2$              ;Br if yes
013474  020127  000000G                 CMP     R1,#FSTIOL      ;Is this a CL line?
013500  103463                          BLO     9$              ;Br if not
013502  012702  000020                  MOV     #16.,R2         ;Set minimum size for CL lines
                                ;
                                ;  Determine how much space to allocate
                                ;
013506  016100  000000G         2$:     MOV     LHIRBA(R1),R0   ;Get requested size
013512  001002                          BNE     8$              ;Br if a size was specified
013514  013700  000000G                 MOV     VNCSLO,R0       ;Use default value
013520  020027  000000G         8$:     CMP     R0,#MAXSLO      ;Constrain size to 255 bytes
013524  101402                          BLOS    3$              ;Br if ok
013526  012700  000000G                 MOV     #MAXSLO,R0      ;Reduce size
013532  020002                  3$:     CMP     R0,R2           ;Compare with acceptable minimum
013534  103001                          BHIS    4$              ;Br if ok
013536  010200                          MOV     R2,R0           ;Use minimum size allowed
013540  010061  000000G         4$:     MOV     R0,LHIRBA(R1)   ;Set # bytes to be allocated for silo
                                ;
                                ;  Allocate the space
                                ;
013544  010561  000000G                 MOV     R5,LHIRBB(R1)   ;Set base address of buffer
013550  010561  000000G                 MOV     R5,LHIRBP(R1)   ;Init pointer where to store next char
013554  010561  000000G                 MOV     R5,LHIRBG(R1)   ;Init pointer where to get next char
013560  010061  000000G                 MOV     R0,LHIRBS(R1)   ;Set # free bytes in buffer
013564  060005                          ADD     R0,R5           ;Allocate space for buffer
013566  010561  000000G                 MOV     R5,LHIRBE(R1)   ;Set address beyond end of buffer
                                ;
                                ;  Set up control information about when to send XON and XOFF
                                ;
013572  006200                          ASR     R0              ;Get 1/2 of buffer size
013574  162700  000002                  SUB     #2,R0           ;Minus two characters
013600  116102  000000G                 MOVB    LHIRBC(R1),R2   ;Get specified size for XOFF point
013604  001002                          BNE     5$              ;Br if value specified
013606  113702  000000G                 MOVB    VNCXOF,R2       ;Try default
013612  020200                  5$:     CMP     R2,R0           ;Is specified value ok?
```

```
     58 013614  101401                          BLOS    6$                ;Br if yes
     59 013616  010002                          MOV     R0,R2             ;No, use size/2-2
     60 013620  110261  000000G        6$:      MOVB    R2,LHIRBC(R1)     ;Set # of chars when XOFF sent
     61 013624  116102  000001G                 MOVB    LHIRBC+1(R1),R2   ;Get specified size for XON point
     62 013630  001002                          BNE     7$                ;Br if a value was specified
     63 013632  113702  000000G                 MOVB    VNCXON,R2         ;Try default
     64 013636  020200                  7$:     CMP     R2,R0             ;Is specified value ok?
     65 013640  101401                          BLOS    10$               ;Br if ok
     66 013642  010002                          MOV     R0,R2             ;No, use size/2-2
     67 013644  110261  000001G        10$:     MOVB    R2,LHIRBC+1(R1)   ;Set # of chars when XON sent
     68                                         ;
     69                                         ;  Do the next line
     70                                         ;
     71 013650  162701  000002          9$:     SUB     #2,R1             ;Get next line index number
     72 013654  003302                          BGT     1$                ;Loop if more to do
     73                                         ;
     74                                         ;  Finished
     75                                         ;
     76 013656  005205                          INC     R5                ;Force R5 to be even
     77 013660  042705  000001                  BIC     #1,R5
     78 013664  012602                          MOV     (SP)+,R2
     79 013666  012601                          MOV     (SP)+,R1
     80 013670  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.
                                ;
013672  010146                  ALBFX:  MOV     R1,-(SP)
013674  010246                          MOV     R2,-(SP)
013676  010346                          MOV     R3,-(SP)
                                ;
                                ;  Allocate character buffers for all lines
                                ;  Note: Character buffer space will be accessed by mapping through PAR 6.
                                ;
013700  012701  000002                  MOV     #2,R1           ;GET 1ST JOB INDEX NUMBER
013704  032761  000000G 000000G 3$:      BIT     #$DEAD,LSW3(R1) ;IS THIS LINE INSTALLED?
013712  001047                          BNE     2$              ;BR IF NOT -- DON'T ALLOCATE ANY BUFFER SPACE
013714  020127  000000G                  CMP     R1,#FSTDL       ;IS THIS A DETACHED JOB LINE?
013720  103403                          BLO     1$              ;BR IF NOT
013722  020127  000000G                  CMP     R1,#LSTDL       ;DETACHED JOB LINE?
013726  101441                          BLOS    2$              ;BR IF DETACHED JOB -- DON'T ALLOCATE  BUFFERS
013730  010561  000000G          1$:     MOV     R5,LTTPAR(R1)   ;SET PHYSICAL MEMORY PAR OFFSET FOR BUFFER
013734  012702  000000G                  MOV     #VPAR6,R2       ;GET VIRTUAL MEMORY ADDRESS FOR BASE OF PAR6
013740  010261  000000G                  MOV     R2,LINBUF(R1)   ;INPUT BUFFER STARTS AT BASE OF PAR6 REGION
013744  016100  000000G                  MOV     LINSIZ(R1),R0   ;GET # BYTES FOR INPUT BUFFER
013750  010061  000000G                  MOV     R0,LINSPC(R1)   ;SET # FREE BYTES IN INPUT BUFFER
013754  060002                          ADD     R0,R2           ;ADVANCE VIRTUAL ADDRESS
013756  010261  000000G                  MOV     R2,LINEND(R1)   ;POINTS PAST END OF INPUT BUFFER
013762  010003                          MOV     R0,R3           ;Get # bytes in input buffer
013764  062703  000007                  ADD     #7,R3           ;Bound up to multiple of 8
013770  072327  177775                  ASH     #-3,R3          ;Get # bytes needed in activation-flag buffer
013774  060302                          ADD     R3,R2           ;Reserve space for activation-flag buffer
013776  060300                          ADD     R3,R0           ;Accumulate total buffer space
014000  010261  000000G                  MOV     R2,LOTBUF(R1)   ;POINTS TO START OF OUTPUT BUFFER AREA
014004  066100  000000G                  ADD     LOTSIZ(R1),R0   ;ACCUMULATE # BYTES IN BOTH BUFFERS
014010  066102  000000G                  ADD     LOTSIZ(R1),R2   ;ADVANCE VIRTUAL ADDRESS COUNTER
014014  010261  000000G                  MOV     R2,LOTEND(R1)   ;SAVE ADDRESS OF END OF OUTPUT BUFFER
014020  062700  000077                  ADD     #77,R0          ;BOUND UP TO MULTIPLE OF 64 BYTES
014024  072027  177772                  ASH     #-6,R0          ;CONVERT TO # 64-BYTE BLOCKS ALLOCATED
014030  060005                          ADD     R0,R5           ;ADVANCE PHYSICAL MEMORY PAR ADDRESS
014032  062701  000002          2$:     ADD     #2,R1           ;ADVANCE LINE NUMBER
014036  020127  000000G                  CMP     R1,#LSTSL       ;HAVE WE DONE ALL LINES YET?
014042  101720                          BLOS    3$              ;BR IF MORE TO DO
                                ;
                                ;  Allocate space for shared file record locking data structures
                                ;
014044  005737  000000G                  TST     VMXSF           ;Shared file support wanted?
014050  001451                          BEQ     5$              ;Br if not
014052  013737  000000G 000000G          MOV     VNUMDC,NUMDCD   ;Set number of data cache blocks
014060  013737  000000G 000000G          MOV     VMXSFC,NUMCDB   ;Set number of free CDB's
014066  010537  000000G                  MOV     R5,LOKMEM       ;Set phys address of base of area
014072  012701  000000G                  MOV     #FF$$SZ,R1      ;Size of an FDB
014076  070137  000000G                  MUL     VMXSF,R1        ;Times number of FDB's
```

```
     58 014102  062701  000000C                 ADD     #<NLINES*FW$$SZ>,R1 ;Space needed for wait blocks
     59 014106  013703  000000G                 MOV     VMLBLK,R3           ;Max blocks a CDB may hold locked
     60 014112  006303                          ASL     R3                  ;Two bytes per entry
     61 014114  062703  000000G                 ADD     #FC$LBN,R3          ;Add base size of a CDB
     62 014120  070337  000000G                 MUL     VMXSFC,R3           ;Times number of shared file channels
     63 014124  060301                          ADD     R3,R1               ;Accumulate space needed
     64 014126  012703  000000G                 MOV     #DC$$SZ,R3          ;Size of a data cache descriptor
     65 014132  070337  000000G                 MUL     VNUMDC,R3           ;Times number of data cache entries
     66 014136  060301                          ADD     R3,R1               ;Reserve space for data cache descriptors
     67 014140  062701  000100                  ADD     #64.,R1             ;Bound up to 64 byte unit
     68 014144  072127  177772                  ASH     #-6,R1              ;Convert to # 64 byte units
     69 014150  042701  176000                  BIC     #176000,R1          ;Clear sign extension
     70 014154  060105                          ADD     R1,R5               ;Reserve space for data structures
     71 014156  010537  000000G                 MOV     R5,LOKCSH           ;Save pointer to start of cache buffer area
     72 014162  013701  000000G                 MOV     VNUMDC,R1           ;# shared-file data cache blocks wanted
     73 014166  070127  000010                  MUL     #8.,R1              ;8 64-byte blocks each (512 bytes each)
     74 014172  060105                          ADD     R1,R5               ;Reserve space for data cache buffers
     75                                 ;
     76                                 ;  Allocate space for mapped I/O buffers
     77                                 ;
     78 014174  105737  000000G         5$:     TSTB    MIOFLG              ;Are any mapped I/O buffers needed?
     79 014200  001415                          BEQ     7$                  ;Br if not
     80 014202  013701  000000G                 MOV     MIOBHD,R1           ;Point to 1st mapped I/O control block
     81 014206  001412                          BEQ     7$                  ;Br if no more buffers needed
     82 014210  010561  000000G         6$:     MOV     R5,MI$SBP(R1)       ;Set address of base of buffer
     83 014214  113703  000000G                 MOVB    VMIOSZ,R3           ;Get # blocks for buffer
     84 014220  072327  000003                  ASH     #3,R3               ;Convert to # 64-byte pages
     85 014224  060305                          ADD     R3,R5               ;Allocate space for buffer
     86 014226  016101  000000G                 MOV     MI$LNK(R1),R1       ;Get address of next control block
     87 014232  001366                          BNE     6$                  ;Loop if more to allocate
     88                                 ;
     89                                 ;  Allocate space for performance monitor data buffer if it is wanted.
     90                                 ;
     91 014234  012701  000000G         7$:     MOV     #PMSIZE,R1          ;DID USER GEN IN PERFORMANCE MONITOR FEATURE?
     92 014240  001407                          BEQ     9$                  ;BR IF NOT
     93 014242  010537  000000G                 MOV     R5,PMPAR            ;SET BASE ADDRESS OF PM BUFFER
     94 014246  062701  000077                  ADD     #77,R1              ;BOUND SIZE UP TO 64-BYTE MULTIPLE
     95 014252  072127  177772                  ASH     #-6,R1              ;CONVERT TO # 64-BYTE BLOCKS
     96 014256  060105                          ADD     R1,R5               ;ADVANCE FREE MEMORY POINTER
     97                                 ;
     98                                 ;  Finished
     99                                 ;
    100 014260  012603                  9$:     MOV     (SP)+,R3
    101 014262  012602                          MOV     (SP)+,R2
    102 014264  012601                          MOV     (SP)+,R1
    103 014266  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.
                                ;
014270  010246                  OPNKMN: MOV     R2,-(SP)
                                ;
                                ;  Lookup TSKMON file.
                                ;
                                        .LOOKUP #AREA,#1,#KMNNAM ;TRY TO FILE KMON SAV FILE
014312  103517                          BCS     9$              ;BR IF NOT THERE
                                ;
                                ;  Read block 0 of save file and extract some information.
                                ;
014314  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
014320                                  .READW  #AREA,#1,R2,#256.,#0 ;READ BLOCK 0 OF SAV FILE
                                ;  Determine size of kmon
014354  016200  000050                  MOV     50(R2),R0       ;GET TOP ADDRESS OF KMON
014360  062700  000003                  ADD     #3,R0           ;BOUND UP TO NEXT WORD
014364  042700  000001                  BIC     #1,R0           ;FORCE EVEN
014370  010037  000000G                 MOV     R0,KMNTOP
                                ;  Determine number of 256-word memory pages needed while kmon running.
014374  162700  000000G                 SUB     #KMNBAS,R0      ;BASE ADDRESS OF KMON
014400  010037  000000G                 MOV     R0,KMNHI        ;TOP OF TSKMON - KMNBAS
014404  062700  000777                  ADD     #511.,R0        ;BOUND UP TO PAGE SIZE
014410  000241                          CLC
014412  006000                          ROR     R0              ;CVT TO # WORDS
014414  000300                          SWAB    R0              ;CVT TO # PAGES
014416  042700  177400                  BIC     #^C377,R0
014422  063700  000000G                 ADD     CXTPAG,R0       ;# PAGES NEEDED FOR JOB CONTEXT AREA
014426  010037  000000G                 MOV     R0,KMNPGS       ;# 256-wd pages needed for kmon + context area
                                ;  Determine Kmon stack pointer.
014432  016237  000042  000000G         MOV     42(R2),KMNSTK   ;INITIAL STACK POINTER FOR KMON
                                ;  Determine Kmon starting address.
014440  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)
                                ;
014446  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.
                                ;
014454  012700  000000G                 MOV     #KMNCHN,R0      ;GET KMON CHANNEL SAVE AREA
014460  013702  000072'                 MOV     KMNNAM,R2       ;GET KMON DEVICE NAME
```

```
     58 014464  004737  027126'                  CALL    SETCHN          ;SAVE CHANNEL STATUS
     59                                  ;
     60                                  ;  Lookup CCL.SAV and save channel status for it.
     61                                  ;
     62 014470                                    .LOOKUP #AREA,#1,#CCLNAM;LOOKUP SY:CCL.SAV
     63 014510  103410                            BCS     8$              ;BR IF CAN'T FIND CCL
     64 014512  012700  000000G                   MOV     #CCLSAV,R0      ;CHANNEL SAVE AREA
     65 014516  013702  000102'                   MOV     CCLNAM,R2       ;DEVICE NAME
     66 014522  004737  027126'                   CALL    SETCHN          ;SAVE CHANNEL STATUS
     67                                  ;
     68                                  ;  Finished
     69                                  ;
     70 014526  012602                   10$:     MOV     (SP)+,R2
     71 014530  000207                            RETURN
     72                                  ;
     73                                  ;  Error:  We could not find SY:CCL.SAV
     74                                  ;
     75 014532                           8$:      .PRINT  #TSXHD
     76 014540                                    .PRINT  #NOCCL          ;PRINT ERROR MESSAGE
     77 014546  000137  004216'                   JMP     INISTP          ;ABORT INITIALIZATION
     78                                  ;
     79                                  ;  Error: We could not locate TSKMON SAV file.
     80                                  ;
     81 014552                           9$:      .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     82 014560                                    .PRINT  #NOKMON
     83 014566  000137  004216'                   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.
                                ;
014572  010146                  CLINIT: MOV     R1,-(SP)
014574  010246                          MOV     R2,-(SP)
014576  010346                          MOV     R3,-(SP)
                                ;
                                ;  Initialize tables for each CL unit
                                ;
014600  005003                          CLR     R3              ;Accumulate ring buffer sizes in R3
014602  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.
                                ;
014606  016102  000000G         1$:     MOV     CL$LIX(R1),R2   ;Is this CL unit associated with a line?
014612  001416                          BEQ     5$              ;Br if not
014614  012762  000000G 000000G        MOV     #$SXON,LSW10(R2);Send XON when we start the line
014622  010162  000000G                MOV     R1,LCLUNT(R2)   ;Associate the CL unit with this line
014626  005762  000000G                TST     RSR(R2)         ;Does this unit have a specified RSR addr?
014632  001006                          BNE     5$              ;Br if yes
014634  005762  000000G                TST     LMXNUM(R2)      ;Is this a mux line?
014640  001003                          BNE     5$              ;Br if yes
014642  005061  000000G                CLR     CL$EPS(R1)      ;Say no endstring buffer
014646  000472                          BR      4$              ;Line is not genned in
                                ;
                                ;  Allocate and set up pointers for the output ring buffers
                                ;
014650  010561  000000G         5$:     MOV     R5,CL$ORB(R1)   ;Start of output ring buffer
014654  010561  000000G                MOV     R5,CL$ORP(R1)   ;Input character pointer
014660  010561  000000G                MOV     R5,CL$ORG(R1)   ;Next available character pointer
014664  012700  000000G                MOV     #CLORSZ,R0      ;Get default output ring buffer size
014670  005702                          TST     R2              ;Is this CL unit connected to a line?
014672  001402                          BEQ     6$              ;Br if not
014674  016200  000000G                MOV     LOTSIZ(R2),R0   ;Get size of output ring buffer
014700  010061  000000G         6$:     MOV     R0,CL$ORA(R1)   ;Set size of output ring buffer
014704  010061  000000G                MOV     R0,CL$ORS(R1)   ;Available space in ring buffer
014710  060005                          ADD     R0,R5           ;Point beyond end of ring buffer
014712  010561  000000G                MOV     R5,CL$ORE(R1)   ;Address past end of ring buffer
014716  060003                          ADD     R0,R3           ;Accumulate size of output ring buffers
                                ;
                                ;  Allocate space for end-of-file string buffer
                                ;
014720  010561  000000G                 MOV     R5,CL$EPS(R1)   ;Set pointer to end-of-file string buffer
014724  005061  000000G                 CLR     CL$EPP(R1)      ;No string to print yet
014730  062705  000010G                 ADD     #<CLEOFS+1>,R5  ;Reserve space for buffer
014734  062703  000010G                 ADD     #<CLEOFS+1>,R3  ;Accumulate buffer space
                                ;
                                ;  Initialize end-of-file form-feed count
                                ;
```

```
                                        CLR     CL$EPN(R1)      ;Init ENDPAGE=0
                                ;
                                ;  Initialize option word
                                ;
                                        MOV     #<CO$DEF>,R0    ;Get default option flags
                                        TST     R2              ;Is this CL unit connected with a line?
                                        BEQ     7$              ;Br if not
                                        MOV     ILSW2(R2),R2    ;Get line options
                                        BIT     #$TAB,R2        ;Does hardware support tabs?
                                        BEQ     2$              ;Br if not
                                        BIS     #CO$TAB,R0      ;Set hardware-tab flag
                                2$:     BIT     #$FORM,R2       ;Does hardware support form feeds?
                                        BEQ     3$              ;Br if not
                                        BIS     #CO$FF,R0       ;Set hardware-form-feed flag
                                3$:     BIT     #$8BIT,R2       ;Does hardware want 8 bit support?
                                        BEQ     7$              ;Br if not
                                        BIS     #CO$8BT,R0      ;Enable 8 bit support for CL line
                                7$:     MOV     R0,CL$OPT(R1)   ;Set options for this CL line
                                ;
                                ;  Initialize page length
                                ;
                                        MOV     #66.,CL$LEN(R1) ;Say page length = 66 lines
                                ;
                                ;  Initialize status flags
                                ;
                                        CLR     CL$STA(R1)      ;Initialize status flags
                                ;
                                ;  Do next line
                                ;
                                4$:     SUB     #2,R1           ;Get index # of next unit
                                        BGE     1$              ;Loop if more units to initialize
                                ;
                                ;  Make a device table entry for "CL" device
                                ;
                                        ADD     #2,NUMDEV       ;One more device
                                        MOV     NUMDEV,R1       ;Get device table index
                                        MOV     R1,CLDEVX       ;Remember index number of CL device
                                        MOV     R50CL,PNAME(R1) ;Set device name ("CL")
                                        MOV     #CLSTS,DVSTAT(R1) ;Set dev status flags
                                        MOV     #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ;Device info flags
                                        CLR     DEVSIZ(R1)      ;Clear device size
                                        MOV     #CLHEAD+6,HANENT(R1) ;Set handler entry point (4th word)
                                        ADD     #CLSIZE,R3      ;Get size of handler
                                        ADD     #<<CLTOTL*46.)+<NIOL*32.>>,R3 ;Add size of tables in TSGEN
                                        MOV     R3,HANSIZ(R1)   ;Set size of handler
                                        INC     R5              ;Make sure free-memory pointer is even
                                        BIC     #1,R5
                                ;
                                ;  Make a device table entry for C1 if there are more than 8 CL units
                                ;
                                        CMP     #CLTOTL,#8.     ;Are there more than 8 CL units?
                                        BLOS    13$             ;Br if not -- Don't need C1
                                        ADD     #2,NUMDEV       ;One more device
                                        MOV     NUMDEV,R1       ;Get device table index
                                        MOV     R1,C1DEVX       ;Remember index number of CL device
                                        MOV     R50C1,PNAME(R1) ;Set device name ("C1")
                                        MOV     #CLSTS,DVSTAT(R1) ;Set dev status flags
```

```
     115 015200  012761  000000C 000000G          MOV     #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ;Device info flags
     116 015206  005061  000000G                 CLR     DEVSIZ(R1)      ;Clear device size
     117 015212  012761  000006G 000000G          MOV     #CLHEAD+6,HANENT(R1) ;Set handler entry point (4th word)
     118 015220  012761  000004  000000G          MOV     #4.,HANSIZ(R1)  ;Set size of handler
     119                                 ;
     120                                 ;  Set the version number to be returned by the .SPFUN used by
     121                                 ;  VTCOM to see if it is matched to the correct version of XL/XC.
     122                                 ;
     123 015226  105737  000000G         13$:    TSTB    CLVERS          ;Was a version specified in TSGEN?
     124 015232  001023                         BNE     9$              ;Br if yes
     125 015234  113703  000000G                 MOVB    SYSVER,R3       ;Get current RT-11 version number
     126 015240  000303                         SWAB    R3              ;Put in high order byte
     127 015242  153703  000000G                 BISB    SYSUPD,R3       ;Put update number in low-order byte
     128 015246  012702  000522'                 MOV     #CLVTBL,R2      ;Point to table with CL version numbers
     129 015252  020322                 10$:    CMP     R3,(R2)+        ;Is this entry for our version?
     130 015254  001407                         BEQ     12$             ;Br if yes
     131 015256  005722                         TST     (R2)+           ;Skip cell with CL version
     132 015260  020227  000556'                 CMP     R2,#CLVEND      ;Checked all table entries?
     133 015264  103772                         BLO     10$             ;Loop if not
     134 015266  012701  000021                 MOV     #17.,R1         ;Set default CL version if not found
     135 015272  000401                         BR      11$
     136 015274  011201                 12$:    MOV     (R2),R1         ;Get correct CL version
     137 015276  110137  000000G         11$:    MOVB    R1,CLVERS       ;Set CL version number
     138                                 ;
     139                                 ;  Finished
     140                                 ;
     141 015302  012603                 9$:     MOV     (SP)+,R3
     142 015304  012602                         MOV     (SP)+,R2
     143 015306  012601                         MOV     (SP)+,R1
     144 015310  000207                         RETURN
```

```
                                        .SBTTL  LDINIT -- Determine LD translation table format
                                ;-----------------------------------------------------------------------
                                ;  RT-11 V5.4 changed that format of the LD translation tables.
                                ;  Determine which version of RT-11 is being used and set the
                                ;  appropriate value for LDVERS to generate correct table
                                ;  format in response to LD .SPFUN 372
                                ;
015312  010146                  LDINIT: MOV     R1,-(SP)
015314  113701  000000G                 MOVB    LDVERS,R1       ;See if value was specified in TSGEN
015320  001022                          BNE     9$              ;If value specified in TSGEN, use it
015322  010246                          MOV     R2,-(SP)
015324  010346                          MOV     R3,-(SP)
015326  013703  000000G                 MOV     SYSVER,R3       ;Get current RT version and update
015332  012701  000001                  MOV     #1,R1           ;Assume version format RT5.3 or earlier
015336  012702  000556'                 MOV     #LD1TBL,R2      ;Point to table of versions using format 1
015342  022203                  1$:     CMP     (R2)+,R3        ;Does this entry match actual version?
015344  001404                          BEQ     5$              ;Exit if so
015346  005712                          TST     @R2             ;End of table?
015350  001374                          BNE     1$              ;Keep looking if more entries
015352  012701  000002                  MOV     #2,R1           ;If not in format 1 table, use format 2
015356  110137  000000G         5$:     MOVB    R1,LDVERS       ;Remember for LD SPFUN 372
015362  012603                          MOV     (SP)+,R3
015364  012602                          MOV     (SP)+,R2
015366  012601                  9$:     MOV     (SP)+,R1
015370  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
                                ;
015372  010246                  INDINI: MOV     R2,-(SP)
015374  010346                          MOV     R3,-(SP)
                                ;
                                ;  Determine if IND support is wanted
                                ;
015376  005037  000000G                 CLR     INDSAV          ;ASSUME IND SUPPORT NOT WANTED
                                ;
                                ;  Lookup SY:IND.SAV file
                                ;
015402  013737  000000G 000264'         MOV     SYNAME,INDNAM   ;LOOK UP IND ON BOOT DEVICE
015410                                  .LOOKUP #AREA,#1,#INDNAM ;TRY TO FIND SY:IND.SAV
015430  103002                          BCC     4$              ;BR IF FOUND IND
015432  000137  016064'                 JMP     9$              ;IF CAN'T FIND IND, THEN NO IND SUPPORT
                                ;
                                ;  Set up information about IND overlay data segment
                                ;
015436  013703  000152'         4$:     MOV     WRKBUF,R3       ;Get pointer to work buffer
015442                                  .READW  #AREA,#1,R3,#256.,#0 ;READ IN BLOCK 0 OF SAV FILE
015476  016302  000064                  MOV     64(R3),R2       ;GET POINTER TO OVERLAY TABLE
015502                                  .READW  #AREA,#1,R3,#256.,#1 ;READ IN BLOCK 1 WITH OVERLAY TABLE
015540  162702  001000                  SUB     #1000,R2        ;GET ADDRESS OF OVERLAY TABLE REL TO BLOCK 1
015544  060302                          ADD     R3,R2           ;ADD BASE ADDRESS WHERE BLOCK 1 DATA IS
015546  011203                          MOV     (R2),R3         ;Get virtual address of segment 0
015550  020312                  5$:     CMP     R3,(R2)         ;Search for 1st segment with different addr
015552  001003                          BNE     6$              ;Br if found it (this is the data segment)
015554  062702  000006                  ADD     #6,R2           ;Point to overlay table entry for next seg
015560  000773                          BR      5$
015562  005722                  6$:     TST     (R2)+           ;Point to word with block # if SAV file
015564  012237  000000G                 MOV     (R2)+,INDDBL    ;GET BLOCK # IN SAV FILE OF DATA OVERLAY
015570  011202                          MOV     (R2),R2         ;GET # OF WORDS IN OVERLAY SEGMENT
015572  062702  000377                  ADD     #255.,R2        ;ROUND UP TO NEXT BLOCK
015576  000302                          SWAB    R2              ;CONVERT # WORDS TO # BLOCKS
015600  042702  177400                  BIC     #^C<377>,R2
015604  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.
                                ;
015610  012700  000000G                 MOV     #INDSAV,R0      ;GET ADDRESS OF SAVESTATUS BLOCK
015614  013702  000264'                 MOV     INDNAM,R2       ;GET RAD50 DEVICE NAME
015620  004737  027126'                 CALL    SETCHN          ;SAVE FILE STATUS
                                ;
                                ;  Determine how much space is needed for SY:TSXIND.TSX swap file
                                ;
015624  013703  000000G                 MOV     INDDBS,R3       ;GET # BLOCKS NEEDED PER JOB
015630  070327  000000C                 MUL     #<LSTSL/2>,R3   ;TIMES TOTAL NUMBER OF JOBS
```

```
                                ;
                                ;  Load Rt-11 device handler for ind swap file.
                                ;
015634  013700  000000G                MOV     INDFIL,R0       ;Get name of the device
015640  004737  027506'                CALL    RTFTCH          ;Try to fetch the RT-11 device handler
015644  103522                         BCS     11$             ;Br if error on handler fetch
                                ;
                                ;  See if TSXIND file already exists
                                ;
015646                                  .LOOKUP #AREA,#1,#INDFIL ;DOES SY:TSXIND.TSX FILE EXIST NOW?
015666  103415                         BCS     1$              ;BR IF NOT
015670  020003                         CMP     R0,R3           ;IS IT OF THE CORRECT SIZE?
015672  001462                         BEQ     2$              ;BR IF YES
015674                                  .PURGE  #1              ;FILE IS OF WRONG SIZE
015702                                  .DELETE #AREA,#1,#INDFIL;DELETE OLD FILE
                                ;
                                ;  File does not now exist
                                ;  Create new file
                                ;
015722                          1$:     .ENTER  #AREA,#1,#INDFIL,R3 ;CREATE NEW TSXIND FILE
015746  103451                         BCS     10$             ;BR IF ERROR ON CREATE
015750  010302                         MOV     R3,R2           ;# BLOCKS IN FILE
015752  005302                         DEC     R2              ;GET # OF LAST BLOCK IN FILE
015754                                  .WRITW  #AREA,#1,WRKBUF,#256.,R2 ;WRITE TO LAST BLOCK OF FILE
016012                                  .CLOSE  #1              ;NOW CLOSE THE FILE
016020                                  .LOOKUP #AREA,#1,#INDFIL ;REOPEN TSXIND FILE WITH LOOKUP
                                ;
                                ;  Do .SAVESTATUS for SY:TSXIND.TSX file
                                ;
016040  012700  000000G         2$:     MOV     #INDTSV,R0      ;POINT TO SAVESTATUS BLOCK
016044  013702  000000G                 MOV     INDFIL,R2       ;GET RAD50 DEVICE NAME
016050  004737  027126'                 CALL    SETCHN          ;SAVE FILE INFO
016054                                  .RELEAS #INDFIL         ;Release device handler
                                ;
                                ;  Finished
                                ;
016064  012603                  9$:     MOV     (SP)+,R3
016066  012602                          MOV     (SP)+,R2
016070  000207                          RETURN
                                ;
                                ;  Error occurred while opening SY:TSXIND.TSX file
                                ;
016072                          10$:    .PRINT  #TSXHD          ;Print error message
016100                                  .PRINT  #INDOPN
016106  004737  011276'                 CALL    SPNEED          ;Print info about number of blocks needed
                                ;
                                ;  Error: Invalid device specification.
                                ;
016112  010001                  11$:    MOV     R0,R1           ;Save device name
016114                                  .PRINT  #TSXHD          ;Print error message
016122                                  .PRINT  #INDOPN
016130  004737  011324'                 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.
                                ;
016134  010246                 UCLINI: MOV     R2,-(SP)
016136  010346                         MOV     R3,-(SP)
                                ;
                                ;  Determine if TSXUCL data file is needed
                                ;
016140  105737  000000G                 TSTB    VU$CL           ;Is TSXUCL being used at all?
016144  001536                         BEQ     9$              ;Br if not
016146  013702  000000G                 MOV     VUCLMC,R2       ;Get maximum number of commands
016152  001533                         BEQ     9$              ;Br if none allowed
                                ;
                                ;  Determine number of blocks needed in data file for each job
                                ;
016154  012700  000000G                 MOV     #UK$$SZ,R0      ;Size of each keyword descriptor
016160  062700  000000G                 ADD     #US$$SZ,R0      ;Size of each command string descriptor
016164  070200                         MUL     R0,R2           ;Compute total # bytes for keywords+commands
016166  062703  000777G                 ADD     #UC$$SZ+511.,R3 ;Add space for control information & round up
016172  005502                         ADC     R2              ;Propogate carry
016174  071227  001000                 DIV     #512.,R2        ;Convert to # of blocks needed
016200  010237  000000G                 MOV     R2,UCLBLK       ;Save number of blocks needed per job
                                ;
                                ;  Multiply by number of jobs to get total file size
                                ;
016204  070227  000000C                 MUL     #<LSTSL/2>,R2   ;Times total number of jobs
                                ;
                                ;  Load Rt-11 device handler for ind swap file.
                                ;
016210  013700  000000G                 MOV     UCLDAT,R0       ;Get name of the device
016214  004737  027506'                 CALL    RTFTCH          ;Try to fetch the RT-11 device handler
016220  103523                         BCS     11$             ;Br if error on handler fetch
                                ;
                                ;  The total required file size is now in R3.
                                ;  See if the file already exists.
                                ;
016222                                  .LOOKUP #AREA,#1,#UCLDAT ;See if the file exists now
016242  103415                         BCS     1$              ;Br if file does not exist
016244  020003                         CMP     R0,R3           ;Is existing file of correct size?
016246  001446                         BEQ     2$              ;Br if yes -- use the old file
016250                                  .PURGE  #1              ;Purge the channel
016256                                  .DELETE #AREA,#1,#UCLDAT;Delete the old file
                                ;
                                ;  Create a new data file
                                ;
016276                          1$:     .ENTER  #AREA,#1,#UCLDAT,R3 ;Create new data file
016322  103452                         BCS     10$             ;Br if error creating the file
016324  005303                         DEC     R3              ;Get # of last block in the file
016326                                  .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.)
                                        ;
016364                                  2$:     .CSTAT  #AREA,#1,#NFSBLK        ;GET CHANNEL STATUS INFORMATION
016404  013702  000032'                        MOV     <NFSBLK+12>,R2  ;FETCH DEVICE NAME IN RAD50
016410  063702  000030'                        ADD     <NFSBLK+10>,R2  ;ADD IN DEVICE UNIT NUMBER
016414  062702  000036                        ADD     #^R  0,R2       ;CONVERT UNIT NUMBER TO RAD50
016420  010237  000000G                       MOV     R2,UCLDAT       ;SET PHYSICAL NAME BACK INTO TSGEN CELL
016424                                          .CLOSE  #1              ;Close the file
016432                                          .RELEAS #UCLDAT         ;Release device handler
                                        ;
                                        ;  Finished
                                        ;
016442  012603                          9$:     MOV     (SP)+,R3
016444  012602                                  MOV     (SP)+,R2
016446  000207                                  RETURN
                                        ;
                                        ;  Error creating the data file
                                        ;
016450                                  10$:    .PRINT  #TSXHD          ;Print error message
016456                                          .PRINT  #UCLOPN
016464  004737  011276'                        CALL    SPNEED          ;Print info about number of blocks needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
016470  010001                          11$:    MOV     R0,R1           ;Save device name
016472                                          .PRINT  #TSXHD          ;Print error message
016500                                          .PRINT  #UCLOPN
016506  004737  011324'                        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.
                                ;
016512  010146                  MEMINI: MOV     R1,-(SP)
016514  010246                          MOV     R2,-(SP)
016516  010346                          MOV     R3,-(SP)
016520  010446                          MOV     R4,-(SP)
016522  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize all pages for a 1-to-1 mapping.
                                ;
016524  012700  000000G         12$:    MOV     #KPAR0,R0       ;Kernel mode PAR 0
016530  012701  000000G                 MOV     #UPAR0,R1       ;User mode PAR 0
016534  012702  000000G                 MOV     #KPDR0,R2       ;Kernel mode PDR 0
016540  012703  000000G                 MOV     #UPDR0,R3       ;User mode PDR 0
016544  012704  000010                  MOV     #8.,R4          ;Initialize 8 pages
016550  005005                          CLR     R5              ;Set initial PAR value
016552  010520                  2$:     MOV     R5,(R0)+        ;Set kernel PAR
016554  010521                          MOV     R5,(R1)+        ;Set user PAR value
016556  012722  077406                  MOV     #077406,(R2)+   ;Set kernel PDR
016562  012723  077406                  MOV     #077406,(R3)+   ;Set user PDR value
016566  062705  000200                  ADD     #200,R5         ;Advance block number
016572  077411                          SOB     R4,2$           ;Init all pages
                                ;
                                ;  Map kernel mode I/O page (160000) to 17760000.
                                ;
016574  012737  000000G 000000G         MOV     #IOPAGE,@#KPAR7 ;Map I/O page
                                ;
                                ;  Finished
                                ;
016602  012605                          MOV     (SP)+,R5
016604  012604                          MOV     (SP)+,R4
016606  012603                          MOV     (SP)+,R3
016610  012602                          MOV     (SP)+,R2
016612  012601                          MOV     (SP)+,R1
016614  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

016616  010146                  MEMTST: MOV     R1,-(SP)
016620  010246                          MOV     R2,-(SP)
016622  010346                          MOV     R3,-(SP)
016624  010446                          MOV     R4,-(SP)
016626  010546                          MOV     R5,-(SP)
016630  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.
                                ;
016634  012737  017170' 000004          MOV     #TRCSET,@#4     ;Catch trap
016642  000240                          NOP                     ;Clean out 11/73 pipeline
016644  000240                          NOP                     ;Before attempting trap
016646  005737  000000G               TST     @#SR3MMR        ;Try to access status register 3
016652  103402                          BCS     22$             ;Br if MMU 3 status register is non-existent
016654  105237  000000G               INCB    SR3FLG          ;No trap.  We must have SR3
                                ;
                                ;  If we are running on a Professional, there is a register that tells
                                ;  us how much memory is installed on the machine.
                                ;
016660                          22$:
                                .IF     NE,PROASM
                                        TSTB    PROFLG          ;Are we running on a Professional?
                                        BEQ     26$             ;Br if not
                                        CLR     R5              ;Load byte without sign extension
                                        BISB    @#173050,R5     ;Get 32Kb top of system RAM boundary
                                        ASH     #9.,R5          ;Convert to # 64 byte blocks
                                        SUB     #10,R5          ;Don't use the last 512 bytes of memory
                                        BR      7$
                                26$:
                                .ENDC   ;NE,PROASM
                                .IF     NE,<PROASM-1>           ;Assemble if could be on a PDP-11
                                ;
                                ;  We are not running on a Professional.
```

```
                                        ;  Test each page above TSX to see where the top of memory is.
                                        ;
016660  012737  017200' 000020          MOV     #RTNKM,@#20     ;Use IOT instruction to get out of user mode
016666  005037  000022                  CLR     @#22
016672  052737  000000G 000000G        BIS     #MMENBL,@#SROMMR;Enable memory management
016700  105737  000000G                TSTB    SR3FLG          ;Does maching have mem management reg # 3?
016704  001403                          BEQ     4$              ;Br if non-existent
016706  052737  000000G 000000G        BIS     #EMMAP,@#SR3MMR ;Enable 22-bit extended memory
                                        ;
                                        ;  Map user page 7 to each successive 256-word block and attempt to access.
                                        ;
016714  012705  002000          4$:     MOV     #1024.,R5       ;Start checking at 64Kb
016720  010537  000000G         5$:     MOV     R5,@#UPAR7      ;Map user page 7 to page to be tested
016724  052737  000000G 000000G        BIS     #UMODE,@#PSW    ;Go into user mode
016732  005737  160000                TST     @#160000        ;Can we access the page?
                                        ;
                                        ;  Use IOT to get back into kernel mode.
                                        ;
016736  000004                          IOT                     ;Return to kernel mode
016740  103405                          BCS     6$              ;Br if memory is non-existent
016742  062705  000010                  ADD     #10,R5          ;Go try next page
016746  020527  177600                  CMP     R5,#177600      ;Don't enter I/O page
016752  103762                          BLO     5$
                                        ;
                                        ;  Check for potential memory wrap (on 18-bit 256K byte computers).
                                        ;
016754  020527  010000          6$:     CMP     R5,#10000       ;Is physical memory above 256K bytes
016760  101421                          BLOS    7$              ;Br if below 256K bytes
016762  005037  000110                  CLR     @#TSTWRD        ;Clear physical location
016766  012737  010000  000000G        MOV     #10000,@#UPAR7  ;Map to 256K byte boundary
016774  052737  000000G 000000G        BIS     #UMODE,@#PSW    ;Go into user mode
017002  012737  177777  160110          MOV     #-1,@#160000+TSTWRD     ;Store -1 at 256K physical location
017010  000004                          IOT                     ;Return to kernel mode
017012  005737  000110                  TST     @#TSTWRD        ;Test physical location
017016  001402                          BEQ     7$              ;Br if physical location is clear
017020  012705  010000                  MOV     #10000,R5       ;Constrain memory to 256K byte total
                                .ENDC   ;NE,<PROASM-1>
                                        ;
                                        ;  Reached end of available memory.
                                        ;
017024  010537  000000G         7$:     MOV     R5,PHYMEM       ;set physical memory size
017030  020537  000000G                 CMP     R5,MAPSIZ       ;Constrain kernel to user specified cutoff
017034  101402                          BLOS    8$              ;Br if below user specified
017036  013705  000000G                 MOV     MAPSIZ,R5       ;Only use this much memory
017042  010537  000000G         8$:     MOV     R5,$MEMSZ       ;Set # 64-byte blocks of total memory
017046  010537  000134'                 MOV     R5,FMEMHI       ;Save base 64-byte block # of top of free mem
                                        ;
                                        ;  Turn off memory management
                                        ;
017052  105737  000000G                TSTB    SR3FLG          ;Do we have memory management reg # 3?
017056  001403                          BEQ     9$              ;Br if non-existent
017060  042737  000000G 000000G        BIC     #EMMAP,@#SR3MMR ;Disable extended memory management
017066  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
                                        ;
017074  023727  000134' 010000          CMP     FMEMHI,#4096.   ;Does machine have at least 256Kb?
```

```
    115 017102  103411                          BLO     25$             ;Br if not
    116 017104  105237  000000G                  INCB    MEM256          ;Remember machine has at least 256kb
    117 017110  123727  000000G 000000G         CMPB    VBUSTP,#QBUS    ;Is this a Q-bus machine?
    118 017116  001003                          BNE     25$             ;Br if not
    119 017120  052737  000001  000306'         BIS     #EXTLSI,ICONFG  ;Set extended-LSI flag in ICONFG
    120                                 ;
    121                                 ;  See if this machine needs UNIBUS mapping
    122                                 ;
    123 017126  123727  000000G 000000G 25$:    CMPB    VBUSTP,#UNIBUS  ;Is this a UNIBUS machine?
    124 017134  001005                          BNE     29$             ;Br if not
    125 017136  105737  000000G                  TSTB    MEM256          ;Does machine have at least 256kb of memory?
    126 017142  001402                          BEQ     29$             ;Br if not
    127 017144  105237  000000G                  INCB    UBUSMP          ;Say UNIBUS mapping is needed
    128                                 ;
    129                                 ;  Finished
    130                                 ;
    131 017150  012637  000004          29$:    MOV     (SP)+,@#4       ;Reset trap vector
    132 017154  012605                          MOV     (SP)+,R5
    133 017156  012604                          MOV     (SP)+,R4
    134 017160  012603                          MOV     (SP)+,R3
    135 017162  012602                          MOV     (SP)+,R2
    136 017164  012601                          MOV     (SP)+,R1
    137 017166  000207                          RETURN
    138                                 ;
    139                                 ;  Trap - return with C-bit set.
    140                                 ;
    141 017170  052766  000001  000002  TRCSET: BIS     #1,2(SP)        ;Set c-bit for return
    142 017176  000002                          RTI                     ;Return from trap
    143                                 ;
    144                                 ;  IOT - return at kernel mode with c-bit preserved.
    145                                 ;
    146 017200  042766  000000G 000002  RTNKM:  BIC     #UMODE,2(SP)    ;Clear user mode - return to kernel
    147 017206  000002                          RTI                     ;Return from trap
    148                                 ;
    149                                 ;  Error: System does not have memory management hardware.
    150                                 ;
    151 017210                          NOXM:   .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
    152 017216                                  .PRINT  #NXMMSG
    153 017224  000137  004216'                 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.
                                ;
017230                          CXTALC:
                                ;
                                ;  Get size of base portion of job context area
                                ;
017230  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.
                                ;
017234  062700  000077                  ADD     #63.,R0         ;Bound up to 64 byte boundary
017240  042700  000077                  BIC     #77,R0
017244  010037  000000G                 MOV     R0,CXTRMN       ;Offset to start of simulated RMON
017250  062737  000000G 000000G         ADD     #CXTBAS,CXTRMN  ;Add base virtual address of context area
017256  062700  000001G                 ADD     #MVSIZ+1,R0     ;Add space for simulated RMON & channels
                                ;
                                ;  Save number of words needed for context area
                                ;
017262  006200                          ASR     R0              ;Convert to # words
017264  010037  000000G                 MOV     R0,CXTWDS       ;This is # words for whole job context area
                                ;
                                ;  Compute PDR value to use to map to job context area
                                ;
017270  062700  000037                  ADD     #31.,R0         ;Bound up to # 32 word units
017274  072027  177773                  ASH     #-5.,R0         ;Get # 32-word units for context area
017300  000300                          SWAB    R0              ;Put # 32-word units in high-order byte
017302  052700  000006                  BIS     #6,R0           ;Set PDR control flags
017306  010037  000000G                 MOV     R0,CXTPDR       ;This is the PDR value
                                ;
                                ;  Compute # 512-byte pages needed for job context block
                                ;
017312  013700  000000G                 MOV     CXTWDS,R0       ;Get back # words for context area
017316  062700  000377                  ADD     #255.,R0        ;Bound up to # 256-word blocks
017322  072027  177770                  ASH     #-8.,R0         ;Get # 256-word pages for context area
017326  010037  000000G                 MOV     R0,CXTPAG       ;# pages for job context area
                                ;
                                ;  Set up PDR value used when mapping to simulated RMON
                                ;
017332  012700  000000G                 MOV     #MVSIZ,R0       ;Get size of monitor vector table
017336  062700  000077                  ADD     #63.,R0         ;Round up to # 32 word blocks
017342  072027  177772                  ASH     #-6,R0          ;Cvt to # 32-word blocks
017346  005300                          DEC     R0              ;Get # blocks - 1
017350  000300                          SWAB    R0              ;Put # blocks in left byte
017352  052700  000006                  BIS     #6,R0           ;Allow read and write access
017356  042700  100261                  BIC     #100261,R0      ;Make sure unused PDR bits are zero
017362  010037  000000G                 MOV     R0,RMNPDR       ;This is PDR value to map to sim. RMON
                                ;
```

```
                                        ;  Finished
                                        ;
017366  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.
                                ;
017370  010246                  MAPALC: MOV     R2,-(SP)
017372  010346                          MOV     R3,-(SP)
017374  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.
                                ;
017376  010503                          MOV     R5,R3           ;Get 64-byte block # of top of free mem
017400  072327  177775                  ASH     #-3,R3          ;Convert to 512-byte page #
017404  042703  160000                  BIC     #160000,R3      ;Kill possible sign extension
017410  013702  000136'                 MOV     FMEMLO,R2       ;Get 64-byte block # of base of free memory
017414  062702  000007                  ADD     #7,R2           ;Round up
017420  072227  177775                  ASH     #-3,R2          ;Convert to 512-byte page #
017424  042702  160000                  BIC     #160000,R2      ;Kill possible sign extension
017430  160203                          SUB     R2,R3           ;Get # bytes needed for allocation table
017432  003440                          BLE     10$             ;Br if memory overflow
017434  010304                          MOV     R3,R4           ;Get # bytes for allocation table
017436  062704  001000                  ADD     #512.,R4        ;Add 1 extra byte and round up to 512-byte
017442  072427  177767                  ASH     #-9.,R4         ;Get # 512-byte units needed for alloc table
                                ;
                                ;  Set up virtual address pointers for the allocation table
                                ;
017446  012700  000000G                 MOV     #VPAR5,R0       ;We will map to alloc table through PAR 5
017452  010037  000000G                 MOV     R0,LOMAP        ;Pointer to 1st entry in alloc table
017456  160200                          SUB     R2,R0           ;Get pseudo virtual address for page # 0
017460  010037  000000G                 MOV     R0,BASMAP       ;This would point to alloc entry for page 0
017464  012700  000000G                 MOV     #VPAR5,R0       ;Get back base address of table
017470  060300                          ADD     R3,R0           ;Add # bytes used by table
017472  160400                          SUB     R4,R0           ;Subtract space used by table itself
017474  010037  000000G                 MOV     R0,HIMAP        ;Virtual address of 1st entry for system page
                                ;
```

```
                                        ;  Allocate space for the allocation table
                                        ;
017500  072427  000003                          ASH     #3,R4           ;Get # 64-byte units for alloc table
017504  160405                          SUB     R4,R5           ;Compute physical 64-byte base for table
017506  020537  000136'                        CMP     R5,FMEMLO       ;Did we run out of memory space?
017512  101410                          BLOS    10$             ;Br if memory overflow
017514  010537  000000G                        MOV     R5,MAPPAR       ;Use this value to map PAR 5 to alloc table
017520  010537  000134'                        MOV     R5,FMEMHI       ;Save new top of free memory area
                                        ;
                                        ;  Finished
                                        ;
017524  012604                          MOV     (SP)+,R4
017526  012603                          MOV     (SP)+,R3
017530  012602                          MOV     (SP)+,R2
017532  000207                          RETURN
                                        ;
                                        ;  Error: Generated system is too large
                                        ;
017534                                  10$:    .PRINT  #TSXHD          ;Print error message heading
017542                                          .PRINT  #PHSOVF         ;Physical memory overflow
017550  000137  004216'                        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)
                                ;
017554  010546                  SETJSZ: MOV     R5,-(SP)
                                ;
                                ;  Determine total number of pages of memory available to user jobs
                                ;
017556  013705  000000G                 MOV     HIMAP,R5        ;POINTER ABOVE LAST FREE PAGE ENTRY
017562  163705  000000G                 SUB     LOMAP,R5        ;GET TOTAL # OF FREE PAGES
017566  010537  000000G                 MOV     R5,FREPGS       ;# FREE PAGES AVAILABLE TO USER JOBS
                                ;
                                ;  Make sure there is enough free space to run TSKMON.
                                ;
017572  020537  000000G                 CMP     R5,KMNPGS       ;COMPARE # FREE PAGES TO # PAGES FOR TSKMON
017576  103436                          BLO     1$              ;BR IF INSUFFICIENT MEMORY TO RUN TSKMON
                                ;
                                ;  Set up max memory limit for jobs
                                ;
017600  013700  000000G                 MOV     CXTPAG,R0       ;# PAGES NEEDED FOR JOB CONTEXT AREA
017604  010037  000000G                 MOV     R0,JCXPGS
017610  160005                          SUB     R0,R5           ;LEAVE ROOM FOR JOB CONTEXT AREA
017612  006205                          ASR     R5              ;Convert # pages to # KB
017614  020537  000000G                 CMP     R5,VHIMEM       ;COMPARE WITH TSGEN SPECIFIED MAX SIZE
017620  101402                          BLOS    10$             ;BR IF CONSTRAINED BY PHYSICAL SIZE
017622  013705  000000G                 MOV     VHIMEM,R5       ;LIMIT BY VALUE SPECIFIED IN TSGEN
017626  010537  000000G          10$:   MOV     R5,MXJMEM       ;MAX # K BYTES AVAILABLE TO A JOB
017632  010500                          MOV     R5,R0
017634  072027  000012                  ASH     #10.,R0         ;CONVERT # KB TO BYTE ADDRESS
017640  001002                          BNE     2$              ;BR IF DIDN'T OVERFLOW 64KB
017642  012700  177774                  MOV     #177774,R0      ;GET 64KB TOP ADDRESS
017646  010037  000000G          2$:    MOV     R0,MXJADR       ;ADDRESS ABOVE TOP OF JOB
                                ;
                                ;  Set default memory size of jobs
                                ;
017652  020537  000000G                 CMP     R5,VDFMEM       ;COMPARE TO DEFAULT SPECIFIED IN TSGEN
017656  101402                          BLOS    11$             ;BR IF CONSTRAINED BY PHYSICAL LIMIT
017660  013705  000000G                 MOV     VDFMEM,R5       ;CONSTRAIN BY TSGEN PARAMETER
017664  010537  000000G          11$:   MOV     R5,DFJMEM       ;SET DEFAULT JOB MEMORY SIZE
                                ;
                                ;  Finished
                                ;
017670  012605                          MOV     (SP)+,R5
017672  000207                          RETURN
                                ;
                                ;  Error -- Insufficient memory space available to run TSKMON.
```

```
     58                                         ;
     59 017674  004737  027642'          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.
                                ;
017700  005727  000000G         PARSET: TST     #MPARFL         ;Does he want to disable memory parity?
017704  001027                        BNE     20$             ;Br if not
017706  010246                        MOV     R2,-(SP)
017710  013746  000004                MOV     @#4,-(SP)       ;Save contents of trap vector
                                ;
                                ;  Catch traps that occur when we access unimplemented parity registers
                                ;
017714  012737  017740' 000004        MOV     #2$,@#4         ;Send traps to 2$
017722  000240                        NOP                     ;Clean out instruction pipeline
017724  000240                        NOP
                                ;
                                ;  Disable parity for each block of memory
                                ;
017726  012702  000000G               MOV     #MPAR0,R2       ;Point to 1st memory control register
017732  042712  000000G         1$:   BIC     #PARENL,(R2)    ;Disable memory parity
017736  000402                        BR      3$              ;We did not trap
017740  062706  000004          2$:   ADD     #4,SP           ;Clean trap PS and PC off of stack
017744  062702  000002          3$:   ADD     #2,R2           ;Point to next parity control register
017750  020227  000000G               CMP     R2,#MPAR16      ;Have we cleared all registers?
017754  101766                        BLOS    1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
017756  012637  000004                MOV     (SP)+,@#4       ;Restore trap vector
017762  012602                        MOV     (SP)+,R2
017764  000207                20$:    RETURN
                                        .IFF    ;NE,<PROASM-1>
                                PARSET: RETURN
                                        .ENDC   ;NE,<PROASM-1>
```

```
                                        .SBTTL  GETHNL -- Load device handlers into memory
                                ;-----------------------------------------------------------------------
                                ;  GETHNL performs two functions:
                                ;   1. Set up information in the device tables about all devices.
                                ;   2. Load those handlers that reside in low memory.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of start of free memory.
                                ;
                                ;  Outputs:
                                ;   R5 = Address of new start of free memory.
                                ;   NMXHAN = Number of handlers to load into extended memory.
                                ;
017766  010146                  GETHNL: MOV     R1,-(SP)
017770  010246                          MOV     R2,-(SP)
017772  010446                          MOV     R4,-(SP)
                                ;
                                ;  Begin loop to check all handlers specified in TSGEN with DEVDEF.
                                ;
017774  005001                          CLR     R1              ;Init device table index
017776  020127  000000C         1$:     CMP     R1,#<AHEND-AUTHAN> ;Done all devices?
020002  103015                          BHIS    2$              ;Br if yes
020004  016102  000000G                 MOV     AUTHAN(R1),R2   ;Get the name of the device
020010  001407                          BEQ     3$              ;Ignore null devices
020012  020227  000000G                 CMP     R2,#DMYDEV      ;Is this a dummy device entry?
020016  001404                          BEQ     3$              ;Skip it if yes
020020  016104  000000G                 MOV     DTYPE(R1),R4    ;Get flags specified in TSGEN
                                ;
                                ;  Load this handler
                                ;
020024  004737  020076'                 CALL    LDHAND          ;Try to load handler into memory
                                ;
                                ;  Check next device
                                ;
020030  062701  000002          3$:     ADD     #2,R1           ;Advance device index
020034  000760                          BR      1$              ;See if more devices to load
                                ;
                                ;  Now see if there are spooled devices to contend with
                                ;
020036  012704  000000G         2$:     MOV     #SPLND,R4       ;Are there any spooled devices?
020042  001411                          BEQ     9$              ;Br if not
020044  012701  000000G                 MOV     #SPLDEV,R1      ;Point to spooled device name table
020050  012102                  5$:     MOV     (R1)+,R2        ;Get the name of the next spooled device
020052  010446                          MOV     R4,-(SP)        ;Save device count
020054  005004                          CLR     R4              ;Say no TSGEN flags for device
020056  004737  020076'                 CALL    LDHAND          ;Load the handler
020062  012604                          MOV     (SP)+,R4        ;Recover the device count
020064  077407                          SOB     R4,5$           ;Loop if more handlers to load
                                ;
                                ;  Finished
                                ;
020066  012604                  9$:     MOV     (SP)+,R4
020070  012602                          MOV     (SP)+,R2
020072  012601                          MOV     (SP)+,R1
020074  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$xxx status flags for device from TSGEN.
                                ;   R5 = Address where handler is to be loaded.
                                ;
                                ;  Outputs:
                                ;   R5 = New free memory address.
                                ;   NUMDEV = Incremented by 2.
                                ;   PNAME(i) = Rad-50 name of device.
                                ;   ENTRY(i) = Handler entry point.
                                ;   DVSTAT(i) = Device status flags.
                                ;   DVFLAG(i) = TSX-Plus device status flags.
                                ;   HANPAR(i) = PAR offset if this is a mapped handler.
                                ;
020076  010446                 LDHAND: MOV     R4,-(SP)
                                ;
                                ;  Determine if we should ignore this device
                                ;
020100  004737  020246'                 CALL    INSCK1          ;Should we ignore this device?
020104  103456                         BCS     9$              ;Br if yes
                                ;
                                ;  The initial tests indicate that this handler should be loaded.
                                ;  Now open the handler file and perform some additional checks.
                                ;
020106  004737  020424'                 CALL    INSCK2          ;Perform some additional checks on handler
020112  103450                         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.
                                ;
020114  004737  021036'                 CALL    STDVTB          ;Set up info in tables for this device
                                ;
                                ;  Determine if this handler is to be loaded into low memory or
                                ;  extended memory.
                                ;
020120  016400  000000G                 MOV     DVFLAG(R4),R0   ;Get TSX-Plus status flags for device
020124  032700  000000G                 BIT     #DX$NHM,R0      ;Are we never to map this handler?
020130  001035                         BNE     1$              ;Br if handler cannot be mapped
020132  032700  000000G                 BIT     #DX$MPH,R0      ;Is mapping wanted for this handler?
020136  001432                         BEQ     1$              ;Br if not
020140  032700  000000G                 BIT     #DX$IBH,R0      ;Does this handler have an internal I/O buff?
020144  001412                         BEQ     2$              ;Br if not
020146  105737  000000G                 TSTB    UBUSMP          ;Does this machine have a mapped UNIBUS?
020152  001024                         BNE     1$              ;Br if yes -- Don't map this handler
020154  032700  000000G                 BIT     #DX$MAP,R0      ;Does this handler require I/O mapping?
020160  001404                         BEQ     2$              ;Br if not
020162  032737  000001  000306'        BIT     #EXTLSI,ICONFG  ;Is this a Q-bus system with more than 256Kb?
020170  001015                         BNE     1$              ;Br if yes -- Don't map this handler
                                ;
                                ;  This handler can be mapped and will be loaded in extended memory
```

```
      58                                        ;
      59 020172  005237  000124'          2$:      INC     NMXHAN          ;Count # of mapped handlers
      60 020176  012764  000001  000000G          MOV     #1,HANPAR(R4)   ;Set flag saying handler should be mapped
      61                                        ;
      62                                        ;  Make sure size of mapped handler does not exceed 8KB
      63                                        ;
      64 020204  026427  000000G 020000          CMP     HANSIZ(R4),#8192.  ;Is mapped handler too big?
      65 020212  101410                          BLOS    8$              ;Br if not too big
      66 020214  012700  002452'                 MOV     #HN2BIG,R0      ;Get error message address
      67 020220  000137  022124'                 JMP     HLERR           ;Abort initialization
      68                                        ;
      69                                        ;  This handler must be loaded into low memory
      70                                        ;
      71 020224  005064  000000G          1$:      CLR     HANPAR(R4)      ;Say this handler is not mapped
      72 020230  004737  021146'                 CALL    LDHNLO          ;Load handler into low memory
      73                                        ;
      74                                        ;  Close the handler file
      75                                        ;
      76 020234                           8$:      .CLOSE  #1              ;Close the handler file
      77                                        ;
      78                                        ;  Finished
      79                                        ;
      80 020242  012604                   9$:      MOV     (SP)+,R4
      81 020244  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.
                                ;
020246  010146                  INSCK1: MOV     R1,-(SP)
                                ;
                                ;  Strip off any specified unit number
                                ;
020250  010201                          MOV     R2,R1           ;Get full device name
020252  005000                          CLR     R0              ;Set for divide
020254  071027  000050                  DIV     #50,R0          ;Split off last digit
020260  070027  000050                  MUL     #50,R0          ;Now correct for divide
020264  010102                          MOV     R1,R2           ;Get device name less 3rd digit
020266  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.
                                ;
020272  020237  000176'                  CMP     R2,R50LD        ;Is device name LD?
020276  001004                          BNE     1$              ;Br if not
020300  105737  000000G                 TSTB    VLDSYS          ;Is standard system LD support included?
020304  001044                          BNE     5$              ;Br if yes -- Don't load LD
020306  000417                          BR      3$              ;Load LD
020310  012701  000224'          1$:     MOV     #SKPDEV,R1      ;Point to table of devices to skip
020314  020221                  2$:     CMP     R2,(R1)+        ;Is this a device to be skipped?
020316  001437                          BEQ     5$              ;Br if yes
020320  005711                          TST     (R1)            ;Reached end of skip list?
020322  001374                          BNE     2$              ;Loop if not
                                ;
                                ;  See if we have already loaded the handler for this device
                                ;
020324  013701  000000G                 MOV     NUMDEV,R1       ;Get index for last device
020330  001406                          BEQ     3$              ;Br if no devices installed yet
020332  020261  000000G          4$:     CMP     R2,PNAME(R1)    ;See if this device is already installed
020336  001427                          BEQ     5$              ;Br if already installed
020340  162701  000002                  SUB     #2,R1           ;More installed devices to check?
020344  003372                          BGT     4$              ;Loop if yes
                                ;
                                ;  This handler is to be loaded.
                                ;  Get default TSX-Plus control flags for this device.
                                ;
020346  012701  000342'          3$:     MOV     #DVFLBS,R1      ;Point to start of table
020352  020261  000000           6$:     CMP     R2,DV$NAM(R1)   ;Search for device in the table
020356  001003                          BNE     7$              ;Br if this is not it
020360  056104  000002                  BIS     DV$FLG(R1),R4   ;Combine default flags
020364  000405                          BR      8$
```

```
     58 020366  062701  000004          7$:     ADD     #DV$$SZ,R1      ;Point to next entry
     59 020372  020127  000522'                 CMP     R1,#DVFLND      ;Checked all entries?
     60 020376  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 020400  032704  000000G         8$:     BIT     #DX$DMA,R4      ;Is this a DMA device?
     66 020404  001402                          BEQ     10$             ;Br if not
     67 020406  052704  000000G                 BIS     #DX$EBA,R4      ;Set even-buffer-boundary flag
     68                                 ;
     69                                 ;  Load this handler
     70                                 ;
     71 020412  000241                  10$:    CLC                     ;Set flag saying to load the handler
     72 020414  000401                          BR      9$
     73                                 ;
     74                                 ;  Do not load this handler
     75                                 ;
     76 020416  000261                  5$:     SEC                     ;Set flag saying not to load the handler
     77                                 ;
     78                                 ;  Finished
     79                                 ;
     80 020420  012601                  9$:     MOV     (SP)+,R1
     81 020422  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.
                                ;
020424  010146                  INSCK2: MOV     R1,-(SP)
020426  010246                          MOV     R2,-(SP)
020430  010346                          MOV     R3,-(SP)
020432  010446                          MOV     R4,-(SP)
020434  010546                          MOV     R5,-(SP)
020436  013746  000004                  MOV     @#4,-(SP)       ;Save the bus timeout vector
020442  013746  000010                  MOV     @#10,-(SP)      ;Save illegal instruction vector
                                ;
                                ;  Try to lookup handler file on system disk
                                ;
020446  010237  000246'                 MOV     R2,HANNAM+2     ;Set the device name for the lookup
020452                                  .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
020472  103011                          BCC     1$              ;Br if we found the handler file
                                ;
                                ;  Error -- Cannot find handler file
                                ;
000000G                         TSTB    VINABT          ;Abort or continue on errors?
000000G                         BNE     2$              ;Br if abort on errors
021004'                         JMP     10$             ;Say not to load this handler
001636'                 2$:     MOV     #CFHMSG,R0      ;Can't find handler
022124'                         JMP     HLERR           ;Abort initialization
                                ;
                                ;  We were able to open the handler file.
                                ;  Read in block 0 of handler.
                                ;
020516                  1$:     .READW  #AREA,#1,WRKBUF,#256.,#0 ;Read block 0 into WRKBUF
020554  103004                  BCC     3$              ;Br if read ok
001700'                         MOV     #ERHMSG,R0      ;Error during read
022124'                         JMP     HLERR
                                ;
                                ;  Determine if the handler is supported under the current RT-11 version
                                ;
000330'                 3$:     MOV     #HVTBL,R0       ;Point to table with handler version info
000152'                         MOV     WRKBUF,R1       ;Point to buffer with block 0 of handler
000000G                         MOVB    H.DSTS(R1),R1   ;Get device ID code from handler
000000          51$:    CMPB    R1,HV$ID(R0)    ;Compare handler ID code with table entry
                                BNE     53$             ;Br if this entry not for this handler
000000G 000001                  CMPB    SYSVER,HV$VER(R0);Compare curr RT-11 vers with min acceptable
                                BLO     52$             ;Br if version is not adequate
                                BHI     54$             ;Br if version is ok
000000G 000002                  CMPB    SYSUPD,HV$UPD(R0);Compare update level
                                BHIS    54$             ;Br if update level is ok
000000G                 52$:    TSTB    VINABT          ;Handler not loadable - abort or continue?
                                BEQ     10$             ;Br if continue but don't allow loading
```

```
     58 020640  012700  001744'                    MOV     #ERHNDV,R0      ;Wrong version of RT for handler
     59 020644  000137  022124'                    JMP     HLERR           ;Report error and abort
     60 020650  062700  000003          53$:      ADD     #HV$$SZ,R0      ;Point to next handler version table entry
     61 020654  020027  000341'                    CMP     R0,#HVEND       ;Are there more entries?
     62 020660  103750                             BLO     51$             ;Loop if more to check
     63                                 ;
     64                                 ;  Check handler sysgen options
     65                                 ;
     66 020662  013700  000152'         54$:      MOV     WRKBUF,R0       ;Point to buffer with handler block 0
     67 020666  032760  000000G 000000G           BIT     #SG$MMU,H.GEN(R0);Was handler genned with XM support?
     68 020674  001004                             BNE     4$              ;Br if yes
     69 020676  012700  002045'                    MOV     #HSGER,R0       ;Error if not XM version of handler
     70 020702  000137  022124'                    JMP     HLERR
     71 020706  016037  000000G 000304' 4$:       MOV     H.GEN(R0),HGENFL;Save handler sysgen flags for later
     72                                 ;
     73                                 ;  Check the CSR address specified in the handler to see if the
     74                                 ;  hardware device for this handler exists.
     75                                 ;
     76 020714  012737  017170' 000004             MOV     #TRCSET,@#4     ;Catch bus timeout traps
     77 020722  012737  017170' 000010             MOV     #TRCSET,@#10    ;Catch illegal instruction traps
     78 020730  013700  000152'                    MOV     WRKBUF,R0       ;Point to start of block 0 of handler
     79 020734  016001  000000G                    MOV     H.CSR(R0),R1    ;Get address of CSR for device
     80 020740  001402                             BEQ     5$              ;Br if no CSR specified
     81 020742  005711                             TST     (R1)            ;Is CSR accessible?
     82 020744  103422                             BCS     13$             ;Br if trap occurred while accessing CSR
     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 020746  062700  000000G         5$:       ADD     #H.INS,R0       ;Offset 200 in block 0
     89 020752  005710                             TST     @R0             ;Does any installation code exist?
     90 020754  001415                             BEQ     11$             ;Br if no driver installation code
     91 020756  013746  000000G                    MOV     @#RMON,-(SP)    ;Save RT-11 RMON pointer
     92 020762  012737  000000G 000000G           MOV     #MONVEC,@#RMON  ;Set TSX-Plus RMON pointer
     93 020770  013703  000000G                    MOV     RPRVEC,R3       ;Get pointer to Pro vec addr routine
     94 020774  004710                             CALL    @R0             ;Call the installation code
     95 020776  012637  000000G                    MOV     (SP)+,@#RMON    ;Restore RT-11 RMON pointer
     96 021002  000403                             BR      13$             ;C-flag now indicates handler load status
     97                                 ;
     98                                 ;  Finished with installation verification.
     99                                 ;
    100 021004  000261                  10$:      SEC                     ;Set c-bit for indicating handler
    101 021006  000401                             BR      13$             ;should not be installed
    102 021010  000241                  11$:      CLC                     ;Clear the c-bit for driver installation
    103 021012  012637  000010          13$:      MOV     (SP)+,@#10      ;Restore illegal instruction vector
    104 021016  012637  000004                    MOV     (SP)+,@#4       ;Restore the bus timeout vector
    105 021022  012605                             MOV     (SP)+,R5
    106 021024  012604                             MOV     (SP)+,R4
    107 021026  012603                             MOV     (SP)+,R3
    108 021030  012602                             MOV     (SP)+,R2
    109 021032  012601                             MOV     (SP)+,R1
    110 021034  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.
                                ;
                                ;  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).
                                ;
021036                          STDVTB:
                                ;
                                ;  Increment device counter
                                ;
021036  062737  000002  000000G         ADD     #2,NUMDEV       ;Say another device added to tables
021044  013700  000000G                 MOV     NUMDEV,R0       ;Get device index number
                                ;
                                ;  Set up PNAME and DVFLAG.
                                ;
021050  010260  000000G                 MOV     R2,PNAME(R0)    ;Set permanent device name
021054  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 I/O we do
                                ;  on behalf of the handler is relative to the base of the handler rather than
                                ;  relative to the start of the disk.
                                ;
021060  012760  000001  000000G         MOV     #1,HANDSK(R0)   ;Set block # of block 1 of handler file
                                ;
                                ;  Extract parameters from handler block 0
                                ;
021066  010004                          MOV     R0,R4           ;Carry device index in R4
021070  013700  000152'                 MOV     WRKBUF,R0       ;Point to block 0 of handler
021074  016064  000000G 000000G         MOV     H.SIZ(R0),HANSIZ(R4) ;Set handler size
021102  016064  000000G 000000G         MOV     H.DVSZ(R0),DEVSIZ(R4) ;Number of blocks on device
021110  016064  000000G 000000G         MOV     H.DSTS(R0),DVSTAT(R4) ;Set device status flags
                                ;
                                ;  Disable MOUNTs and data caching for certain devices
                                ;
021116  032764  000000G 000000G         BIT     #DS$DIR,DVSTAT(R4) ;Is this a directory structured device?
021124  001404                          BEQ     1$              ;Br if not -- No mounts allowed
021126  032764  000000G 000000G         BIT     #DS$NRD,DVSTAT(R4) ;Non RT-11 directory structure (mag tape)?
021134  001403                          BEQ     9$              ;Br if not
021136  052764  000000C 000000G 1$:     BIS     #<DX$NMT!DX$NCA>,DVFLAG(R4) ;Disable mounts and data caching
                                ;
                                ;  Finished
                                ;
```

```
     58 021144  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.
                                ;
021146  010346                  LDHNLO: MOV     R3,-(SP)
                                ;
                                ;  Determine if we have enough free memory space to read the handler
                                ;
021150  005064  000000G                 CLR     HANPAR(R4)      ;Say this handler is not mapped
021154  010500                          MOV     R5,R0           ;Get current top of memory address
021156  016403  000000G                 MOV     HANSIZ(R4),R3   ;Get size of handler
021162  060300                          ADD     R3,R0           ;Get address above top of handler
021164  004737  027622'                 CALL    CHKMEM          ;See if handler will fit in memory
                                ;
                                ;  Handler will fit.  Read it into memory.
                                ;
021170  006203                          ASR     R3              ;Get number of words to read
021172                                  .READW  #AREA,#1,R5,R3,#1
021226  103004                          BCC     1$              ;Br if read ok
021230  012700  001700'                 MOV     #ERHMSG,R0      ;Error reading handler
021234  000137  022124'                 JMP     HLERR           ;Abort initialization
                                ;
                                ;  Set address of handler entry point and compute address beyond
                                ;  end of the handler.
                                ;
021240  010564  000000G         1$:     MOV     R5,HANENT(R4)   ;Set address of handler entry point
021244  062764  000006  000000G         ADD     #6,HANENT(R4)   ;(Point to fourth word of handler)
021252  006303                          ASL     R3              ;Convert handler size to bytes
021254  060305                          ADD     R3,R5           ;Point beyond end of handler
                                ;
                                ;  Set up table of addresses of support routines at end of handler.
                                ;
021256  010503                          MOV     R5,R3           ;Get address past end of handler
021260  004737  022032'                 CALL    STHNPV          ;Set up pointer vector in handler
                                ;
                                ;  If handler has any load-time exectuion code, run it now
                                ;
021264  004737  022162'                 CALL    DOHNLC          ;Run any load-time code for handler
                                ;
                                ;  Finished
                                ;
021270  012603                          MOV     (SP)+,R3
021272  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.
                                ;
021274  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.
                                ;
021276  012704  000002                  MOV     #2,R4           ;Get index for first device entry
                                ;
                                ;  See if this device has a mapped handler
                                ;
021302  005764  000000G         1$:     TST     HANPAR(R4)      ;Is this handler mapped?
021306  001402                          BEQ     2$              ;Br if not
                                ;
                                ;  We found an entry for a device with a mapped handler.
                                ;  Load the handler.
                                ;
021310  004737  021332'                 CALL    LDHNHI          ;Load a mapped handler
                                ;
                                ;  Look for more mapped handlers
                                ;
021314  062704  000002          2$:     ADD     #2,R4           ;Increment device index
021320  020437  000000G                 CMP     R4,NUMDEV       ;Checked all devices?
021324  101766                          BLOS    1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
021326  012604                          MOV     (SP)+,R4
021330  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.
                                ;
021332  010146                  LDHNHI: MOV     R1,-(SP)
021334  010246                          MOV     R2,-(SP)
021336  010346                          MOV     R3,-(SP)
021340  010446                          MOV     R4,-(SP)
                                ;
                                ;  Open channel 1 to the handler file.
                                ;
021342  016437  000000G 000246'         MOV     PNAME(R4),HANNAM+2 ;Set the device name for the lookup
021350  016437  000000G 000146'         MOV     PNAME(R4),CURNAM;Set name in case we have an error
021356                                  .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
021376  103004                          BCC     8$              ;Br if we found the handler file
021400  012700  001636'                MOV     #CFHMSG,R0      ;Can't find handler
021404  000137  022124'                JMP     HLERR           ;Abort initialization
                                ;
                                ;  Read block 0 of the handler file and extract some information
                                ;
021410  013702  000152'         8$:     MOV     WRKBUF,R2       ;Get address of work buffer
021414                                  .READW  #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
021450  016237  000000G 000304'         MOV     H.GEN(R2),HGENFL;Save handler sysgen flags
                                ;
                                ;  Set virtual address of handler entry point
                                ;
021456  012764  000006G 000000G         MOV     #VPAR5+6,HANENT(R4) ;Set virtual addr of handler entry point
                                ;
                                ;  Get information about the size of the handler and determine the
                                ;  address in extended memory where the handler is to be loaded.
                                ;
021464  016402  000000G                 MOV     HANSIZ(R4),R2   ;Get size of handler (bytes)
021470  005202                          INC     R2              ;Make sure handler size is even
021472  042702  000001                  BIC     #1,R2
021476  010200                          MOV     R2,R0
021500  062700  000077                  ADD     #63.,R0         ;Round up to # 64-byte blocks
021504  072027  177772                  ASH     #-6,R0          ;Get # 64-byte blocks for handler
021510  060037  000000G                 ADD     R0,MHNSIZ       ;Accumulate total space for mapped handlers
021514  160005                          SUB     R0,R5           ;Reserve room for handler
021516  010564  000000G                 MOV     R5,HANPAR(R4)   ;Set mapping value for handler
021522  010537  000126'                 MOV     R5,HMAP         ;Set initial PAR base for handler
021526  012737  000001  000142'         MOV     #1,FILBLK       ;Set # of block to read from file
                                ;
                                ;  Begin loop to read handler into memory
                                ;
021534  010203                  1$:     MOV     R2,R3           ;Get remaining size of handler
021536  020327  001000                  CMP     R3,#512.        ;Compare with max we can read at one time
021542  101402                          BLOS    2$              ;Br if we can read remainder of handler
021544  012703  001000                  MOV     #512.,R3        ;Read one block
021550  160302                  2$:     SUB     R3,R2           ;Reduce amt of handler left to read
```

```
                                        ;
                                        ;  Read next block of handler
                                        ;
021552  006203                          ASR     R3              ;Get # words to read
021554  013701  000152'                MOV     WRKBUF,R1       ;Get address of buffer for read
021560                                  .READW  #AREA,#1,R1,R3,FILBLK ;Read a block
021614  103004                          BCC     3$              ;Br if read ok
021616  012700  001700'                MOV     #ERHMSG,R0      ;Get error message
021622  000137  022124'                JMP     HLERR           ;Abort initialization
                                        ;
                                        ;  Move the code we just read into the XM area for the handler
                                        ;
021626  012700  000000G         3$:     MOV     #VPAR5,R0       ;Get virtual address of mapped region
021632                                  DISABL                  ;** Disable interrupts **
021640  013746  000000G                 MOV     @#KPAR5,-(SP)   ;;;Save current mapping of PAR 5
021644  013737  000126' 000000G         MOV     HMAP,@#KPAR5    ;;;Set up mapping to get to XM area
021652  052737  000000G 000000G         BIS     #MMENBL,@#SROMMR;;;Enable memory management
021660  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
021664  001403                          BEQ     4$              ;;;Br if not
021666  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
021674  012120                  4$:     MOV     (R1)+,(R0)+     ;;;Move from WRKBUF to XM region
021676  077302                          SOB     R3,4$           ;;;Loop till all moved
021700  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
021704  001403                          BEQ     5$              ;;;Br if not
021706  042737  000000G 000000G         BIC     #EMMAP,@#SR3MMR ;;;Disable extended memory addressing
021714  042737  000000G 000000G 5$:     BIC     #MMENBL,@#SROMMR;;;Enable memory management
021722  012637  000000G                 MOV     (SP)+,@#KPAR5   ;;;Replace PAR 5 mapping
021726                                  ENABL                   ; ** Enable interrupts **
                                        ;
                                        ;  See if there is more to read
                                        ;
021734  062737  000010  000126'         ADD     #8.,HMAP        ;Increase XM region base
021742  005237  000142'                 INC     FILBLK          ;Increment file block number
021746  005702                          TST     R2              ;Is there more to read?
021750  001271                          BNE     1$              ;Loop if more to read
                                        ;
                                        ;  We have finished moving the handler into its XM region.
                                        ;  Set up addresses of system routines in a vector at the end of the handler
                                        ;
021752  012703  000000G                 MOV     #VPAR5,R3       ;Get virtual address of handler base
021756  066403  000000G                 ADD     HANSIZ(R4),R3   ;Get virtual address beyond end of handler
021762  004737  022662'                 CALL    HANMAP          ;;;Map KPAR5 to the handler
021766  004737  022032'                 CALL    STHNPV          ;;;Set up handler pointer vector
021772  004737  022736'                 CALL    HANUMP          ;Restore mapping
                                        ;
                                        ;  If handler has any load-time code, run it now
                                        ;
021776  010537  000134'                 MOV     R5,FMEMHI       ;Set addr of top of free memory area
022002  004737  022162'                 CALL    DOHNLC          ;Run any load-time code for handler
022006  013705  000134'                 MOV     FMEMHI,R5       ;Get new top of free memory address
                                        ;
                                        ;  Close the handler file
                                        ;
022012                                  .CLOSE  #1              ;Close the handler file
                                        ;
                                        ;  Finished
                                        ;
```

```
    115 022020  012604                          MOV     (SP)+,R4
    116 022022  012603                          MOV     (SP)+,R3
    117 022024  012602                          MOV     (SP)+,R2
    118 022026  012601                          MOV     (SP)+,R1
    119 022030  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.
                                ;
022032  010346                  STHNPV: MOV     R3,-(SP)
                                ;
                                ;  Set up addresses in the pointer vector
                                ;
022034  012743  000000G                 MOV     #FORK,-(R3)     ;Address of fork routine
022040  012743  000000G                 MOV     #INTEN,-(R3)    ;Address of inten routine
022044  032737  000000G 000304'         BIT     #SG$IOT,HGENFL  ;Does handler want timeout support?
022052  001402                          BEQ     2$              ;Br if not
022054  012743  000000G                 MOV     #IOTIMR,-(R3)   ;Set address of timeout support routine
022060  032737  000000G 000304' 2$:     BIT     #SG$ELG,HGENFL  ;Does handler want error logging support?
022066  001402                          BEQ     3$              ;Br if not
022070  012743  000000G                 MOV     #ERRLOG,-(R3)   ;Set address of error logging routine
022074  012743  000000G         3$:     MOV     #PTWRD,-(R3)
022100  012743  000000G                 MOV     #PTBYT,-(R3)
022104  012743  000000G                 MOV     #GTBYT,-(R3)
022110  012743  000000G                 MOV     #MPPHY,-(R3)
022114  012743  000000G                 MOV     #RELOC,-(R3)
                                ;
                                ;  Finished
                                ;
022120  012603                          MOV     (SP)+,R3
022122  000207                          RETURN
```

```
      1                                     ;
      2                                     ;  Error occured while loading device handler.
      3                                     ;  R0 = error message address;  CURNAM = device name.
      4                                     ;
      5 022124  010001                      HLERR:  MOV     R0,R1           ;SAVE ERROR MESSAGE ADDRESS
      6 022126                                      .PRINT  #TSXHD          ;PRINT ERROR MESSAGE HEADING
      7 022134                                      .PRINT  R1              ;PRINT ERROR MESSAGE
      8 022140  013700  000146'                     MOV     CURNAM,R0       ;GET RAD50 DEVICE NAME
      9 022144  004737  030012'                     CALL    PRTR50          ;PRINT DEVICE NAME
     10 022150                                      .PRINT  #CRLF
     11 022156  000137  004216'                     JMP     INISTP          ;ABORT INITIALIZATION
```

```
                                        .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.
                                ;
022162  010146                  DOHNLC: MOV     R1,-(SP)
022164  010246                          MOV     R2,-(SP)
022166  010346                          MOV     R3,-(SP)
022170  010446                          MOV     R4,-(SP)
022172  010546                          MOV     R5,-(SP)
                                ;
                                ;  Examine 1st word of handler to see if it could have any load-time code.
                                ;
022174  016405  000000G                 MOV     HANENT(R4),R5   ;Get address of handler entry point
022200  004737  022662'                 CALL    HANMAP          ;;;Map Kpar5 to handler if mapped handler
022204  016500  000004                  MOV     4(R5),R0        ;;;Get 1st instruction located at 4 in handler
022210  004737  022736'                 CALL    HANUMP          ;Restore normal mapping
022214  020027  000240                  CMP     R0,#240         ;Is it a NOP?
022220  103516                          BLO     7$              ;Br if can't be any load code
022222  020027  000277                  CMP     R0,#277         ;
022226  101113                          BHI     7$              ;Br if can't be any load code
022230  132700  000004                  BITB    #4,R0           ;Is there load code?
022234  001510                          BEQ     7$              ;Br if not
                                ;
                                ;  Handler may have load code.
                                ;  Read block 0 of handler and get offset to load code.
                                ;
022236  013702  000152'                 MOV     WRKBUF,R2       ;Get addr of our work buffer
022242                                  .READW  #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
022276  022227  031066                  CMP     (R2)+,#^RHAN    ;Is this a new type handler?
022302  001065                          BNE     7$              ;Br if not
022304  016203  000004                  MOV     4(R2),R3        ;Get offset to load code
022310  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.
                                ;
022312  020327  001000                  CMP     R3,#1000        ;Is load code in block 0 of handler?
022316  103424                          BLO     1$              ;Br if yes
022320  010302                          MOV     R3,R2           ;Get offset to start of load code
022322  072227  177767                  ASH     #-9.,R2         ;Convert to a block number
022326  042702  177400                  BIC     #^C377,R2       ;Clear all but block number
022332                                  .READW  #AREA,#1,WRKBUF,#512.,R2 ;Read 2 blocks from handler file
                                ;
                                ;  The load code is now in WRKBUF.  Set up and execute it.
                                ;
022370  010437  000144'         1$:     MOV     R4,CURDEV       ;Save current device index number
022374  042703  177000                  BIC     #^C777,R3       ;Get offset within block of load code entry pt
022400  010300                          MOV     R3,R0           ;Get entry point offset
022402  063700  000152'                 ADD     WRKBUF,R0       ;Add base address
```

```
     58 022406  013701  000000G                 MOV     RPRVEC,R1       ;Get pointer to GETVEC routine for Pro
     59 022412  012702  000000C                 MOV     #MAXDEV*2,R2    ;Get 2** entries in device tables
     60 022416  012703  000004                  MOV     #4,R3           ;Set code saying this is load code
     61 022422  012705  000000G                 MOV     #HANENT,R5      ;Point to handler entry address vector
     62 022426  060405                          ADD     R4,R5           ;Point to entry cell for this handler
     63 022430  004737  022662'                 CALL    HANMAP          ;;;Map Kpar5 to handler if it is a mapped
     64 022434  012704  022512'                 MOV     #LDREAD,R4      ;;;Get pointer to Read routine
     65 022440  004710                          CALL    (R0)            ;;;Execute the load code
     66 022442  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 022444  012737  001400  000000G         MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
     72 022452  004737  022736'                 CALL    HANUMP          ;Unmap the handler
     73 022456  000241                  7$:     CLC                     ;Clear the carry flag for return
     74 022460  000406                          BR      9$
     75                                 ;
     76                                 ;  Error occurred in fetch/load code
     77                                 ;
     78 022462  012737  001400  000000G 2$:      MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
     79 022470  004737  022736'                 CALL    HANUMP          ;Unmap the handler
     80 022474  000261                          SEC                     ;Set the carry flag for return
     81                                 ;
     82                                 ;  Finished
     83                                 ;
     84 022476  012605                  9$:     MOV     (SP)+,R5
     85 022500  012604                          MOV     (SP)+,R4
     86 022502  012603                          MOV     (SP)+,R3
     87 022504  012602                          MOV     (SP)+,R2
     88 022506  012601                          MOV     (SP)+,R1
     89 022510  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
                                ;
022512  010046                  LDREAD: MOV     R0,-(SP)
022514  010446                          MOV     R4,-(SP)        ;;;
022516  010004                          MOV     R0,R4           ;;;Get starting block number
                                ;
                                ;  Save current mapping information
                                ;
022520  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256?
022524  001402                          BEQ     1$              ;;;Br if not
022526  013746  000000G                 MOV     @#SR3MMR,-(SP)  ;;;Save extended memory address register
022532  013746  000000G          1$:    MOV     @#SROMMR,-(SP)  ;;;Save memory mapping
022536  013746  000000G                 MOV     @#KPAR5,-(SP)   ;;;Save current KPAR5 mapping
022542  013746  000000G                 MOV     @#KPAR6,-(SP)   ;;;Save current KPAR6 mapping
022546  012737  001400  000000G         MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
                                ;
                                ;  Turn off handler mapping
                                ;
022554  004737  022736'               CALL    HANUMP          ;Turn off handler mapping
                                ;
                                ;  Read the requested data from the handler
                                ;
022560                                .READW  #AREA,#1,R2,R1,R4 ;Read the blocks
022612  103420                        BCS     9$              ;Br if read error
                                ;
                                ;  Restore handler mapping
                                ;
022614  013704  000144'               MOV     CURDEV,R4       ;Get current device index
022620  004737  022662'               CALL    HANMAP          ;;;Map Kpar5 if necessary
                                ;
                                ;  Restore mapping information
                                ;
022624  012637  000000G               MOV     (SP)+,@#KPAR6   ;;;Restore KPAR6 mapping
022630  012637  000000G               MOV     (SP)+,@#KPAR5   ;;;Restore KPAR5 mapping
022634  012637  000000G               MOV     (SP)+,@#SROMMR  ;;;Restore memory mapping
022640  105737  000000G               TSTB    MEM256          ;;;Does machine have > 256?
022644  001402                        BEQ     2$              ;;;Br if not
022646  012637  000000G               MOV     (SP)+,@#SR3MMR  ;;;Restore extended memory address register
                                ;
                                ;  Finished
                                ;
022652  000241                  2$:   CLC                     ;;;Signal success on return
022654  012604                  9$:   MOV     (SP)+,R4
022656  012600                        MOV     (SP)+,R0
022660  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
                                ;
022662                          HANMAP:
                                ;
                                ;  Disable interrupts
                                ;
022662                                  DISABL                  ;;;Disable interrupts
                                ;
                                ;  Change RMON pointer to point to TSX-Plus vector
                                ;
022670  012737  000000G 000000G         MOV     #MONVEC,@#RMON  ;;;Say TSX-Plus is the monitor
                                ;
                                ;  See if this handler is mapped
                                ;
022676  005764  000000G                 TST     HANPAR(R4)      ;;;Is this handler mapped?
022702  001403                          BEQ     9$              ;;;Br if not
                                ;
                                ;  This handler is mapped.
                                ;  Set up mapping to access it.
                                ;
022704  016437  000000G 000000G         MOV     HANPAR(R4),@#KPAR5;;;Map KPAR5 to the handler code
022712  052737  000000G 000000G 9$:     BIS     #MMENBL,@#SROMMR;;;Enable memory mapping
022720  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
022724  001403                          BEQ     10$             ;;;Br if not
022726  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
                                ;
                                ;  Finished
                                ;
022734  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.
                                ;
022736                          HANUMP:
                                ;
                                ;  Turn off memory management
                                ;
022736  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
022742  001403                          BEQ     1$              ;;;Br if not
022744  042737  000000G 000000G         BIC     #EMMAP,@#SR3MMR ;;;Turn off extended memory addressing
022752  042737  000000G 000000G 1$:     BIC     #MMENBL,@#SROMMR;;;Turn off memory mapping
022760  012737  001200  000000G         MOV     #1200,@#KPAR5   ;;;Reset KPAR5 to its normal mapping
```

```
      58                                        ;
      59                                        ;  Restore RMON pointer to RT11
      60                                        ;
      61 022766  013737  000046' 000000G              MOV     RTMNVC,@#RMON   ;;;Reset RMON pointer
      62                                        ;
      63                                        ;  Enable interrupts
      64                                        ;
      65 022774                                       ENABL                   ;Enable interrupts
      66                                        ;
      67                                        ;  Finished
      68                                        ;
      69 023002  000207                               RETURN
```

```
                                        .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.
                                ;
023004  010246                  FNDHRB: MOV     R2,-(SP)
                                ;
                                ;  Search for specified RCB and also remember if we see a free RCB
                                ;
023006  005002                          CLR     R2              ;Say no free RCB found
023010  013701  000000G                 MOV     HANRCB,R1       ;Point to start of RCB area
023014  005721                          TST     (R1)+           ;Skip over -1 word at front
023016  005711                  1$:     TST     (R1)            ;What is the status of this RCB
023020  001002                          BNE     2$              ;Br if this is not a free RCB
023022  010102                          MOV     R1,R2           ;Remember address of a free RCB
023024  000412                          BR      3$
023026  021127  177777          2$:     CMP     (R1),#-1        ;Are we at the end of the list?
023032  001412                          BEQ     4$              ;Br if yes
023034  021561  000006                  CMP     (R5),6(R1)      ;Compare the names
023040  001004                          BNE     3$              ;Br if don't match
023042  026561  000002  000010          CMP     2(R5),10(R1)    ;Compare 2nd half of name
023050  001417                          BEQ     6$              ;Br if found the RCB we were searching for
023052  062701  000012          3$:     ADD     #12,R1          ;Point to the next RCB
023056  000757                          BR      1$              ;Continue searching
                                ;
                                ;  We could not find the specified RCB.
                                ;  If there was a free one, initialize the name.
                                ;
023060  010201                  4$:     MOV     R2,R1           ;Was there a free RCB?
023062  001410                          BEQ     5$              ;Br if not
023064  011561  000006                  MOV     (R5),6(R1)      ;Set name in the RCB
023070  016561  000002  000010          MOV     2(R5),10(R1)    ;(2nd word of name)
023076  063761  000144' 000010          ADD     CURDEV,10(R1)   ;Make name unique to device
023104  000261                  5$:     SEC                     ;Signal that we did not find the RCB
023106  000401                          BR      9$
                                ;
                                ;  We found the RCB
                                ;
023110  000241                  6$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
023112  012602                  9$:     MOV     (SP)+,R2
023114  000207                          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.
                                ;
023116  010046                 HANXMR: MOV     R0,-(SP)
                                ;
                                ;  Get the total amount of free memory space available now
                                ;  and see if the requested region can be allocated.
                                ;
023120  013700  000134'                 MOV     FMEMHI,R0       ;Top of free memory
023124  163700  000136'                 SUB     FMEMLO,R0       ;-Base of free memory
023130  020200                          CMP     R2,R0           ;Do we have room for the requested region?
023132  101006                          BHI     8$              ;Br if not
                                ;
                                ;  There is room for the region so allocate it from the top of memory
                                ;
023134  160237  000134'                 SUB     R2,FMEMHI       ;Allocate the region
023140  013701  000134'                 MOV     FMEMHI,R1       ;Return the address of the base of the region
023144  000241                          CLC                     ;Signal success on return
023146  000402                          BR      9$
                                ;
                                ;  We are not able to allocate the region.
                                ;  Return with C-flag set and the size of the largest possible region in R2.
                                ;
023150  010002                 8$:     MOV     R0,R2           ;Get the size of the largest possible region
023152  000261                          SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
023154  012600                 9$:     MOV     (SP)+,R0
023156  000207                          RETURN
```

```
                                        .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$MAP flag is set in the DVFLAG word for devices that require mapping.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to low-memory free area
                                ;
                                ;  Outputs:
                                ;   MIOFLAG = 0==>I/O mapping not required for any device;
                                ;             1==>I/O mapping required for some device.
                                ;   R5 = Pointer to new low-memory free area.
                                ;
023160  010146                  SETMIO: MOV     R1,-(SP)
                                ;
                                ;  Determine if this machine requires mapping at all
                                ;
023162  005727  000000G                 TST     #MIODBG         ;Are we debugging mapped I/O system?
023166  001017                          BNE     2$              ;Br if yes
023170  032737  000001  000306'         BIT     #EXTLSI,ICONFG  ;Is this a Q-bus machine with more than 256Kb?
023176  001013                          BNE     2$              ;Br if yes
                                ;
                                ;  This is not a Q-bus system with more than 256Kb.
                                ;  Mapping is not required at all.
                                ;
023200  012701  000002                  MOV     #2,R1           ;Get initial device index number
023204  042761  000000G 000000G 1$:     BIC     #DX$MAP,DVFLAG(R1) ;Clear mapped-I/O flag
023212  062701  000002                  ADD     #2,R1           ;Get next device index
023216  020137  000000G                 CMP     R1,NUMDEV       ;More to do?
023222  101770                          BLOS    1$              ;Br if yes
023224  000464                          BR      9$
                                ;
                                ;  This is a Q-bus system with more than 256Kb.
                                ;  See if any devices have requested mapped I/O.
                                ;
023226  005000                  2$:     CLR     R0              ;Clear composite flag word
023230  012701  000002                  MOV     #2,R1           ;Initialize device index number
023234  056100  000000G         3$:     BIS     DVFLAG(R1),R0   ;Combine flags from all devices
023240  062701  000002                  ADD     #2,R1           ;Get next device index number
023244  020137  000000G                 CMP     R1,NUMDEV       ;Checked all devices?
023250  101771                          BLOS    3$              ;Br if not
023252  032700  000000G                 BIT     #DX$MAP,R0      ;Does any device require mapping?
023256  001447                          BEQ     9$              ;Br if not
                                ;
                                ;  I/O mapping is required
                                ;
023260  105237  000000G                 INCB    MIOFLG          ;Remember that mapping is required
                                ;
                                ;  Zero the area where we will build the control structures
                                ;
023264  113701  000000G                 MOVB    VMIOBF,R1       ;Get # buffers wanted
023270  070127  000000G                 MUL     #MI$$SZ,R1      ;Times size for each control block
023274  062701  000000C                 ADD     #MIONWB*MW$$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
                                ;
                                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$FLG(seg) = 0==>Load seg into high memory; 1==>Load over TSINIT.
                                ;   OVLBAS = Address of location within TSINIT where overlays start.
                                ;   R5 = Pointer past last overlay loaded over TSINIT.
                                ;
023402  010146                 OVLPOS: MOV     R1,-(SP)
023404  010246                         MOV     R2,-(SP)
023406  010346                         MOV     R3,-(SP)
023410  010446                         MOV     R4,-(SP)
023412  010504                         MOV     R5,R4           ;Get address where we may load overlays
                                ;
                                ;  Build the table that holds information about the overlays
                                ;
023414  004737  023612'                 CALL    OVLBLD          ;Build overlay information table
                                ;
                                ;  First determine how much space will be used by those overlays that are
                                ;  forced to be loaded over TSINIT.
                                ;
023420  062704  000077                  ADD     #63.,R4         ;Bound address to 64-byte boundary
023424  042704  000077                  BIC     #77,R4
023430  010437  000140'                 MOV     R4,OVLBAS       ;Remember address where we load overlays
023434  012702  000576'                 MOV     #OSTABL,R2      ;Point to start of table
023440  005762  000000         1$:     TST     OS$SIZ(R2)      ;Is this overlay to be loaded?
023444  001423                         BEQ     2$              ;Br if not
023446  016200  000004                 MOV     OS$OVL(R2),R0   ;Point to linker-built entry
023452  016000  000000G                MOV     0.ADR(R0),R0    ;Get Rad50 segment ID
023456  012701  001026'                 MOV     #LOWOVL,R1      ;Point to table of overlays to go over TSINIT
023462  020021                 6$:     CMP     R0,(R1)+        ;Must this overlay go over TSINIT?
023464  001404                         BEQ     4$              ;Br if yes
023466  020127  001034'                 CMP     R1,#LOWEND      ;End of low-overlay table?
023472  103773                         BLO     6$              ;Br if not
023474  000407                         BR      2$              ;This overlay is not forced over TSINIT
023476  005262  000002         4$:     INC     OS$FLG(R2)      ;Set flag saying load over TSINIT
023502  016200  000000                 MOV     OS$SIZ(R2),R0   ;Get # 64-byte blocks needed for overlay
023506  072027  000006                 ASH     #6,R0           ;Get # bytes needed for overlay
023512  060004                         ADD     R0,R4           ;Advance address within TSINIT
023514  062702  000006         2$:     ADD     #OS$$SZ,R2      ;Point to entry for next segment
023520  020237  001024'                 CMP     R2,OSLAST       ;Have we finished?
023524  103745                         BLO     1$              ;Loop if not
                                ;
                                ;  Determine how much memory space is available in TSINIT for other overlays
                                ;
023526  020427  030004'                 CMP     R4,#INITOP-50.  ;Any space left for other overlays?
023532  103021                         BHIS    9$              ;Br if not
023534  012705  030066'                 MOV     #INITOP,R5      ;Point to top of overlay area
023540  160405                         SUB     R4,R5           ;Total space available for overlays
023542  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.
                                        ;
023546  004737  025004'        3$:     CALL    OVLTRY          ;Try to find largest overlay that will fit
023552  103411                          BCS     9$              ;Br if no more overlays will fit
023554  005262  000002                  INC     OS$FLG(R2)      ;Remember to load over TSINIT
023560  016200  000000                  MOV     OS$SIZ(R2),R0   ;Get # 64-byte blocks needed for overlay
023564  160005                          SUB     R0,R5           ;Reduce remaining free space in TSINIT
023566  072027  000006                  ASH     #6,R0           ;Get # bytes needed for overlay
023572  060004                          ADD     R0,R4           ;Advance overlay address in TSINIT
023574  000764                          BR      3$              ;See if we can find more segments to load
                                        ;
                                        ;  Finished
                                        ;
023576  010405                  9$:     MOV     R4,R5           ;Return top-of-overlay address in R5
023600  012604                          MOV     (SP)+,R4
023602  012603                          MOV     (SP)+,R3
023604  012602                          MOV     (SP)+,R2
023606  012601                          MOV     (SP)+,R1
023610  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.
                                ;
023612  010146                  OVLBLD: MOV     R1,-(SP)
023614  010246                          MOV     R2,-(SP)
023616  010346                          MOV     R3,-(SP)
                                ;
                                ;  Read 1st block of SAV file to get pointer to overlay table
                                ;
023620  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
023624                                  .READW  #AREA,#17,R2,#256.,#0 ;read first block of the save file
023660  103444                          BCS     22$             ;Br if error on read
023662  016201  000064                  MOV     64(R2),R1       ;point to the overlay table
023666  001012                          BNE     15$             ;br if overlays exist
                                ;
                                ;  Must be verion 3B overlays structure at absolute location.
                                ;
023670  012737  000137  001000          MOV     #137,@#1000     ;position jump instrucion over 3b ovly handler
023676  012737  000000G 001002          MOV     #$OVRH,@#1002   ;position overlay intercept location
023704  012701  001104                  MOV     #1104,R1        ;point to the overlay table
023710  010137  000000G                 MOV     R1,OVRADD       ;save the address of the overlay table
                                ;
                                ;  Initialize the table that holds information about the overlays
                                ;
023714  012703  000576'         15$:    MOV     #OSTABL,R3      ;Point to table for overlay info
023720  010163  000004          11$:    MOV     R1,OS$OVL(R3)   ;Save pointer to overlay control block
023724  005063  000002                  CLR     OS$FLG(R3)      ;Assume seg will be loaded in high memory
023730  004737  024132'                 CALL    ALCOVL          ;Determine if we should load this overlay
023734  010263  000000                  MOV     R2,OS$SIZ(R3)   ;Remember total size of overlay+data
023740  062703  000006                  ADD     #OS$$SZ,R3      ;Point to next overlay table entry
023744  062701  000006          12$:    ADD     #6,R1           ;find the next region
023750  021127  004537                  CMP     (R1),#4537      ;compare with a <JSR R5,$OVRH> instruction
023754  001361                          BNE     11$             ;Br if not at end
023756  010337  001024'                 MOV     R3,OSLAST       ;Save pointer past last overlay table entry
                                ;
                                ;  Finished
                                ;
023762  012603                          MOV     (SP)+,R3
023764  012602                          MOV     (SP)+,R2
023766  012601                          MOV     (SP)+,R1
023770  000207                          RETURN
                                ;
                                ;  Error -- Read error occured while reading overlay table
                                ;
023772                          22$:    .PRINT  #TSXHD
024000                                  .PRINT  #RDERR
024006  000137  004216'                 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.
                                ;
024012  010146                  GETMAP: MOV     R1,-(SP)
024014  010246                          MOV     R2,-(SP)
024016  010346                          MOV     R3,-(SP)
024020  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.
                                ;
024024  012703  000576'                  MOV     #OSTABL,R3      ;Point to 1st overlay table entry
024030  004737  024276'          1$:     CALL    OPTOVL          ;See if this segment should be loaded
024034  010263  000000                    MOV     R2,OS$SIZ(R3)   ;Save # 64-byte blocks needed for overlay
024040  062703  000006                    ADD     #OS$$SZ,R3      ;Point to next overlay table entry
024044  020337  001024'                    CMP     R3,OSLAST       ;Checked all entries in overlay table?
024050  103767                           BLO     1$              ;Br if not
                                ;
                                ;  Load those overlays that go into high memory
                                ;
024052  012702  000576'                  MOV     #OSTABL,R2      ;Point to 1st overlay entry
024056  005762  000000          3$:     TST     OS$SIZ(R2)      ;Is this overlay segment wanted?
024062  001405                           BEQ     4$              ;Br if not
024064  005762  000002                    TST     OS$FLG(R2)      ;Load over TSINIT or into high memory?
024070  001002                           BNE     4$              ;Br if load over TSINIT
024072  004737  025102'                    CALL    GETOVL          ;Load overlay into high memory
024076  062702  000006          4$:     ADD     #OS$$SZ,R2      ;Point to next overlay table entry
024102  020237  001024'                    CMP     R2,OSLAST       ;Have we done all overlays?
024106  103763                           BLO     3$              ;Loop if not
                                ;
                                ;  Finished
                                ;
024110  013700  000000G          19$:    MOV     SMRSIZ,R0       ;Get memory pointer at start of allocation
024114  160500                           SUB     R5,R0           ;Calc amt of space allocated
024116  010037  000000G                    MOV     R0,SMRSIZ       ;Save total space used for mapped regions
024122  012603                           MOV     (SP)+,R3
024124  012602                           MOV     (SP)+,R2
024126  012601                           MOV     (SP)+,R1
024130  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$xxx)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> This segment is to be loaded.
                                ;   C-flag set     ==> Do not load this overlay segment.
                                ;   R2 = # 64-Byte blocks needed for segment including data areas within it.
                                ;
024132  010146                  ALCOVL: MOV     R1,-(SP)
                                ;
                                ;  Get pointer to linker-build overlay entry for segment
                                ;
024134  016301  000004                  MOV     OS$OVL(R3),R1   ;Get pointer to linker-built entry for seg
                                ;
                                ;  Read in the first block of the overlay segment
                                ;
024140  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
024144                                  .READW  #AREA,#17,R2,#256.,0.BLK(R1) ;read the first block
024202  103415                          BCS     3$              ;Br if read error
                                ;
                                ;  Save the 3 character Rad50 segment ID in the O.ADR cell of the
                                ;  linker-built overlay table entry for this segment.
                                ;
024204  016261  000002  000000G         MOV     2(R2),O.ADR(R1) ;save the rad50 overlay identifier
                                ;
                                ;  Make sure the segment is not larger than 8Kb
                                ;
024212  016102  000000G                 MOV     O.SIZ(R1),R2    ;get the word count of the code region
024216  006302                          ASL     R2              ;convert to byte count
024220  020227  020000                  CMP     R2,#20000       ;check for 8kb overflow
024224  101014                          BHI     21$             ;Br if region is too big
                                ;
                                ;  Don't load some optional segments if features were not selected
                                ;  in TSGEN.
                                ;
024226  004737  024276'                 CALL    OPTOVL          ;See if we want to load this segment
                                ;
                                ;  Finished
                                ;  The C-flag is set or reset by OPTOVL.
                                ;
024232  012601                          MOV     (SP)+,R1
024234  000207                          RETURN
                                ;
                                ;  Error -- Error on reading from SAV file
                                ;
024236                          3$:     .PRINT  #TSXHD          ;Print heading
024244                                  .PRINT  #RDERR          ;Read error
024252  000137  004216'                JMP     INISTP          ;Abort initialization
                                ;
                                ;  Error -- Insufficient memory space to load run-time systems
                                ;
```

```
     58 024256                          21$:    .PRINT  #TSXHD          ;PRINT HEADING
     59 024264                                  .PRINT  #TSXSIZ         ;PRINT ERROR MESSAGE
     60 024272  000137  004216'                 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$xxx)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Load this overlay.
                                ;   C-flag set     ==> Do not load this overlay.
                                ;   R2 = # 64-byte blocks needed for code + data for the segment.
                                ;
024276  010346                  OPTOVL: MOV     R3,-(SP)
024300  010446                          MOV     R4,-(SP)
024302  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get the name of the overlay segment
                                ;
024304  016305  000004                  MOV     OS$OVL(R3),R5   ;Get pointer to linker-built entry
024310  016504  000000G                 MOV     O.ADR(R5),R4    ;Get name of the segment
                                ;
                                ;  Get size of code portion of overlay segment
                                ;
024314  016502  000000G                 MOV     O.SIZ(R5),R2    ;Get # words needed by code portion of seg
024320  006302                          ASL     R2              ;Convert to # bytes
                                ;
                                ;  See if this is an optional segment that we need to deal with specially
                                ;
024322  012700  024346'                 MOV     #OVLLST,R0      ;Point to overlay name list
024326  020420                  1$:     CMP     R4,(R0)+        ;Found name of overlay?
024330  001405                          BEQ     2$              ;Br if yes
024332  005720                          TST     (R0)+           ;No -- Skip over address word
024334  020027  024432'                 CMP     R0,#OVLEND      ;Checked all names in the list?
024340  103772                          BLO     1$              ;Loop if not
024342  000577                          BR      OOXYES          ;Load this overlay
                                ;
                                ;  Branch off to processing routine
                                ;
024344  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
                                ;
024346                          OVLLST:
024346                                  OVLTBL  USR             ;TSUSR  -- File management
024352                                  OVLTBL  SPL             ;TSSPOL -- Spooling system
024356                                  OVLTBL  LOK             ;TSLOCK -- Shared file record locking
024362                                  OVLTBL  MSG             ;TSMSG  -- Inter-job message communication
024366                                  OVLTBL  SWP             ;TSSWAP -- Job swapper
024372                                  OVLTBL  PLS             ;TSPLAS -- PLAS support
024376                                  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
                                ;
024432  013703  000000G         OORUSR: MOV     VNFCSH,R3       ;Get # file cache entries
024436  070327  000000G                 MUL     #FC$$SZ,R3      ;Multiply by size of each entry
024442  060302                         ADD     R3,R2           ;Allocate space for directory cache
024444  000536                         BR      OOXYES          ;Load the segment
                                ;
                                ;  Spooling system
                                ;
024446  005727  000000G         OORSPL: TST     #SPLND          ;Are there any spooled devices?
024452  001530                         BEQ     OOXNO           ;Br if not
024454  062702  000000C                 ADD     #<SPLNB*512.>,R2;Reserve room for spool buffers
024460  013703  000000G                 MOV     NSPLBL,R3       ;Get # blocks for spool file
024464  062703  000007                 ADD     #7,R3           ;Bound up to byte boundary
024470  072327  177775                 ASH     #-3,R3          ;Divide by 8 to get # bytes for table
024474  005203                         INC     R3              ;Round up to word boundary
024476  042703  000001                 BIC     #1,R3
024502  060302                         ADD     R3,R2           ;Add space for spool file allocation table
024504  000516                         BR      OOXYES          ;Load the segment
                                ;
                                ;  Record locking system
                                ;
024506  005737  000000G         OORLOK: TST     VMXSF           ;Any shared files?
024512  001510                         BEQ     OOXNO           ;Br if not
024514  005737  000000G                 TST     VNUMDC          ;Shared file data caching wanted?
024520  001110                         BNE     OOXYES          ;Br if yes
024522  162702  000000G                 SUB     #DCCSIZ,R2      ;Reduce size of segment - Leave out cache code
024526  000505                         BR      OOXYES          ;Load the segment
                                ;
                                ;  Message communication system
                                ;
024530  013703  000000G         OORMSG: MOV     VMAXMC,R3       ;Is message communication facility wanted?
024534  001477                         BEQ     OOXNO           ;Br if not
024536  070327  000000G                 MUL     #MB$$SZ,R3      ;Space for message channel blocks
024542  060302                         ADD     R3,R2
024544  013703  000000G                 MOV     VMXMRB,R3       ;Number of message request blocks
024550  070327  000000G                 MUL     #MR$$SZ,R3      ;Times size of request block
024554  060302                         ADD     R3,R2
024556  013703  000000G                 MOV     VMSCHR,R3       ;Max # chars in a message
024562  005203                         INC     R3              ;Bound up to word
024564  042703  000001                 BIC     #1,R3           ;Reserve whole number of words
024570  062703  000000G                 ADD     #MU$TXT,R3      ;Plus space for message header
024574  070337  000000G                 MUL     VMXMSG,R3       ;Times maximum number of messages
024600  060302                         ADD     R3,R2           ;Space for message buffers
024602  000457                         BR      OOXYES
                                ;
                                ;  PLAS support
                                ;
024604  013703  000000G         OORPLS: MOV     VPLAS,R3        ;PLAS support wanted?
```

```
     115 024610  001451                        BEQ     OOXNO           ;Br if not
     116 024612  062703  000021                ADD     #17.,R3         ;Bound up # blocks
     117 024616  072327  177775                ASH     #-3,R3          ;Get # bytes needed for swap file bit map
     118 024622  060302                        ADD     R3,R2           ;Reserve room for swap file bit map
     119 024624  000446                        BR      OOXYES          ;Load the segment
     120                             ;
     121                             ;  Job swapper
     122                             ;
     123 024626  105737  000000G     OORSWP: TSTB    VSWPFL          ;Is this a swapping system?
     124 024632  001440                        BEQ     OOXNO           ;Br if not
     125 024634  000442                        BR      OOXYES          ;Br if yes -- Load the segment
     126                             ;
     127                             ;  Single line editor
     128                             ;
     129 024636  105737  000000G     OORSLE: TSTB    VSLEDT          ;Is SL editor wanted?
     130 024642  001434                        BEQ     OOXNO           ;Br if not
     131 024644  000436                        BR      OOXYES          ;Load the segment
     132                             ;
     133                             ;  Display windows
     134                             ;
     135 024646  013703  000000G     OORWIN: MOV     VMXWIN,R3       ;Are any display windows wanted?
     136 024652  001430                        BEQ     OOXNO           ;Br if not
     137 024654  070327  000000G               MUL     #DW$$SZ,R3      ;Amt of space needed for window control blks
     138 024660  060302                        ADD     R3,R2           ;Add to size of overlay
     139 024662  000427                        BR      OOXYES          ;Load the segment
     140                             ;
     141                             ;  Mapped I/O
     142                             ;
     143 024664  105737  000000G     OORMIO: TSTB    MIOFLG          ;Is I/O mapping needed?
     144 024670  001421                        BEQ     OOXNO           ;Br if not
     145 024672  000423                        BR      OOXYES          ;Load the segment
     146                             ;
     147                             ;  CL handler
     148                             ;
     149 024674  005727  000000G     OORCLO: TST     #CLTOTL         ;Any I/O lines?
     150 024700  001415                        BEQ     OOXNO           ;Br if not
     151 024702  000417                        BR      OOXYES          ;Yes, load the segment
     152                             ;
     153                             ;  Program debugger
     154                             ;
     155 024704  105737  000000G     OORDBG: TSTB    VDBFLG          ;Is the program debugger wanted?
     156 024710  001411                        BEQ     OOXNO           ;Br if not
     157 024712  000413                        BR      OOXYES          ;Load this segment
     158                             ;
     159                             ;  Data caching
     160                             ;
     161 024714  005737  000000G     OORCSH: TST     CSHALC          ;Is data caching wanted?
     162 024720  001405                        BEQ     OOXNO           ;Br if not
     163 024722  000407                        BR      OOXYES          ;Load this segment
     164                             ;
     165                             ;  Crash dump generator
     166                             ;
     167 024724  105737  000000G     OORDMP: TSTB    VSYDMP          ;Is dump facility wanted?
     168 024730  001401                        BEQ     OOXNO           ;Br if not
     169 024732  000403                        BR      OOXYES          ;Br if yes
     170                             ;
     171                             ;  Don't load this segment
```

```
    172                                 ;
    173 024734  005002                  OOXNO:  CLR     R2              ;Say no space needed for overlay
    174 024736  000261                          SEC                     ;Signal don't load the segment
    175 024740  000415                          BR      OOXFIN
    176                                 ;
    177                                 ;  Load this segment
    178                                 ;
    179 024742  005202                  OOXYES: INC     R2              ;Make sure size is even
    180 024744  042702  000001                  BIC     #1,R2
    181 024750  020227  020000                  CMP     R2,#8192.       ;Don't allow code + data to exceed 8Kb
    182 024754  101402                          BLOS    1$              ;Br if ok
    183 024756  012702  020000                  MOV     #8192.,R2       ;Note, init code in segment will truncate dat
    184 024762  062702  000077          1$:     ADD     #63.,R2         ;Convert to # 64-byte blocks
    185 024766  072227  177772                  ASH     #-6,R2
    186 024772  000241                          CLC                     ;Signal to load the segment
    187                                 ;
    188                                 ;  Finished
    189                                 ;
    190 024774  012605                  OOXFIN: MOV     (SP)+,R5
    191 024776  012604                          MOV     (SP)+,R4
    192 025000  012603                          MOV     (SP)+,R3
    193 025002  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.
                                ;
025004  010346                  OVLTRY: MOV     R3,-(SP)
                                ;
                                ;  Begin loop to examine all segments
                                ;
025006  005002                          CLR     R2              ;Say we haven't found any segment yet
025010  012703  000576'                 MOV     #OSTABL,R3      ;Point to entry for 1st segment
025014  005763  000000          1$:     TST     OS$SIZ(R3)      ;Is this segment to be loaded?
025020  001415                          BEQ     2$              ;Br if not
025022  005763  000002                  TST     OS$FLG(R3)      ;Is this segment already over TSINIT?
025026  001012                          BNE     2$              ;Br if yes
025030  026305  000000                  CMP     OS$SIZ(R3),R5   ;Will this segment fit?
025034  101007                          BHI     2$              ;Br if not
025036  005702                          TST     R2              ;Have we found any other seg yet?
025040  001404                          BEQ     3$              ;Br if not
025042  026362  000000  000000          CMP     OS$SIZ(R3),OS$SIZ(R2) ;Is new seg larger than old?
025050  101401                          BLOS    2$              ;Br if not
025052  010302                  3$:     MOV     R3,R2           ;Remember largest segment
025054  062703  000006          2$:     ADD     #OS$$SZ,R3      ;Point to entry for next segment
025060  020337  001024'                 CMP     R3,OSLAST       ;Have we checked all segments?
025064  103753                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
025066  000241                          CLC                     ;Assume we found a segment
025070  005702                          TST     R2              ;Did we find a segment that will fit?
025072  001001                          BNE     9$              ;Br if yes
025074  000261                          SEC                     ;Signal failure on return
025076  012603                  9$:     MOV     (SP)+,R3
025100  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.
                                ;
025102                          GETOVL:
                                ;
                                ;  Allocate space for the overlay segment
                                ;
025102  166205  000000                  SUB     OS$SIZ(R2),R5   ;Allocate space for overlay
025106  020527  001600                  CMP     R5,#1600        ;Are we about to run over RT-11?
025112  103405                          BLO     10$             ;Br if yes -- Insufficient memory
                                ;
                                ;  Remember the base address of some key segments
                                ;
025114  004737  004604'                 CALL    KEYSEG          ;Remember address of some segments
                                ;
                                ;  Load the segment
                                ;
025120  004737  025146'                 CALL    LODOVL          ;Load the segment
                                ;
                                ;  Finished
                                ;
025124  000207                          RETURN
                                ;
                                ;  Error: Memory overflow
                                ;
025126                          10$:    .PRINT  #TSXHD
025134                                  .PRINT  #TSXSIZ
025142  000137  004216'                 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.
                                ;
025146  010146                  LODOVL: MOV     R1,-(SP)
025150  010246                          MOV     R2,-(SP)
025152  010346                          MOV     R3,-(SP)
025154  010446                          MOV     R4,-(SP)
025156  010546                          MOV     R5,-(SP)

                                ;
                                ;  Get info about size of the overlay and position within SAV file
                                ;
025160  016201  000004                  MOV     OS$OVL(R2),R1   ;Get pointer to linker-built segment entry
025164  016103  000000G                 MOV     O.SIZ(R1),R3    ;Get size of overlay segment (# words)
025170  016137  000000G 000142'         MOV     O.BLK(R1),FILBLK;Get block in SAV file where segment starts
025176  010302                          MOV     R3,R2           ;Get total number of words in segment
025200  062702  000377                  ADD     #255.,R2        ;round to the nearest number of blocks
025204  000302                          SWAB    R2              ;Divide by 256. words per segment
025206  042702  177400                  BIC     #177400,R2      ;kill sign extension bits
025212  010561  000000G                 MOV     R5,O.PAR(R1)    ;Remember where segment is being loaded
                                ;
                                ;  Read next block of overlay segment into low-memory buffer
                                ;
025216  013704  000152'         10$:    MOV     WRKBUF,R4       ;Point to work buffer
025222                                  .READW  #AREA,#17,R4,#256.,FILBLK       ;read a block
025260  103466                          BCS     22$             ;read error occured
                                ;
                                ;  Move from low buffer to high position in memory
                                ;
025262  012701  000000G                 MOV     #VPAR5,R1       ;get the virtual address of the mapped region
025266  012700  000400                  MOV     #256.,R0        ;obtain the number of words to move
025272  020300                          CMP     R3,R0           ;Do we need to move as many as 256 words?
025274  103001                          BHIS    2$              ;Br if yes
025276  010300                          MOV     R3,R0           ;Get number of words to move for last block
025300  160003                  2$:     SUB     R0,R3           ;Get number of words left after this move
025302                                  DISABL                  ;** Disable interrupts **
025310  013746  000000G                 MOV     @#KPAR5,-(SP)   ;save the contents of the mapping register
025314  010537  000000G                 MOV     R5,@#KPAR5      ;change the mapping register
025320  052737  000000G 000000G         BIS     #MMENBL,@#SROMMR;enable memory management
025326  105737  000000G                 TSTB    MEM256          ;Does machine have at least 256Kb of memory?
025332  001403                          BEQ     11$             ;Br if not
025334  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;enable extended memory addressing
025342  012421                  11$:    MOV     (R4)+,(R1)+     ;move into high memory
025344  077002                          SOB     R0,11$
025346  105737  000000G                 TSTB    MEM256          ;Does this machine have at least 256Kb?
025352  001403                          BEQ     12$             ;Br if not
025354  042737  000000G 000000G         BIC     #EMMAP,@#SR3MMR ;disable extended memory management
025362  042737  000000G 000000G 12$:    BIC     #MMENBL,@#SROMMR;disable memory management
025370  012637  000000G                 MOV     (SP)+,@#KPAR5   ;restore the mapping register
025374                                  ENABL                   ;** Enable interrupts **
025402  062705  000010                  ADD     #10,R5          ;advance 64-byte block # by 512-bytes
025406  005237  000142'                 INC     FILBLK          ;increment file block #
```

```
     58 025412  005302                        DEC     R2              ;More to be copied?
     59 025414  001402                        BEQ     5$              ;Br if not
     60 025416  000137  025216'                JMP     10$             ;Read and copy rest of mapped segment
     61                                ;
     62                                ;  Finished loading the segment
     63                                ;
     64 025422  012605                 5$:     MOV     (SP)+,R5
     65 025424  012604                         MOV     (SP)+,R4
     66 025426  012603                         MOV     (SP)+,R3
     67 025430  012602                         MOV     (SP)+,R2
     68 025432  012601                         MOV     (SP)+,R1
     69 025434  000207                         RETURN
     70                                ;
     71                                ;  Error occurred on read
     72                                ;
     73 025436                         22$:    .PRINT  #TSXHD          ;Print heading
     74 025444                                 .PRINT  #RDERR          ;Read error
     75 025452  000137  004216'                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
                                ;
025456  010146                  GETSRT: MOV     R1,-(SP)
025460  010246                          MOV     R2,-(SP)
025462  010346                          MOV     R3,-(SP)
025464  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if this is a dummy run-time entry to allow for patching
                                ;
025466  021127  000000G                 CMP     (R1),#DMYDEV    ;Dummy run-time entry?
025472  001540                          BEQ     7$              ;Br if yes
                                ;
                                ;  Try to open a channel to run-time file
                                ;
025474                                  .LOOKUP #AREA,#1,R1     ;OPEN CHANNEL TO RUN-TIME FILE
025512  103010                          BCC     8$              ;BR IF OPEN WAS SUCCESSFUL
                                ;
                                ;  Cannot open shared run-time file.
                                ;  See if he wants to abort or continue.
                                ;
025514  105737  000000G                 TSTB    VINABT          ;ABORT OR CONTINUE
025520  001132                          BNE     9$              ;BR IF ABORT WANTED
025522  005061  000000G                 CLR     RT$NAM(R1)      ;Mark run-time as not-available
025526  005061  000020G                 CLR     RT$NAM+2(R1)
025532  000520                          BR      7$              ;GO LOAD NEXT RUN-TIME SYSTEM
                                ;
                                ;  Set up information about position of run-time in physical memory
                                ;
025534  116102  000000G         8$:     MOVB    RT$SKP(R1),R2   ;GET # BLOCKS TO SKIP AT FRONT OF RUN-TIME
025540  042702  177400                  BIC     #^C377,R2       ;CLEAR SIGN EXTENSION
025544  160200                          SUB     R2,R0           ;GET # BLOCKS TO READ (LOOKUP SET R0 W  SIZE)
025546  010561  000000G                 MOV     R5,RT$TOP(R1)   ;SET 64-BYTE BLOCK # ABOVE TOP OF RUN-TIME
025552  010003                          MOV     R0,R3           ;GET # 512-BYTE BLOCKS IN RUN-TIME
025554  072027  000003                  ASH     #3,R0           ;CONVERT TO # 64-BYTE BLOCKS
025560  160005                          SUB     R0,R5           ;CALCULATE BASE 64-BYTE BLOCK # OF RUN-TIME
025562  020527  001600                  CMP     R5,#1600        ;ARE WE ABOUT TO RUN OVER RT-11?
025566  103530                          BLO     11$             ;BR IF YES
025570  010561  000000G                 MOV     R5,RT$BAS(R1)   ;SET BASE 64-BYTE BLOCK # OF RUN-TIME
                                ;
                                ;  Read run-time system into memory and position in high-memory
                                ;
025574  010546                          MOV     R5,-(SP)        ;Save address of bottom of run-time
025576  013704  000152'          4$:     MOV     WRKBUF,R4       ;Point to work buffer
025602                                  .READW  #AREA,#1,R4,#256.,R2 ;READ A BLOCK OF RUN-TIME FILE
                                ;  Use memory management to access high-memory area.
025636  012701  000000G                 MOV     #VPAR6,R1       ;GET VIRTUAL ADDRESS OF PAR6 ADDRESS REGION
025642  010537  000000G                 MOV     R5,@#UPAR6      ;SET USER-MODE PAR6 MAP OFFSET VALUE
```

```
     58 025646  012737  077406  000000G          MOV     #077406,@#UPDR6 ;SET PDR TO ALLOW FULL ACCESS TO PAGE
     59 025654  052737  000000G 000000G          BIS     #UPMODE,@#PSW   ;SET PREVIOUS-MODE = USER FOR MTPD ACCESS
     60 025662  012700  000400                   MOV     #256.,R0        ;GET # WORDS TO MOVE
     61 025666                                   DISABL                  ;** Disable interrupts **
     62 025674  052737  000000G 000000G          BIS     #MMENBL,@#SROMMR;enable memory management
     63 025702  105737  000000G                  TSTB    MEM256          ;DOES THIS MACHINE HAVE AT LEAST 256KB?
     64 025706  001403                           BEQ     3$              ;BR IF NOT
     65 025710  052737  000000G 000000G          BIS     #EMMAP,@#SR3MMR ;enable extended memory addressing
     66 025716  012446                  3$:     MOV     (R4)+,-(SP)     ;TRANSFER DATA FROM BUFFER TO HIGH MEMORY
     67 025720  106621                           MTPD    (R1)+
     68 025722  077003                           SOB     R0,3$
     69 025724  105737  000000G                  TSTB    MEM256          ;DOES THIS MACHINE HAVE AT LEAST 256KB?
     70 025730  001403                           BEQ     31$             ;BR IF NOT
     71 025732  042737  000000G 000000G          BIC     #EMMAP,@#SR3MMR ;DISABLE EXTENDED MEMORY MANAGEMENT
     72 025740  042737  000000G 000000G 31$:     BIC     #MMENBL,@#SROMMR;DISABLE MEMORY MANAGEMENT
     73 025746                                   ENABL                   ;** Enable interrupts **
     74 025754  062705  000010                   ADD     #10,R5          ;ADVANCE 64-BYTE BLOCK # BY 512-BYTES
     75 025760  005202                           INC     R2              ;INC FILE BLOCK #
     76 025762  077373                           SOB     R3,4$           ;READ AND COPY REST OF FILE
     77                                 ;
     78                                 ;  Finished loading the run-time system.
     79                                 ;
     80 025764  012605                           MOV     (SP)+,R5
     81 025766                                   .CLOSE  #1
     82                                 ;
     83                                 ;  Finished
     84                                 ;
     85 025774  012604                  7$:     MOV     (SP)+,R4
     86 025776  012603                           MOV     (SP)+,R3
     87 026000  012602                           MOV     (SP)+,R2
     88 026002  012601                           MOV     (SP)+,R1
     89 026004  000207                           RETURN
     90                                 ;
     91                                 ;  Error -- Cannot find run-time system file
     92                                 ;
     93 026006                          9$:     .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
     94 026014                                  .PRINT  #COSRT          ;PRINT ERROR MESSAGE
     95 026022  012702  000004                   MOV     #4,R2           ;PRINT 4 RAD50 VALUES
     96 026026  012100                  10$:    MOV     (R1)+,R0        ;GET PART OF NAME
     97 026030  004737  030012'                  CALL    PRTR50          ;PRINT RAD50 VALUE
     98 026034  077204                           SOB     R2,10$
     99 026036                                  .PRINT  #CRLF           ;END LINE
    100 026044  000137  004216'                  JMP     INISTP          ;ABORT INITIALIZATION
    101                                 ;
    102                                 ;  Error -- Insufficient memory space to load run-time systems
    103                                 ;
    104 026050                          11$:    .PRINT  #TSXHD          ;PRINT HEADING
    105 026056                                  .PRINT  #SRTOVF         ;PRINT ERROR MESSAGE
    106 026064  000137  004216'                  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.
                                ;
026070  010246                  CSHBUF: MOV     R2,-(SP)
026072  010346                          MOV     R3,-(SP)
                                ;
                                ;  See if data caching is wanted
                                ;
026074  013737  000000G 000000G         MOV     CSHALC,VCSHNB   ;Set # blocks in use = # blocks allocated
026102  013702  000000G                 MOV     CSHALC,R2       ;Did used request data caching?
026106  001464                          BEQ     9$              ;Br if not
                                ;
                                ;  Calculate number of 64-byte blocks needed for each cache control table
                                ;
026110  062702  000037                  ADD     #31.,R2         ;Bound up to 32 word block
026114  072227  177773                  ASH     #-5.,R2         ;Convert to # 64-byte blocks
                                ;
                                ;  Compute total space that will be used by all cache data
                                ;
026120  013703  000000G                 MOV     CSHALC,R3       ;Get # blocks in cache
026124  072327  000003                  ASH     #3,R3           ;Get # 64-byte blks used by cache data buffers
026130  012700  000010                  MOV     #8.,R0          ;Get number of cache control tables
026134  060203                  1$:     ADD     R2,R3           ;Accumulate total space needed
026136  077002                          SOB     R0,1$
026140  010337  000000G                 MOV     R3,CSHSIZ       ;Save total space used by cache data
                                ;
                                ;  See if there is enough memory space available for the specified cache
                                ;
026144  010500                          MOV     R5,R0           ;Get top of memory address
026146  160400                          SUB     R4,R0           ;Compute # free 64-byte blocks
026150  020300                          CMP     R3,R0           ;Is there enough total space?
026152  103045                          BHIS    10$             ;Br if not
                                ;
                                ;  Allocate space for cache data buffers
                                ;
026154  013700  000000G                 MOV     CSHALC,R0       ;Get # blocks in data cache
026160  072027  000003                  ASH     #3,R0           ;Get # 64-byte blocks needed for allocation
026164  160005                          SUB     R0,R5           ;Allocate space for cache data buffers
026166  010537  000000G                 MOV     R5,CSHBFP       ;Save pointer to base of buffer area
                                ;
                                ;  Allocate space for each control table
                                ;
026172  160205                          SUB     R2,R5           ;Allocate space for table
026174  010537  000000G                 MOV     R5,CA$BLK       ;Block number associated with entry
026200  160205                          SUB     R2,R5           ;Allocate space for table
026202  010537  000000G                 MOV     R5,CA$DVU       ;Device and unit number
026206  160205                          SUB     R2,R5           ;Allocate space for table
026210  010537  000000G                 MOV     R5,CA$WCT       ;Number of words
026214  160205                          SUB     R2,R5           ;Allocate space for table
```

```
     58 026216  010537  000000@                    MOV     R5,CA$UFL       ;LRU chain forward link
     59 026222  160205                             SUB     R2,R5           ;Allocate space for table
     60 026224  010537  000000@                    MOV     R5,CA$UBL       ;LRU chain backward link
     61 026230  160205                             SUB     R2,R5           ;Allocate space for table
     62 026232  010537  000000@                    MOV     R5,CA$HFL       ;Hash chain forward link
     63 026236  160205                             SUB     R2,R5           ;Allocate space for table
     64 026240  010537  000000@                    MOV     R5,CA$HBL       ;Hash chain backward link
     65 026244  160205                             SUB     R2,R5           ;Allocate space for table
     66 026246  010537  000000@                    MOV     R5,CA$HSH       ;Hash chain list heads
     67 026252  020527  001600                     CMP     R5,#1600        ;Did we run over RT-11?
     68 026256  101403                             BLOS    10$             ;Br if yes
     69                                 ;
     70                                 ;  Finished
     71                                 ;
     72 026260  012603                  9$:        MOV     (SP)+,R3
     73 026262  012602                             MOV     (SP)+,R2
     74 026264  000207                             RETURN
     75                                 ;
     76                                 ;  Insufficent memory space available for cache data
     77                                 ;
     78 026266                          10$:       .PRINT  #TSXHD          ;Print heading
     79 026274                                     .PRINT  #CSHOVF         ;Overlow message
     80 026302  000137  004216'                    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.
                                ;
026306  010146                  GETODT: MOV     R1,-(SP)
026310  010246                          MOV     R2,-(SP)
026312  010346                          MOV     R3,-(SP)
026314  010446                          MOV     R4,-(SP)
                                ;
                                ;  Try to lookup ODT rel file.
                                ;
026316                                  .LOOKUP #AREA,#1,#ODTBLK ;LOOKUP ODT REL FILE
026336  103010                        BCC     1$              ;BR IF FOUND IT
026340                                  .PRINT  #TSXHD          ;CAN'T FIND ODT
026346                                  .PRINT  #NOODT
026354  000137  004216'                JMP     INISTP          ;ABORT INITIALIZATION
                                ;
                                ;  Read first block of ODT file and determine size of ODT.
                                ;
026360  062705  000310          1$:     ADD     #200.,R5        ;RESERVE SPACE FOR ODT STACK
026364  010500                          MOV     R5,R0           ;CHECK MEMORY ADDRESS FOR OVERFLOW
026366  062700  001000                  ADD     #512.,R0
026372  004737  027622'                CALL    CHKMEM
026376                                  .READW  #AREA,#1,R5,#256.,#0
026432  103002                        BCC     2$              ;Br if no read error
026434  000137  027016'                JMP     ODTRDX          ;Read error
026440  016502  000052          2$:     MOV     RSZ(R5),R2      ;GET SIZE OF ODT
026444  010203                          MOV     R2,R3
026446  060503                          ADD     R5,R3           ;GET ADDRESS ABOVE TOP OF ODT
026450  010300                          MOV     R3,R0           ;CHECK MEMORY ADDRESS FOR OVERFLOW
026452  004737  027622'                CALL    CHKMEM
026456  000241                          CLC
026460  006002                          ROR     R2              ;GET # WORDS IN ODT
                                ; Get starting address of ODT
026462  016500  000040                  MOV     STA(R5),R0      ;GET OFFSET TO START ADDRESS
026466  162700  001000                  SUB     #ODTBAS,R0      ;CALCULATE ABSOLUTE STARTING ADDRESS
026472  060500                          ADD     R5,R0
026474  010037  000300'                MOV     R0,ODTSTA       ;THIS IS REAL STARTING ADDRESS
026500  016501  000062                  MOV     RBD(R5),R1      ;GET # OF BLOCK WITH RELOCATION INFO
                                ;
                                ;  Read in ODT rel file image.
                                ;
026504                                  .READW  #AREA,#1,R5,R2,#1
026540  103526                        BCS     ODTRDX          ;BR IF READ ERROR
                                ;
                                ;  Relocate addresses in ODT.
                                ;  R5 = Address of base of ODT;  R3 = Address above top of ODT.
                                ;  R1 = Block number in rel file of start of relocation info.
                                ;
```

```
     58 026542  010337  000036'          RELFIL: MOV     R3,ODTTOP       ;SAVE ADDRESS ABOVE TOP OF ODT
     59 026546  010337  000274'                  MOV     R3,RLBF
     60                                  .IF     NE,PROCID               ;Only if PRO protection code is included
     61                                          TSTB    PROFLG          ;Are we running on a Pro?
     62                                          BNE     1$              ;Br if yes
     63                                  .ENDC   ;NE,PROCID
     64 026552  013737  000152' 000274'         MOV     WRKBUF,RLBF     ;READ RELOCATION INFO HERE
     65 026560  013737  000274' 000276' 1$:     MOV     RLBF,RLBFND
     66 026566  062737  002000  000276'         ADD     #1024.,RLBFND   ;GET ADDRESS OF END OF BUFFER AREA
     67 026574  010504                          MOV     R5,R4           ;GET BASE ADDRESS OF ODT
     68 026576  162704  001000                  SUB     #ODTBAS,R4      ;SUBTRACT LINK BASE ADDRESS
     69                                  ;  Read in relocation address list.
     70 026602                          4$:     .READW  #AREA,#1,RLBF,#512.,R1
     71 026640  103003                          BCC     7$              ;BR IF NO READ ERROR
     72 026642  105737  000052                  TSTB    @#52            ;END OF FILE IS OK
     73 026646  001063                          BNE     ODTRDX          ;BR IF READ ERROR
     74                                  ;  Relocate some addresses in ODT.
     75 026650  013702  000274'         7$:     MOV     RLBF,R2         ;POINT TO RELOCATION INFO
     76 026654  012203                  3$:     MOV     (R2)+,R3        ;GET ADDRESS OF LOCATION TO RELOCATE
     77 026656  020327  177776                  CMP     R3,#-2          ;TIME TO STOP?
     78 026662  001416                          BEQ     9$              ;BR IF FINISHED
     79 026664  012200                          MOV     (R2)+,R0        ;GET VALUE TO RELOCATE
     80 026666  006303                          ASL     R3              ;CVT TO BYTE ADDRESS
     81 026670  103002                          BCC     5$              ;BR IF ADDITIVE RELOCATION
     82 026672  160400                          SUB     R4,R0           ;RELOCATE THE ADDRESS
     83 026674  000401                          BR      6$
     84 026676  060400                  5$:     ADD     R4,R0           ;RELOCATE THE ADDRESS
     85 026700  060503                  6$:     ADD     R5,R3           ;GET LOCATION WHERE WORD GOES
     86 026702  010013                          MOV     R0,@R3          ;STORE RELOCATED ADDRESS
     87 026704  020237  000276'                 CMP     R2,RLBFND       ;TIME TO READ NEXT BUFFER FULL?
     88 026710  103761                          BLO     3$              ;BR IF NOT
     89 026712  062701  000002                  ADD     #2,R1           ;ADVANCE BLOCK #
     90 026716  000731                          BR      4$              ;GO READ NEXT BUFFER FULL
     91                                  ;
     92                                  ;  Finished relocation.
     93                                  ;  Close ODT rel file.
     94                                  ;
     95 026720                          9$:     .CLOSE  #1
     96                                  ;
     97                                  ;  Direct interrupts to 60 and 64 to an RTI instruction
     98                                  ;
     99                                  ;       MOV     #DORTI,@#60     ;Catch interrupt 60
    100                                  ;       MOV     #DORTI,@#64     ;Catch interrupt 64
    101                                  ;
    102                                  ;  Load registers with the following values for initial entry to ODT:
    103                                  ;   R0 = Base of TSINIT
    104                                  ;   R1 = Important breakpoint (^R) in TSX
    105                                  ;   R2 = Base of TSGEN
    106                                  ;   R3 = Base of TSEXEC
    107                                  ;   R4 = Base of TSEMT
    108                                  ;   R5 = Return address to start execution
    109                                  ;   0(SP) = Address of mapsys routine
    110                                  ;   2(SP) = Address of  sysmap cell
    111                                  ;
    112 026726  012700  000000'                 MOV     #TSINIT,R0
    113 026732  012701  000000G                 MOV     #BRKPT,R1
    114 026736  012702  000000G                 MOV     #TSGEN,R2
```

```
     115 026742  012703  0000000                MOV     #TSEXEC,R3
     116 026746  012704  0000000                MOV     #TSEMT,R4
     117 026752  012746  0000000                MOV     #SYSMAP,-(SP)   ;PASS ADDRESS OF SYSMAP CELL TO ODT
     118 026756  012746  0000000                MOV     #MAPSYS,-(SP)   ;PASS ADDRESS OF MAPSYS ROUTINE
     119 026762  012705  026772'                MOV     #10$,R5         ;ADDRESS FOR ODT TO RETURN TO
     120                                ;
     121                                ;  Enter ODT
     122                                ;
     123 026766  000177  151306                 JMP     @ODTSTA         ;JUMP TO START OF ODT
     124                                ;
     125                                ;  Return from ODT.
     126                                ;  Continue initialization of TSX.
     127                                ;
     128 026772  013737  000014  0000000 10$:   MOV     @#14,ODTTRP     ;SAVE ODT BREAKPOINT ENTRY ADDRESS
     129 027000  013705  000036'                MOV     ODTTOP,R5       ;ADDRESS ABOVE TOP OF ODT
     130 027004  012604                         MOV     (SP)+,R4
     131 027006  012603                         MOV     (SP)+,R3
     132 027010  012602                         MOV     (SP)+,R2
     133 027012  012601                         MOV     (SP)+,R1
     134 027014  000207                         RETURN
     135                                ;
     136                                ;  Error while reading ODT rel file.
     137                                ;
     138 027016                         ODTRDX: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     139 027024                                 .PRINT  #ODTRDM
     140 027032  000137  004216'                JMP     INISTP          ;ABORT INITIALIZATION
     141                                ;
     142                                ;  RTI instruction to disable interrupts
     143                                ;
     144 027036  000002                 DORTI:  RTI
     145                                        .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.
                                ;
027040  010146                  OPNCHN: MOV     R1,-(SP)
027042  010346                          MOV     R3,-(SP)
027044  010003                          MOV     R0,R3           ;Carry channel block address in R3
                                ;
                                ;  Initialize the channel block
                                ;
027046  010301                          MOV     R3,R1           ;Point to the channel block
027050  012700  000000C                 MOV     #<CHNSIZ/2>,R0  ;Get # words to zero
027054  005021                  2$:     CLR     (R1)+           ;Zero the channel block
027056  077002                          SOB     R0,2$
027060  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 #
                                ;
027066  010200                          MOV     R2,R0           ;Get the full device name
027070  004737  012632'                 CALL    CVTDVU          ;Convert to dev # and unit #
027074  103411                          BCS     9$              ;Br if we don't recognize the device name
027076  010001                          MOV     R0,R1           ;Get index # and unit #
027100  000301                          SWAB    R1              ;Get unit # to low byte
027102  110163  000000G                 MOVB    R1,C.DEVQ(R3)   ;Set unit # in channel block
027106  042700  000000C                 BIC     #^C<CS$NMX>,R0  ;Clear all but device index number in R0
027112  050063  000000G                 BIS     R0,C.CSW(R3)    ;Store device index # into CSW
                                ;
                                ;  Success
                                ;
027116  000241                          CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
027120  012603                  9$:     MOV     (SP)+,R3
027122  012601                          MOV     (SP)+,R1
027124  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.
                                ;
027126  010146                  SETCHN: MOV     R1,-(SP)
027130  010246                          MOV     R2,-(SP)
027132  010346                          MOV     R3,-(SP)
027134  010001                          MOV     R0,R1           ;GET ADDRESS OF TSX CHANNEL BLOCK
                                ;
                                ;  Do .SAVESTATUS to store channel information into TSX channel block.
                                ;
027136                                  .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.
                                ;
027154  011103                          MOV     (R1),R3         ;GET CSW FOR CHANNEL
027156  042703  177701                  BIC     #^C76,R3        ;GET RT-11 DEVICE INDEX NUMBER
027162                                  .GVAL   #AREA,#404      ;GET RT-11 OFFSET TO PNAME TABLE
027202  060003                          ADD     R0,R3           ;GET ADDRESS OF NAME OF DEVICE IN PNAME TABLE
027204                                  .GVAL   #AREA,R3        ;GET NAME OF DEVICE FROM RT-11
027222  013703  000000G                 MOV     NUMDEV,R3       ;GET INDEX # FOR LAST TSX DEVICE
027226  020063  000000G          1$:    CMP     R0,PNAME(R3)    ;LOOK FOR DEVICE IN OUR TABLES
027232  001404                          BEQ     2$              ;BR IF FOUND
027234  162703  000002                  SUB     #2,R3           ;CHECK NEXT ENTRY
027240  002372                          BGE     1$              ;BR IF MORE TO CHECK
027242  000407                          BR      MTSXDV          ;VERY STRANGE THAT WE DIDN'T FIND IT
027244  042711  000076          2$:    BIC     #76,(R1)        ;CLEAR OUT RT-11 DEVICE #
027250  050311                          BIS     R3,(R1)         ;STORE TSX DEVICE #
                                ;
                                ;  Finished
                                ;
027252  012603                          MOV     (SP)+,R3
027254  012602                          MOV     (SP)+,R2
027256  012601                          MOV     (SP)+,R1
027260  000207                          RETURN
                                ;
                                ;  Error: Could not locate Rt-11 device number in TSX device table.
                                ;
027262                          MTSXDV: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
027270                                  .PRINT  #REQMIS         ;Missing a required device
027276  010200                          MOV     R2,R0           ;GET RAD50 DEVICE NAME
027300  004737  030012'                  CALL    PRTR50          ;DISPLAY DEVICE NAME
027304                                  .PRINT  #CRLF
```

```
     58 027312  000137  004216'                    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
                                ;
027316  010146                  SETSY:  MOV     R1,-(SP)
027320  010246                          MOV     R2,-(SP)
                                ;
                                ;  Set up system device unit number
                                ;
027322                                  .GVAL   #AREA,#274      ;Get system unit # from RT-11 (high byte)
027342  010037  000000G                 MOV     R0,SYUNIT       ;Set system unit number
                                ;
                                ;  Set up system device index number
                                ;
027346                                  .GVAL   #AREA,#364      ;Get RT-11 system device index number
027366  010002                          MOV     R0,R2           ;Save device index number
027370                                  .GVAL   #AREA,#404      ;Get offset within RMON of PNAME table
027410  060002                          ADD     R0,R2           ;Get offset to name of SY device
027412                                  .GVAL   #AREA,R2        ;Get name of RT-11 system device
027430  013701  000000G                 MOV     NUMDEV,R1       ;Get index to last TSX-Plus device entry
027434  020061  000000G          1$:    CMP     R0,PNAME(R1)    ;Search for device in TSX tables
027440  001405                          BEQ     2$              ;Br if found it
027442  162701  000002                  SUB     #2,R1           ;Keep looking if more
027446  002372                          BGE     1$
027450  010002                          MOV     R0,R2           ;Save name of system device
027452  000703                          BR      MTSXDV          ;Missing device error
027454  010137  000000G          2$:    MOV     R1,SYINDX       ;Store index # of TSX-Plus system device
                                ;
                                ;  Set up RAD50 name of SY disk
                                ;
027460  113702  000001G                 MOVB    SYUNIT+1,R2     ;GET SYSTEM UNIT NUMBER
027464  062702  000036                  ADD     #36,R2          ;PUT IN "0" AS 3'RD CHARACTER OF NAME
027470  066102  000000G                 ADD     PNAME(R1),R2    ;ADD DEVICE NAME
027474  010237  000000G                 MOV     R2,SYNAME       ;THIS IS THE FULL SY DISK NAME
                                ;
                                ;  Finished
                                ;
027500  012602                          MOV     (SP)+,R2
027502  012601                          MOV     (SP)+,R1
027504  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.
                                ;
027506  010046                  RTFTCH: MOV     R0,-(SP)
027510  010246                          MOV     R2,-(SP)
027512  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set the name of the device being fetched
                                ;
027514  010037  000130'                 MOV     R0,FETDEV       ;Set name of device whose handler to fetch
                                ;
                                ;  Do a .DSTAT to get information about the handler
                                ;
027520                                  .DSTAT  #DSTBLK,#FETDEV ;Get information about the device handler
027532  103425                          BCS     9$              ;Br if device not recognized
                                ;
                                ;  Determine if the handler is currently resident
                                ;
027534  005737  000116'                 TST     DSTBLK+4        ;Is the handler resident now?
027540  001021                          BNE     8$              ;Br if yes
                                ;
                                ;  The handler is not currently resident.
                                ;  See if it will fit in WRKBUF.
                                ;
027542  013702  000152'                 MOV     WRKBUF,R2       ;Set address where handler will be loaded
027546  013700  000114'                 MOV     DSTBLK+2,R0     ;Get the size of the handler
027552  020037  000154'                 CMP     R0,WRKSIZ       ;Will handler fit in WRKBUF?
027556  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.
                                ;
027560  060500                          ADD     R5,R0           ;Get address above top of area needed
027562  020037  000132'                 CMP     R0,TOPMEM       ;Is there room for handler?
027566  101013                          BHI     10$             ;Br if not
027570  010502                          MOV     R5,R2           ;Set address where handler is to be loaded
                                ;
                                ;  Fetch the handler
                                ;
027572                          1$:     .FETCH  R2,#FETDEV      ;Try to fetch the handler
027602  103401                          BCS     9$              ;Br if error on fetch
                                ;
                                ;  We successfully fetched the handler
                                ;
```

```
     58 027604  000241                  8$:     CLC                     ;Signal success on return
     59                                 ;
     60                                 ;  Finished
     61                                 ;
     62 027606  012605                  9$:     MOV     (SP)+,R5
     63 027610  012602                          MOV     (SP)+,R2
     64 027612  012600                          MOV     (SP)+,R0
     65 027614  000207                          RETURN
     66                                 ;
     67                                 ;  Insufficient memory available to load the handler
     68                                 ;
     69 027616  004737  027642'         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.
                                ;
027622  020037  000302'         CHKMEM: CMP     R0,MEMLIM       ;IS THE ADDRESS OK?
027626  103402                          BLO     1$              ;BR IF OK
027630  004737  027642'                 CALL    SIZERR          ;Generated system is too big -- abort
027634  004737  027662'         1$:     CALL    CCATST          ;CHECK FOR ^C ABORT REQUEST
027640  000207                          RETURN

                                ;-------------------------------------------------------------------------
                                ;  Generated system is too big.  Abort the initialization.
                                ;
027642                          SIZERR: .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
027650                                  .PRINT  #TOOBIG         ;PRINT ERROR MESSAGE
027656  000137  004216'                 JMP     INISTP          ;ABORT INITIALIZATION

                                ;-------------------------------------------------------------------------
                                ;  Check for control-C and abort initialization if requested.
                                ;
027662  005737  000040'         CCATST: TST     CCAFLG          ;DID USER REQUEST ^C ABORT?
027666  001402                          BEQ     1$              ;BRANCH IF NOT
027670  000137  004216'                 JMP     INISTP          ;ELSE ABORT INITIALIZATION
027674  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.
                                ;
027676  010146                 PRTOCT: MOV     R1,-(SP)
027700  010246                         MOV     R2,-(SP)
027702  010001                         MOV     R0,R1           ;GET VALUE TO PRINT
027704  012702  000006                 MOV     #6,R2           ;PRINT 6 DIGITS
027710  005000                         CLR     R0
027712  073027  000001                 ASHC    #1,R0           ;GET 1ST OCTAL DIGIT (1 BIT)
027716  000403                         BR      2$
027720  005000                 1$:     CLR     R0              ;INITIALIZE FOR SHIFT
027722  073027  000003                 ASHC    #3,R0           ;SHIFT AN OCTAL DIGIT INTO R0
027726  062700  000060         2$:     ADD     #'0,R0          ;CONVERT TO ASCII CHARACTER
027732                                 .TTYOUT                 ;PRINT THE CHARACTER
027736  077210                         SOB     R2,1$           ;LOOP AND PRINT MORE DIGITS
027740  012602                         MOV     (SP)+,R2
027742  012601                         MOV     (SP)+,R1
027744  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
                                ;
027746  010146                  PRTDEC: MOV     R1,-(SP)
027750  005046                          CLR     -(SP)           ;NULL ON STACK TO STOP US
                                ;
                                ;  Convert value to ascii digit string and stack the digits.
                                ;
027752  010001                          MOV     R0,R1           ;GET VALUE TO BE CONVERTED
027754  005000                  1$:     CLR     R0              ;SET HIGH-ORDER PART OF VALUE TO 0
027756  071027  000012                  DIV     #10.,R0         ;DIVIDE R0-R1 BY 10.
027762  062701  000060                  ADD     #'0,R1          ;CONVERT REMAINDER TO ASCII DIGIT
027766  010146                          MOV     R1,-(SP)        ;AND STACK THE DIGIT
027770  010001                          MOV     R0,R1           ;GET QUOTIENT
027772  001370                          BNE     1$              ;BR IF MORE DIGITS TO CONVERT
                                ;
                                ;  Finished conversion.  Print result.
                                ;
027774  012600                  2$:     MOV     (SP)+,R0        ;GET A DIGIT FROM THE STACK
027776  001403                          BEQ     3$              ;BR IF REACHED END
030000                                  .TTYOUT                 ;PRINT THE DIGIT
030004  000773                          BR      2$              ;PRINT MORE
                                ;
                                ;  Finished
                                ;
030006  012601                  3$:     MOV     (SP)+,R1
030010  000207                          RETURN
```

```
                                        .SBTTL  PRTR50 -- Print Rad-50 value
                                ;-------------------------------------------------------------------------
                                ;  PRTR50 is called to print a Rad-50 value.
                                ;
                                ;  Inputs:
                                ;   R0 = value to be printed.
                                ;
030012  010146                  PRTR50: MOV     R1,-(SP)
030014  010246                          MOV     R2,-(SP)
                                ;
                                ;  Convert value to ascii string and stack the characters.
                                ;
030016  012702  000003                  MOV     #3,R2           ;GET # CHARS TO CVT
030022  010001                          MOV     R0,R1           ;GET VALUE TO BE CONVERTED
030024  005000                  1$:     CLR     R0              ;CLEAR HIGH-ORDER VALUE
030026  071027  000050                  DIV     #50,R0          ;DIVIDE R0-R1 BY 50
030032  116101  003464'                 MOVB    R50CHR(R1),R1   ;CONVERT REMAINDER TO ASCII CHARACTER
030036  010146                          MOV     R1,-(SP)        ;STACK THE CHARACTER
030040  010001                          MOV     R0,R1           ;GET QUOTIENT
030042  077210                          SOB     R2,1$           ;BR IF MORE CHARS TO CONVERT
                                ;
                                ;  Finished conversion.  Print the result.
                                ;
030044  012702  000003                  MOV     #3,R2           ;GET # CHARS TO PRINT
030050  012600                  2$:     MOV     (SP)+,R0        ;GET NEXT CHARACTER
030052                                  .TTYOUT                 ;PRINT THE CHARACTER
030056  077204                          SOB     R2,2$           ;LOOP IF MORE CHARS TO PRINT
                                ;
                                ;  Finished
                                ;
030060  012602                          MOV     (SP)+,R2
030062  012601                          MOV     (SP)+,R1
030064  000207                          RETURN
                                ;-------------------------------------------------------------------------
                                ;  Define top of TSINIT
                                ;
030066                          INITOP:
                                ;
```

```
                        .IF     NE,PROCID       ;Only assemble for protected Pro 350 version
                        ;
                        ;  The following startup code is only included for the Pro version.
                        ;  It is loaded here and executed very early during initialization
                        ;  and subsequently overwritten by I/O buffers.
                        ;
                                .SBTTL  INSCHK -- Installation validation subroutines for Pro-350
                                .MCALL  .PRINT
                        ;
                        ;  Reserve an arg block area for encryption calls
                        ;
                        EDARGB: .WORD   -32.                    ;# OF BYTES TO BE DECRYPTED (EDMTH3)
                        EDADDR: .WORD   DSKBUF                  ;POINTER TO BUFFER TO BE DECRYPTED
                        ;
                        ;  Recover license number and decrypt disk image of Pro ID to intermed. state
                        ;
                        INSCHK: MOV     R1,-(SP)                ;SAVE REGISTERS
                                MOV     R2,-(SP)
                                MOV     R3,-(SP)
                                MOV     R4,-(SP)
                                MOV     R5,-(SP)
                                MOV     (PC)+,R0                ;DECRYPT LICENSE NUMBER
                                .RAD50  /SCB/                   ;WITH THIS CODE
                                XOR     R0,LICNUM               ;BY XORING IT
                                MOV     LICNUM,TSXSIT           ;MOVE LICENCE NUMBER TO TSGEN CELL
                                MOV     #EDARGB,R0              ;POINT TO ENC/DEC ARG BLOCK (PRESET)
                                CALL    EDMTH3                  ;DECRYPT TO INTERMED STATE
                        ;
                        ;  Copy Pro ID ROM low bytes into memory, and encrypt 1 step
                        ;
                        IDADDR  = 173600        ;ADDRESS OF START OF PRO 350 ID ROM
                                MOV     #IDADDR,R1              ;GET POINTER TO PRO ID ROM
                                MOV     #ROMBUF,R2              ;POINTER TO COPY OF HARDWARE ID
                                MOV     R2,EDADDR               ;SAVE ADDRESS FOR ENCRYPTION
                                NEG     EDARGB                  ;MAKE +32. FOR ENCRYPTION
                                MOV     EDARGB,R0               ;ALSO USE AS LOOP COUNTER
                        3$:     MOVB    (R1)+,(R2)+             ;GET NEXT LOW BYTE
                                INC     R1                      ;SKIP ID ROM HIGH BYTES
                                SOB     R0,3$                   ;REPEAT THROUGH 32 BYTE ROM
                                MOV     #EDARGB,R0              ;POINT TO ENCRYPTION ARG BLOCK
                                CALL    EDMTH2                  ;PERFORM METHOD 2 ENCRYPTION
                        ;
                        ;  Have intermediate state of both hardware and disk copies of Pro ID
                        ;  in memory. Verify them against each other and correct memory image
                        ;  of SCHED at the same time.
                        ;
                                MOV     #ROMBUF,R1              ;POINT TO HARDWARE COPY OF ID
                                MOV     #DSKBUF,R2              ;POINT TO DISK COPY OF ID
                                MOV     #SCHED,R4               ;POINT TO CODE TO BE CORRECTED
                                MOV     EDARGB,R3               ;INIT LOOP COUNTER
                                CALL    GETLIC                  ;USE LIC # AS SEED FOR EDPRNW, IN R0
                        4$:     CMPB    (R1)+,(R2)+             ;VERIFY ID'S ARE THE SAME
                                BNE     5$                      ;ABORT IF NO MATCH ON ANY BYTE
                                CALL    EDPRNW                  ;RANDOMIZE R0 FOR XOR (LIC# INIT SEED)
                                MOV     @R4,R5                  ;GET ENCRYPTED CODE
                                XOR     R0,R5                   ;RESTORE FUNCTIONAL CODE
                                MOV     R5,(R4)+                ;PUT DECRYPTED CODE BACK IN MEMORY
```

```
                        SOB     R3,4$                   ;REPEAT THROUGH ID TESTS
                        MOV     (SP)+,R5                ;RESTORE REGISTERS
                        MOV     (SP)+,R4
                        MOV     (SP)+,R3
                        MOV     (SP)+,R2
                        MOV     (SP)+,R1
                        RETURN                          ;ID CHECKS AND CODE DECRYPTED
                ;
                5$:     .PRINT  #TSXHD                  ;?TSX-F
                        .PRINT  #NOTLIC                 ;NOT LICENSED FOR THIS MACHINE
                        JMP     INISTP                  ;ID'S DON'T MATCH, ABORT INIT.
                ;
                        .NLIST  BEX
                NOTLIC: .ASCIZ  /This copy of TSX-Plus not licensed for use on this machine./
                        .LIST   BEX
                        .EVEN
                ;
                ;  Subroutine to recover incremental license number.  Assume it has been
                ;  decrypted already by XORing with .RAD50 /SCB/.
                ;
                GETLIC: MOV     LICNUM,R0               ;RETRIEVE DECRYPTED LIC # INTO R0
                        RETURN
                ;
                ;  Reserve room for both disk and hardware copies of the Pro ID number
                ;  and for the incremental license number
                ;
                DSKBUF: .BLKB   32.                     ;DISK IMAGE OF PRO ID
                LICNUM: .WORD   0                       ;INCREMENTAL LICENSE NUMBER
                ROMBUF: .BLKB   32.                     ;COPY OF ROM ID LOW BYTES
```

```
                        .SBTTL  EDEXPL -- Comments on encryption methods
                ;
                ;  Encryption and decryption methods used here depend heavily on
                ;  pseudo-random numbers generated by the linear congrutential method.
                ;  See Hull and Dobell, SIAM Review, 4, 230, 1962.
                ;
                ;  For the linear congruence relation:
                ;
                ;    X(I) == ( A * X(I-1) + C ) MOD M
                ;
                ;    X(I) is in the range 0 to M-1
                ;
                ;  The sequence has full period M, provided that:
                ;    1) C is relatively prime to M
                ;    2) If p is a prime factor of M, A MOD p == 1
                ;    3) If 4 is a factor of M, A MOD 4 == 1
                ;
                ;  In the special case where M is a power of 2, these rules simplify to
                ;    1) C must be odd
                ;    2) A MOD 4 == 1
                ;
                        .SBTTL  EDMTH2 -- Encryption method 2 (XOR with PRN high bytes)
                ;
                ;  Using the license number as the initial seed, mask out the low 3 bits,
                ;  add 1 and call the PRN generator this many times to form the seed,
                ;  XOR each byte in the input buffer with the high byte of the next PRN
                ;  and replace the result in the input buffer. Decryption is accomplished
                ;  by a second application of the same process.
                ;
                ;  Inputs:
                ;       R0      Points to an arg block of the form:
                ;               R0 --->  buff_siz       ;word holding byte length of buffer
                ;                        buff_addr      ;address of buffer to be encrypted
                ;  Outputs:
                ;       R0      Randomized
                ;       input buffer encrypted
                ;
                EDMTH2:
                        MOV     R1,-(SP)        ;Save registers
                        MOV     R2,-(SP)
                        MOV     R3,-(SP)
                ;
                ;  Fetch byte count, buffer pointer and initialize PRN seed
                ;
                        MOV     (R0)+,R3        ;Fetch byte count of input buffer
                        MOV     (R0),R1         ;Fetch pointer to input buffer
                        CALL    GETLIC          ;Use license number as initial PRN seed
                        MOV     R0,R2           ;Copy license number to form repeat count
                        BIC     #^C7,R2         ;No more than 8 repeats
                        INC     R2              ;Make sure there is at least one
                2$:     CALL    EDPRNW          ;Get a new PRN
                        SOB     R2,2$           ;Advance the seed between 1 and 8 times
                ;
                ;  Now sweep the buffer, XORing each byte with the high PRN byte
                ;
                1$:     CALL    EDPRNW          ;With seed in R0, get next random number
                        MOVB    (R1),R2         ;Get next input byte
```

```
     58                                      SWAB    R0              ;Reverse PRN high and low bytes
     59                                      XOR     R0,R2           ;Encrypt the byte
     60                                      SWAB    R0              ;Restore PRN high and low bytes for next seed
     61                                      MOVB    R2,(R1)+        ;Save encrypted bytes back into input buffer
     62                                      SOB     R3,1$           ;Repeat for entire input buffer
     63                              ;
     64                                      MOV     (SP)+,R3        ;Restore registers
     65                                      MOV     (SP)+,R2
     66                                      MOV     (SP)+,R1
     67                                      RETURN
```

```
                        .SBTTL  EDMTH3 -- Encryption/decryption meth 3 (swap bytes&shift bits)
                ;
                ;  Using a prn of repeat length same as input string length, select prn
                ;  numbered bytes from the input string, combine them into a word,
                ;  shift the combined bytes a random number of bits, recombine the shifted
                ;  bits and set the confused bytes back into the prn selected string bytes.
                ;  Sign of the byte count indicates: + = encryption; - = decryption.
                ;  If the byte count is 0 or 1, no encryption occurs. If the byte count is
                ;  odd, then one random selected byte will not be encrypted.
                ;
                ;  Inputs:
                ;       R0      Points to an arg block of the form:
                ;               R0 --->  buff_siz       ;Word holding byte length of buffer.
                                                        ;Note that buffer must be 512 or less
                                                        ;in length. Flag encryption by using
                                                        ;positive byte count ( 2 to 512. ).
                                                        ;Flag decryption by using negative
                                                        ;byte count (-2 to -512. ).
                                                        ;
                ;                       buff_addr       ;Address of buffer to be encrypted
                ;  Outputs:
                ;       R0      Randomized
                ;       Input buffer encrypted
                ;
EDMTH3: MOV     R1,-(SP)                ;Save registers
                MOV     R2,-(SP)
                MOV     R3,-(SP)
                MOV     R4,-(SP)
                MOV     R5,-(SP)
                MOV     (R0)+,R3                ;Save the string length
                MOV     @R0,-(SP)               ;And save the input buffer pointer
;  Initialize prn generator of desired length
                MOV     R3,R0                   ;Recover string length
                BGE     1$                      ;Branch if encryption
                NEG     R0                      ;If decryption, get real repeat
                INC     R3                      ;If neg, correct for ASR round down
1$:             CALL    INPRNM                  ;Set up for desired repeat length
;  Start encryption loop through string
                ASR     R3                      ;Repeat for 1/2 the string length
                CALL    GETLIC                  ;Get lic. num. for initial seed in R0
                CALL    EDPRNM                  ;Seed PRN generator (-adjacent pairs)
2$:             TST     R3                      ;Less than 2 bytes left?
                BEQ     9$                      ;Quit if so (odd len -> 1 byte unch.)
                CLR     R4                      ;Clean out shifting registers
                CLR     R5
                MOV     @SP,R1                  ;Retrieve buffer pointer
                MOV     R1,R2                   ;And second copy
;  Select first random byte
                CALL    EDPRNM                  ;Randomize in range 0 - <strlen-1>
                ADD     R0,R1                   ;Point to first random byte of pair
;  Select second random byte
                CALL    EDPRNM                  ;Randomize again
                ADD     R0,R2                   ;Point to next random byte
;  Use part of PRNM as semi-random shift amount
                MOV     R0,-(SP)                ;Save EDPRNM seed for later
                BIC     #^C6,R0                 ;Get a semi-random shift amount
;  Select encryption or decryption
```

```
                        TST     R3                      ;Positive for encryption
                        BMI     3$                      ;Branch if decrypting
                ;  Do this part for encryption
                        BISB    @R1,R4                  ;Get first byte without sign extend
                        SWAB    R4                      ;And put it in the high byte
                        BISB    @R2,R4                  ;Combine it with first byte
                        CLC                             ;Always do at least one shift
                        ROR     R4                      ;Shift once
                        ROR     R5                      ;Get low bit into r5
                        NEG     R0                      ;Right shifts for encryption
                        DEC     R3                      ;Reduce count of pairs remaining
                        BR      4$                      ;Skip decryption stuff
                ;  Do this part for decryption
                3$:     BISB    @R1,R5                  ;Get first byte without sign extend
                        SWAB    R5                      ;And put it in the high byte
                        BISB    @R2,R5                  ;Combine it with the first byte
                        CLC                             ;Always do at least one shift
                        ROL     R5                      ;Shift once
                        ROL     R4                      ;Get high bit into R4
                        INC     R3                      ;Reduce count of pairs remaining
                ;  Shift and recombine the {en!de}crypted bytes
                4$:     ASHC    R0,R4                   ;Shift combined bytes 0,2,4 or 6 more
                        BIS     R5,R4                   ;Recombine bytes
                        MOV     (SP)+,R0                ;Recover EDPRNM seed
                ;  Now put encrypted bytes back into input string
                        MOVB    R4,@R2                  ;Store low byte at second byte place
                        SWAB    R4                      ;Get high byte
                        MOVB    R4,@R1                  ;Store high byte at first byte place
                        BR      2$                      ;Repeat through string
                ;  Done, restore registers and return
                9$:     MOV     (SP)+,R0                ;Just pop saved buffer address
                        MOV     (SP)+,R5                ;Restore registers
                        MOV     (SP)+,R4
                        MOV     (SP)+,R3
                        MOV     (SP)+,R2
                        MOV     (SP)+,R1
                        RETURN
```

```
                        .SBTTL  EDPRNW -- Pseudo random number generator with MOD 2^16
                ;
                ;  Linear congruential pseudo-random number generator with maximum repeat
                ;  length of 65536 (2^16).  cf.  Hull and Dobell and Knuth, vol 2.
                ;
                ;  Inputs:
                ;       R0      Seed value
                ;
                ;  Outputs:
                ;       R0      New PRN, should be used for next seed
                ;
                EDPRNW:
                        MOV     R4,-(SP)        ;Save registers
                        MOV     R5,-(SP)
                        MOV     R0,R4           ;Get seed to be multiplied
                        MOV     (PC)+,R0        ;Fetch multiplier
                EDPRNA: .WORD   104375          ;Multiplier, can be replaced
                        MUL     R0,R4           ;Multiply by A
                        ADD     (PC)+,R5        ;Add C
                EDPRNC: .WORD   012705          ;Additive, can be replaced
                        MOV     R5,R0           ;Return result mod 65536. as PRN
                        MOV     (SP)+,R5        ;Restore registers
                        MOV     (SP)+,R4
                        RETURN
```

```
                                        .SBTTL  INPRNM -- Initialize PRN generator with repeat range M
                                ;
                                ;  Using the Hull and Dobell rules, determine acceptable values for
                                ;  A and C to get a repeat range of M.
                                ;
                                ;  Outputs:
                                ;       EDMULA  Set with first acceptable multiplier
                                ;       EDADDC  Set with first acceptable additive factor
                                ;       EDMODM  Set with desired repeat length
                                ;
                                INPRNM:
                                        MOV     #32.,EDMODM     ;Get repeat length to cover Pro ID
                                        MOV     #5,EDMULA       ;Use first valid A
                                        MOV     #3,EDADDC       ;And first valid C
                                        RETURN
```

```
                        .SBTTL  EDPRNM -- Generate pseudo-random number in specified range M
                ;
                ;  ************************************************************************
                ;  * INPRNM MUST BE CALLED BEFORE FIRST SEED IS PASSED TO THIS ROUTINE!!!! *
                ;  ************************************************************************
                ;
                ;  Using linear congruential method (cf. Hull and Dobell), generate
                ;  pseudo-random number using seed passed in R0.  Return new PRN in R0.
                ;
                ;  Inputs:
                ;       R0      Seed value, must be in range 0 to M (EDMODM)
                ;
                ;  Outputs:
                ;       R0      New pseudo-random number, should be used for next seed
                ;
                EDPRNM:
                        MOV     R4,-(SP)        ;Save R4 and R5
                        MOV     R5,-(SP)
                        CMP     R0,EDMODM       ;Is seed in range 0 to EDMODM?
                        BLO     1$              ;Branch and proceed if so
                        MOV     R0,R5           ;Set up to divide it by EDMODM
                        CLR     R4              ;Set up for divide
                        DIV     EDMODM,R4       ;Divide it
                        MOV     R5,R0           ;And use remainder as seed
                1$:     MOV     R0,R4           ;Get current seed ready to be multiplied
                        MUL     (PC)+,R4        ;Multiply by chosen A
                EDMULA: .WORD   25173.          ;Replace at run-time with 5
                        ADD     (PC)+,R5        ;Add in C
                EDADDC: .WORD   13849.          ;Replace at run-time with 3
                        CLR     R4              ;Clear high word for division
                        DIV     (PC)+,R4        ;Perform mod M
                EDMODM: .WORD   256.            ;Replace at run-time with 32.
                        MOV     R5,R0           ;Return remainder
                        MOV     (SP)+,R5        ;Restore R4 and R5
                        MOV     (SP)+,R4
                        RETURN
                ;
                .IFF    ;NE,PROCID              ;Assemble if protection code not included
 030066         DSKBUF:                         ;Define dummy DSKBUF global symbol
                .ENDC   ;NE,PROCID
                ;
                ;  Address of real top of TSINIT, including PRO init code
                ;
 030066         PROITP:
 000000                 .CSECT  TSXEND
                        .END
        000001
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  11342 Words  ( 45 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:03:19.50
```

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

Cross reference table (CREF V05.04)

```
$8BIT     1-129     39-72
$DEAD     1-122     14-18     14-65     27-69     28-27     37-20
$FORM     1-125     39-69
$HARD     1-154     14-20     14-67     22-30     27-27
$MEMSZ    1-141     44-102*
$NOIN     1-112     14-71
$OVRH     1-136     66-25
$PHONE    1-109     14-69
$SXON     1-90      39-25
$TAB      1-125     39-66
...V1      6-29      6-29      6-45      6-46     23-19     23-23     23-31     23-32     23-33     23-36     23-54     23-55
          23-61     23-63     23-67     23-78     23-176    24-221    24-226    24-227    24-243    24-243    27-84     27-85
          27-91     27-92     27-96     27-97     27-101    28-50     28-50     28-50     28-57     28-57     28-58     28-58
          28-58     28-62     28-62     28-62     28-70     28-70     28-72     28-72     28-84     28-95     28-96     28-102
          28-103    28-108    28-111    28-116    28-119    29-30     29-30     29-30     29-42     29-42     29-43     29-43
          29-43     29-47     29-47     29-47     29-55     29-55     29-56     29-56     29-68     29-77     29-78     29-84
          29-85     30-60     30-61     30-85     30-85     30-85     30-89     30-89     30-90     30-90     30-90     30-94
          30-94     30-94     30-99     30-99     30-101    30-101    30-110    30-131    30-132    30-138    30-139    38-21
          38-21     38-21     38-27     38-27     38-62     38-62     38-62     38-75     38-76     38-81     38-82     41-22
          41-22     41-22     41-29     41-29     41-31     41-31     41-68     41-68     41-68     41-72     41-73     41-73
          41-73     41-78     41-78     41-78     41-82     41-82     41-83     41-83     41-84     41-84     41-84     41-91
          41-101    41-102    41-108    41-109    42-43     42-43     42-43     42-47     42-48     42-48     42-48     42-52
          42-52     42-52     42-55     42-55     42-60     42-65     42-65     42-66     42-76     42-77     42-83     42-84
          44-151    44-152    46-76     46-77     50-76     50-76     52-25     52-25     52-25     52-39     52-39     54-25
          54-25     56-21     56-21     56-21     56-29     56-29     56-63     56-63     56-111    56-111    58-6      58-7
          58-10     59-36     59-36     59-50     59-50     60-35     60-35     66-17     66-17     66-51     66-52     68-24
          68-24     68-52     68-53     68-58     68-59     71-34     71-35     72-30     72-30     72-73     72-74     73-25
          73-25     73-25     73-54     73-54     73-81     73-81     73-93     73-94     73-99     73-104    73-105    74-78
          74-79     75-20     75-20     75-20     75-22     75-23     75-32     75-32     75-51     75-51     75-70     75-70
          75-95     75-95     75-138    75-139    77-25     77-25     77-32     77-34     77-53     77-54     77-57     78-19
          78-24     78-26     78-28     79-27     79-53     80-20     80-21     81-18     82-26     83-26
...V2      6-29      6-29      6-29#     6-29#    24-243    24-243    24-243#   24-243#   28-50     28-50     28-50#    28-50#
          28-57     28-57#    28-58     28-58     28-58#    28-58#    28-62     28-62     28-62#    28-62#    28-70     28-70
          28-70#    28-70#    28-72     28-72#    29-30     29-30     29-30#    29-30#    29-42     29-42#    29-43     29-43
          29-43#    29-43#    29-47     29-47     29-47#    29-47#    29-55     29-55     29-55#    29-55#    29-56     29-56#
          30-85     30-85     30-85#    30-85#    30-89     30-89#    30-90     30-90     30-90#    30-90#    30-94     30-94
          30-94#    30-94#    30-99     30-99     30-99#    30-99#    30-101    30-101#   38-21     38-21     38-21#    38-21#
          38-27     38-27     38-27#    38-27#    38-62     38-62     38-62#    38-62#    41-22     41-22     41-22#    41-22#
          41-29     41-29     41-29#    41-29#    41-31     41-31     41-31#    41-31#    41-68     41-68     41-68#    41-68#
          41-72     41-72#    41-73     41-73     41-73#    41-73#    41-78     41-78     41-78#    41-78#    41-82     41-82
          41-82#    41-82#    41-83     41-83#    41-84     41-84     41-84#    41-84#    42-43     42-43     42-43#    42-43#
          42-47     42-47#    42-48     42-48     42-48#    42-48#    42-52     42-52     42-52#    42-52#    42-55     42-55
          42-55#    42-55#    42-60     42-60     42-60#    42-60#    42-65     42-65#    50-76     50-76#    52-25     52-25
          52-25#    52-25#    52-39     52-39     52-39#    52-39#    54-25     54-25     54-25#    54-25#    56-21     56-21
          56-21#    56-21#    56-29     56-29     56-29#    56-29#    56-63     56-63     56-63#    56-63#    56-111    56-111#
          59-36     59-36     59-36#    59-36#    59-50     59-50     59-50#    59-50#    60-35     60-35     60-35#    60-35#
          66-17     66-17     66-17#    66-17#    68-24     68-24     68-24#    68-24#    72-30     72-30     72-30#    72-30#
          73-25     73-25     73-25#    73-25#    73-54     73-54     73-54#    73-54#    73-81     73-81#    75-20     75-20
          75-20#    75-20#    75-32     75-32     75-32#    75-32#    75-51     75-51     75-51#    75-51#    75-70     75-70
          75-70#    75-70#    75-95     75-95#    77-25     77-25     77-25#    77-25#
AHEND      1-106     49-21
ALBFX     24-161     37-12#
ALCHRB    23-142     26-12#
ALCOVL    66-34      68-15#
ALCSLO    23-103     36-12#
ALCWRK    23-138     25-12#
```

Cross reference table (CREF V05.04)

```
ALOCBF   23-102   35-14#
AREA      1-182#   6-29    23-19   23-23   23-61   23-63   23-78   23-176  24-221  24-243  28-50   28-58
         28-62    28-70    29-30   29-43   29-47   29-55   30-85   30-90   30-94   30-99   38-21   38-27
         38-62    41-22    41-29   41-31   41-68   41-73   41-78   41-82   41-84   42-43   42-48   42-52
         42-55    42-60    52-25   52-39   54-25   56-21   56-29   56-63   59-36   59-50   60-35   66-17
         68-24    72-30    73-25   73-54   75-20   75-32   75-51   75-70   77-25   77-32   77-34   78-19
         78-24    78-26    78-28
AUTHAN    1-106   49-21    49-23
BADDEV   28-104   28-116#  29-86   30-140  41-110  42-85
BADLIN    4-6#    27-85
BADOPN    4-11#   28-96    28-103
BADVEC   27-47    27-49    27-51   27-91#
BASMAP    1-100   46-52*
BDLMSG    4-8#    27-97
BDSPOP    4-14#   30-61
BDVMSG    4-7#    27-92
BOSF      4-15#   30-132   30-139
BRKPT     1-101   75-113
BYTES     4-32#   23-36
C.CSW     1-142   31-30*   76-24*  76-35*
C.DEVQ    1-142   31-31*   76-33*
C.SBLK    1-142   31-23*
C1DEVX    1-142   31-27    39-112*
CA$BLK    1-147   74-52*
CA$DVU    1-147   74-54*
CA$HBL    1-148   74-64*
CA$HFL    1-148   74-62*
CA$HSH    1-148   74-66*
CA$UBL    1-148   74-60*
CA$UFL    1-148   74-58*
CA$WCT    1-147   74-56*
CC$$SZ    1-149   35-74
CCAFLG    1-186#  23-23    80-27
CCATST   24-247   80-14    80-27#
CCBHD     1-149   35-73*
CCLNAM    1-193#  38-62    38-65
CCLSAV    1-127   38-64
CD$$SZ    1-86    35-64
CDX$DH    1-141   35-104
CDX$DL    1-150   14-22
CDX$DZ    1-149   14-50    14-75   18-37
CDX$VH    1-152   14-52    14-77   18-40   35-106
CFHMSG    4-18#   28-116   52-33   56-23
CHAIN     1-116   23-85
CHKCLD   30-40    32-13#
CHKMEM   24-149   35-140   54-20   75-31   75-39   80-11#
CHNSIZ    1-121   76-21
CKLIN    24-34#
CL$EPN    1-126   39-58*
CL$EPP    1-126   39-52*
CL$EPS    1-125   39-31*   39-51*
CL$LEN    1-156   39-79*
CL$LIX    1-155   22-19*   39-23
CL$OPT    1-126   39-75*
CL$ORA    1-124   39-43*
CL$ORB    1-124   39-36*
```

Cross reference table (CREF V05.04)

```
CL$ORE    1-124     39-46*
CL$ORG    1-124     39-38*
CL$ORP    1-124     39-37*
CL$ORS    1-126     39-44*
CL$STA    1-126     39-83*
CLDEVX    1-142     31-29     39-94*
CLEOFS    1-125     39-53     39-54
CLHEAD    1-142     39-99     39-117
CLINCP    1-154     22-21
CLINIT   24-73      39-11#
CLK100    1-187#     5-165    23-14*
CLKRTI    1-106      5-71
CLKVEC    1-157      5-71*     5-72*     5-165*    23-14     24-78*
CLORSZ    1-111     39-39
CLOTIR    1-154     22-20
CLSIZE    1-142     39-100
CLSTS     1-117     39-96     39-114
CLTOTL    1-112     24-71     31-18     39-18     39-101    39-108    69-149
CLVEND    3-68#     39-132
CLVERS    1-110     39-123    39-137*
CLVTBL    3-60#     39-128
CO$8BT    1-129     39-74
CO$DEF    1-125     39-62
CO$FF     1-125     39-71
CO$TAB    1-125     39-68
CONFG2    1-104     23-165*   23-166*   23-169*   24-79
CONFIG    1-103     23-163*   24-5
CONSPC    4-13#     28-108
COSRT     4-34#     73-94
CRLF      4-9#      27-96     27-101    28-111    28-119    58-10     73-99     77-57
CS$ENT    1-144     76-24
CS$NMX    1-88      76-34
CS$OPN    1-144     76-24
CSHALC    1-121     35-71     69-161    74-17     74-18     74-28     74-44
CSHBAS    1-104      7-39      9-37*
CSHBFP    1-147     74-47*
CSHBUF   24-206     74-12#
CSHDEV    1-86      35-63*
CSHDVN    1-86      35-67*
CSHOVF    4-39#     74-79
CSHSIZ    1-147     74-33*
CSHVEC    1-105      7-42
CURDEV    1-204#    59-54*    60-40     62-45
CURNAM    1-205#    51-25*    56-20*    58-8
CVTDVU   30-37      33-14#    76-29
CW$50H    1-103     24-5
CW$BTH    1-102     23-161
CW$ESP    1-112     23-166
CW$FB     1-102     23-162
CW$FGJ    1-102     23-162
CW$GDH    1-102     23-161
CW$LGS    1-102     23-161
CW$PRO    1-106     23-116    24-79
CW$QBS    1-114     23-169
CW$USR    1-103     23-162
CW$XM     1-103     23-162
```

Cross reference table (CREF V05.04)

```
CXTALC  24-93     45-12#
CXTBAS   1-111    45-24
CXTBUF   1-118    35-134*
CXTPAG   1-87     28-40     38-41     45-45*    47-31
CXTPDR   1-110    45-38*
CXTRMN   1-111    45-23*    45-24*
CXTSIZ   1-110    45-16
CXTWDS   1-110    45-30*    45-42
DATIMH   1-104    24-29*
DATIML   1-104    24-30*
DC$$SZ   1-132    37-64
DCCSIZ   1-85     69-92
DEVSIZ   1-93     24-50*    24-65*    39-98*    39-116*   53-45*
DEVVEC   5-139    11-6#
DFJMEM   1-133    47-50*
DHBFSZ   1-108    35-110    35-112
DHINIT  14-56     19-8#
DHLPRM  14-82     15-8#
DHOINT   1-135    18-42
DI$CL    1-117
DI$DU    1-155     2-30
DI$LD    1-117    24-63
DI$MU    1-155     2-32
DI$PI    1-156
DI$TT    1-118    24-48
DI$XL    1-155     2-31
DM$CSR   1-151    15-13     15-15*    15-16*    19-27
DM$LSR   1-151    15-17*
DMYDEV   1-95     30-34     30-57     49-25     73-20
DOHNLC  54-45     56-106    59-13#
DORTI   75-144#
DS$ABT   1-124
DS$DIR   1-138    24-63     53-50
DS$NRD   1-129    53-52
DS$SFN   1-139    24-63
DS$VSZ   1-138    24-63
DSKBUF   1-58     89-39#
DSTBLK   1-194#   79-27     79-32     79-39
DTLX     1-103    38-52*
DTYPE    1-128    49-27
DV$$SZ   3-6#     51-58
DV$FLG   3-5#     51-56
DV$NAM   3-4#     51-54
DVFLAG   1-138    24-49*    24-64*    39-97*    39-115*   50-43     53-30*    53-54*    64-31*    64-42
DVFLBS   3-13#    51-53
DVFLND   3-42#    51-59
DVSTAT   1-89     24-48*    24-63*    39-96*    39-114*   53-46*    53-50     53-52
DW$$SZ   1-114    69-137
DX$DMA   1-138     3-16      3-18      3-19      3-20      3-21      3-22      3-23      3-26      3-28      3-31      3-32
         3-33      3-34      3-37      3-38      51-65
DX$EBA   1-92      3-39      24-64     51-67
DX$IBH   1-97      3-18      3-31      3-32      3-33      3-34      50-48
DX$MAP   1-145     50-52     64-31     64-46
DX$MPH   1-97      3-14      3-15      3-16      3-18      3-24      3-25      3-27      3-29      3-30      3-31      3-32
         3-33      3-35      3-36      3-38      3-40      3-41      50-46
DX$NCA   1-157     3-39      39-97     39-115    53-54
```

Cross reference table (CREF V05.04)

```
DX$NHM    1-97      3-17      3-19      3-23      3-26      3-34      3-39      50-44
DX$NMT    1-129     39-97     39-115    53-54
DX$NRD    1-130     39-97     39-115
DX$NST    1-137     3-23      3-34
DZINIT   14-58      17-8#
DZOINT    1-135     18-36
EMMAP     1-99      5-123     44-65     44-109    56-77     56-82     61-36     61-55     72-47     72-52     73-65     73-71
EMTENT    1-107     5-89
ENTVEC    7-35      7-43      7-51      8-11#
ERHMSG    4-19#     52-41     54-27     56-65
ERHNDV    4-20#     52-58
ERP      27-86      27-94#
ERRLOG    1-134     57-22
EXCBUF    1-84      25-17
EXTLSI    1-245#    44-119    50-54     64-24
FC$$SZ    1-134     69-69
FC$LBN    1-115     37-61
FETDEV    1-198#    79-23*    79-27     79-53
FF$$SZ    1-115     37-56
FILBLK    1-203#    56-49*    56-63     56-90*    72-20*    72-30     72-57*
FMEMHI    1-200#    24-168    44-103*   44-114    46-65*    56-105*   56-107    63-27     63-34*    63-35
FMEMLO    1-201#    24-167*   46-37     46-62     63-28
FNDHRB    1-58      62-18#
FORCEO   30-26      34-8#
FORK      1-89      57-15
FPTRAP    1-117      5-95
FQ$$SZ    1-95      35-37
FREIOQ    1-117     35-18*
FREPGS    1-118     47-22*
FRKGEN    1-95      35-34     35-37
FRKINI    1-95      35-36*
FSTDL     1-94      37-22
FSTIOL    1-154     22-11     35-102    36-24
FW$$SZ    1-115     37-58
GENTOP    1-156     23-27
GETHNH   24-202     55-13#
GETHNL   24-97      49-14#
GETMAP   24-172     67-12#
GETODT   23-149     75-13#
GETOVL   67-34      71-12#
GETSRT   24-180     73-13#
GTBYT     1-101     57-25
GTLIN     1-229#    23-51
H.CSR     1-95      52-79
H.DSTS    1-89      52-48     53-46
H.DVSZ    1-93      53-45
H.GEN     1-89      52-67     52-71     56-30
H.INS     1-95      52-88
H.SIZ     1-93      53-44
HANDSK    1-83      53-38*
HANENT    1-89      11-14     24-51*    24-66*    39-99*    39-117*   54-33*    54-34*    56-34*    59-21     59-61
HANMAP   56-99      59-22     59-63     60-41     61-14#
HANNAM    1-230#    52-24*    52-25     56-19*    56-21
HANPAR    1-97      11-20     12-25     12-54     50-60*    50-71*    54-16*    55-23     56-47*    61-26     61-32
HANRCB    1-83      26-20*    62-23
HANRCO    1-83      26-21*    26-22*
```

Cross reference table (CREF V05.04)

```
HANSIZ    1-93     39-102*  39-118*  50-64    53-44*   54-18    56-39    56-98
HANUMP   56-101    59-24    59-72    59-79    60-31    61-49#
HANXIT    1-97     12-59
HANXMR    1-58     63-22#
HF$LIN    1-150
HF$MC     1-151    19-21    19-22
HF$RIE    1-150
HF$TIE    1-150
HF$TSB    1-150
HGENFL    1-237#   52-71*   56-30*   57-17    57-20
HIMAP     1-94      5-133   46-56*   47-20
HLERR    50-67     52-34    52-42    52-59    52-70    54-28    56-24    56-66    58-5#
HMAP      1-197#   56-48*   56-73    56-89*
HN2BIG    4-27#    50-66
HSGER     4-22#    52-69
HV$$SZ    2-20#    52-60
HV$ID     2-17#    52-49
HV$UPD    2-19#    52-54
HV$VER    2-18#    52-51
HVEND     2-33#    52-61
HVTBL     2-29#    52-46
ICONFG    1-241#   44-119*  50-54    64-24
II$$SZ    1-85     35-57
ILSW2     1-112    14-69    39-65
INDDBL    1-139    41-40*
INDDBS    1-139    41-45*   41-56
INDFIL    1-84     41-62    41-68    41-73    41-78    41-84    41-89    41-91
INDINI   24-118    41-12#
INDNAM    1-232#   41-21*   41-22    41-51
INDOPN    4-40#    41-102   41-109
INDSAV    1-139    41-17*   41-50
INDTSV    1-139    41-88
INICLK   24-5#
INIDEV   24-47#
INIJMP    1-107     5-161
INIOVL    5-13      7-9#
INISTP    5-165#   23-37    23-56    24-228   27-102   28-112   28-120   30-64    30-133   38-77    38-83    44-153
         46-78     58-11    66-53    68-54    68-60    71-36    72-75    73-100   73-106   74-80    75-24    75-140
         77-58     80-22    80-29
INITGO    1-57     23-8#    23-104   23-106
INITOP    1-57      1-207   23-132   23-134   65-52    65-54    83-37#
INMXV     1-108    18-16
INRECV    1-108    14-28
INSCK1   50-25     51-16#
INSCK2   50-31     52-14#
INSTBL    1-85     35-54*
INSTBN    1-85     35-59*
INTEN     1-89     57-16
INTMX1    1-109
INTSND    1-113     5-24*    5-152    23-113*
INTSSZ    1-84     23-122
INTSTK    1-113     5-155   23-123*
INVEC     1-87     14-27    27-38
IOMAP     1-99      1-137    5-126    10-37
IOPAGE    1-99     43-29
IOQSIZ    1-118    35-19
```

Cross reference table (CREF V05.04)

```
IOTIMR    1-134     57-19
JCXPGS    1-133     47-32*
JM$$SZ    1-111     35-44
JMPO      1-103      5-78
JSWLOC    1-116     23-85
KEYSEG    7-23       9-10#    71-22
KMNBAS    1-119     38-34
KMNCHN    1-98      38-56
KMNHI     1-96      38-35*
KMNNAM    1-192#    38-21    38-57
KMNPGS    1-98      38-42*   47-26
KMNSTK    1-98      38-44*
KMNSTR    1-98      38-46*
KMNTOP    1-96      38-32*
KPAR0     1-98      43-14
KPAR5     1-135      5-130*  11-9     11-22*   11-62*   12-52    12-55    56-72    56-73*   56-84*   60-25    60-46*
         61-32*    61-57*   72-42    72-43*   72-54*
KPAR6     1-157     59-71*   59-78*   60-26    60-27*   60-45*
KPAR7     1-100     43-29*
KPDR0     1-100     43-16
LCDTYP    1-109     14-22    14-72    35-104   35-106
LCLUNT    1-136     22-13*   22-18    22-26*   39-26*
LD1TBL    3-77#     40-15
LDDEVX    1-138     24-56*   24-61*
LDHAND   49-31      49-46    50-21#
LDHB1B    1-110     35-108*
LDHB1P    1-110     35-109*
LDHB2B    1-110     35-111*
LDHNHI   55-29      56-12#
LDHNLO   50-72      54-12#
LDINIT   24-67      40-8#
LDREAD   59-64      60-15#
LDVERS    1-114     40-9     40-21*
LHIRBA    1-90      36-30    36-39*
LHIRBB    1-90      36-43*
LHIRBC    1-91      36-54    36-60*   36-61    36-67*
LHIRBE    1-90      36-48*
LHIRBG    1-90      36-45*
LHIRBP    1-90      36-44*
LHIRBS    1-91      36-46*
LINBUF    1-94      37-28*
LINCHK   24-34      27-9#
LINEND    1-96      37-32*
LININI    5-113     14-8#
LINIR     1-154     22-21*   22-29*
LINSIZ    1-94      37-29
LINSPC    1-97      37-30*
LINTRP   27-41      27-43    27-68    27-84#
LINTYP   22-9#      27-26
LMXLN     1-150     14-92    15-16    16-14
LMXNUM    1-87      14-73    15-12    16-12    27-29    39-29
LODINI    6-9#       7-24
LODOVL   71-26      72-9#
LOKBAS    1-105      7-47     9-33*
LOKCSH    1-132     37-71*
LOKMEM    1-100     37-55*
```

Cross reference table (CREF V05.04)

```
LOKVEC    1-105     7-50
LOMAP     1-93      5-131    46-50*   47-21
LOTBUF    1-96     37-38*
LOTEND    1-96     37-41*
LOTSIZ    1-96     37-39     37-40    39-42
LOUTIR    1-154    22-20*    22-28*
LOWEND    3-109#   65-39
LOWOVL    3-106#   65-36
LSTDL     1-94     37-24
LSTHL     1-154    14-43     14-64    35-114   36-17
LSTLIN    1-126    27-21
LSTMX     1-116    14-48     35-90
LSTPL     1-87     22-9      28-26    35-100   36-22
LSTSL     1-88     37-46     41-57    42-32
LSW10     1-90     39-25*
LSW3      1-112    14-18     14-20    14-65    14-67    14-71*   22-30*   27-27    27-69*   28-27    37-20
LSW5      1-157
LSW6      1-122
LSWPBK    1-88
LTTPAR    1-105    37-26*
LXCL      1-84     22-27*
MA$SYS    1-112     5-135
MAPALC   24-212    46-27#
MAPPAR    1-97      5-130    46-64*
MAPSIZ    1-99     44-99     44-101
MAPSYS    1-136     5-104    75-118
MAXDEV    1-83     59-59
MAXOVL    3-98#     3-100
MAXSLO    1-91     36-33     36-35
MB$$SZ    1-144    69-99
MEM256    1-132     5-124    44-116*  44-125   56-75    56-80    60-21    60-48    61-34    61-53    72-45    72-50
         73-63     73-69
MEMINI   23-155    43-6#
MEMLIM    1-236#   23-72*    80-11
MEMPAR    1-130     5-93
MEMTST   24-89     44-24#
MF$CM     1-151    19-30
MF$CS     1-151    19-29
MF$LE     1-151    15-17
MF$LIN    1-149    15-15
MH$LPR    1-151
MH$PBR    1-152
MH$SCR    1-150    19-12     19-21*   19-22
MHNSIZ    1-107    56-45*
MI$$SZ    1-146    64-56     64-68
MI$LNK    1-145    37-86     64-71*
MI$SBP    1-145    37-82*
MINCTR    1-119
MIOBHD    1-145    37-80     64-65*
MIODBG    1-127    64-22
MIOFLG    1-145    37-78     64-51*   69-143
MIONWB    1-146    64-57     64-76
MIOWHD    1-146    64-75*
MMENBL    1-93      5-118    44-62    44-110   56-74    56-83    61-33    61-56    72-44    72-53    73-62    73-72
MONFQH    1-111    35-43*
MONVEC    1-119     5-17     26-22    52-92    61-22
```

Cross reference table (CREF V05.04)

```
MPARO     1-92      48-20
MPAR16    1-92      48-25
MPARFL    1-92      48-7
MPIVSZ   12-19#     24-105
MPPHY     1-101     57-26
MR$$SZ    1-144     69-102
MSGBAS    1-102      9-21*
MTSXDV   77-40      77-53#    78-35
MU$TXT    1-116     69-107
MUXVEC   17-17      18-9#     19-17     20-17
MVSIZ     1-130     45-25     45-49
MW$$SZ    1-146     64-57     64-78     64-82
MW$LNK    1-146     64-79*
MXCSR     1-87      17-12     17-21*    17-22     27-31     27-72*
MXDTR     1-109     17-26*
MXJADR    1-141     47-43*
MXJMEM    1-133     47-38*
MXLNT     1-117     35-92*
MXLPR     1-112     14-94*
MXRBUF    1-109     17-24     20-27
MXTYPE    1-108     14-50     14-52     18-37     18-40
MXVEC     1-87      18-15     27-33
NDL       1-96      28-25
NDVRCB    1-83      26-16
NEDCHR    1-154     22-28
NEXMSG    4-25#
NFRESB    1-114     30-118*
NFSBLK    1-183#    42-60     42-61     42-62
NIOL      1-121     39-101
NLINES    1-120     37-58
NMSNMB    1-131     35-50
NMXHAN    1-196#    24-102    50-59*
NOCCL     4-17#     38-76
NOCLOK    4-23#     24-227
NOKMON    4-16#     38-82
NOODT     4-26#     75-23
NOSYDV    4-29#
NOXM     23-61      44-151#
NSCP      1-86      35-130
NSIP      1-85      35-56
NSL       1-96      28-25
NSPLBL    1-113     30-80     30-117    69-78
NSPLDV    1-99      30-22*    30-44*    30-50*    30-75     30-114
NSPLFL    1-113     35-78
NUMCCB    1-94      35-74
NUMCDB    1-85      37-54*
NUMDCD    1-132     37-53*
NUMDEV    1-88      11-57     24-52*    24-59*    24-60     33-31     39-92*    39-93     39-110*   39-111    51-43     53-24*
         53-25      55-34     64-33     64-44     77-35     78-29
NUMFRK    1-95      35-34     35-37
NUMIOQ    1-117     35-19
NUMRDB    1-120      1-148    24-176
NXIVMH    1-84      24-104
NXMMSG    4-24#     44-152
O. ADR    1-135      9-15     65-35     68-30*    69-22
O. BLK    1-135      6-29     68-24     72-20
```

Cross reference table (CREF V05.04)

```
O.PAR     1-135     6-34*     72-25*
O.SIZ     1-135     6-21       6-23     68-34     69-26     72-19
ODTBAS    1-178#    75-44     75-68
ODTBLK    1-231#    75-20
ODTFLG    1-184#    23-89*    23-147
ODTRDM     4-28#    75-139
ODTRDX    75-34     75-52     75-73     75-138#
ODTSTA    1-235#    75-46*    75-123
ODTTOP    1-185#    75-58*    75-129
ODTTRP    1-123     75-128*
OORCLO    69-60     69-149#
OORCSH    69-62     69-161#
OORDBG    69-61     69-155#
OORDMP    69-63     69-167#
OORLOK    69-53     69-88#
OORMIO    69-59     69-143#
OORMSG    69-54     69-97#
OORPLS    69-56     69-114#
OORSLE    69-57     69-129#
OORSPL    69-52     69-75#
OORSWP    69-55     69-123#
OORUSR    69-51     69-68#
OORWIN    69-58     69-135#
OOXFIN    69-175    69-190#
OOXNO     69-76     69-89     69-98     69-115    69-124    69-130    69-136    69-144    69-150    69-156    69-162    69-168
          69-173#
OOXYES    69-37     69-71     69-84     69-91     69-93     69-110    69-119    69-125    69-131    69-139    69-145    69-151
          69-157    69-163    69-169    69-179#
OPNCHN    30-48     76-14#
OPNKMN    24-114    38-17#
OPNRSF    24-140    29-11#
OPNSWP    24-136    28-13#
OPTOVL    67-21     68-42     69-15#
OS$$SZ     3-96#     3-100     7-26     65-46     66-36     67-23     67-35     70-31
OS$FLG     3-94#     7-21     65-42*    65-64*    66-33*    67-32     70-22
OS$OVL     3-95#     6-15      9-14     65-34     66-32*    68-19     69-21     72-18
OS$SIZ     3-93#     6-19      7-25     65-32     65-43     65-65     66-35*    67-22*    67-30     70-20     70-24     70-28
          70-28     71-16
OSEND      3-101#
OSLAST     3-102#    7-27     65-47     66-40*    67-24     67-36     70-32
OSTABL     3-100#    3-102     7-20     65-31     66-31     67-20     67-29     70-19
OSZ       1-175#
OTMXV      1-108    18-17
OTRECV     1-108    14-29
OVLBAS     1-202#    7-14     65-30*
OVLBLD    65-23     66-10#
OVLEND    69-35     69-64#
OVLLST    69-31     69-50#
OVLPOS    23-127    65-15#
OVLTRY    65-62     70-14#
OVRADD     1-136    66-27*
PARENL     1-92     48-21
PARSET    24-85     48-7#
PHSOVF     4-33#    46-77
PHYMEM     1-141     5-121    44-98*
PIINSZ    23-112#   23-120
```

Cross reference table (CREF V05.04)

```
PISRT     1-57      1-249#
PMPAR     1-131     37-93*
PMSIZE    1-131     37-91
PNAME     1-89      24-47*    24-62*    33-32     39-95*    39-113*   51-45     53-29*    56-19     56-20     77-36     78-30
          78-42
PPTERM    1-57      1-254#
PROASM    1-20      1-32      1-57      1-63      5-29      5-143     10-1      13-1      14-1      24-38     24-187    27-1
          44-45     44-55     48-1      64-1
PROBUF    1-58      1-206#    23-119*
PROCID    1-24      23-95     23-146    75-1      75-60     84-1
PROFLG    1-98      5-53      23-118*
PROITP    1-57      89-44#
PRTDEC    23-35     27-100    28-110    82-9#
PRTOCT    27-95     81-8#
PRTR50    28-118    30-63     58-9      73-97     77-56     83-8#
PSW       1-94      44-71*    44-88*    56-71*    56-85*    61-18*    61-65*    72-41*    72-55*    73-59*    73-61*    73-73*
PTBYT     1-100     57-24
PTWRD     1-100     57-23
PVSPBL    1-114     30-115
QBUS      1-137     23-167    44-117
R50C1     1-222#    32-30     39-113
R50C10    1-223#    32-32     32-36
R50C17    1-224#    32-34
R50CHR    4-42#     83-17
R50CL     1-219#    32-17     39-95
R50CL0    1-220#    32-19     32-23
R50CL7    1-221#    32-21
R50CSH    1-213#    9-35
R50LD     1-217#    24-62     51-30
R50LOK    1-211#    9-31
R50LS     1-226#
R50MSG    1-209#    9-19
R50ODT    1-227#    23-87
R50PI     1-218#
R50SY     1-216#
R50TIO    1-214#    9-39
R50TT     1-215#    24-47
R50USR    1-212#    9-27
R50VM     1-225#
R50WIN    1-210#    9-23
RBD       1-177#    75-47
RBR       1-116     14-102
RC$$SZ    1-121     35-28
RDB       1-120     24-178
RDBEND    1-120     24-182
RDERR     4-37#     6-46      66-52     68-53     72-74
RDINT     1-116     14-106
REDUCE    4-31#     23-33
RELFIL    75-58#
RELOC     1-101     57-27
REQMIS    4-10#     77-54
RID       1-176#
RLBF      1-233#    75-59*    75-64*    75-65     75-70     75-75
RLBFND    1-234#    75-65*    75-66*    75-87
RMNPDR    1-112     45-56*
RMON      1-104     5-17*     5-167*    23-13     23-115    23-159    52-91     52-92*    52-95*    61-22*    61-61*
```

```
RPRVEC    1-102     11-48     52-93     59-58
RSFBLK    1-140     29-23     29-30     29-43     29-47     29-63     29-68
RSFERR    4-12#     29-78     29-85
RSR       1-87      14-104    27-37     39-27
RSZ       1-174#    75-35
RT$$SZ    1-122     24-181
RT$BAS    1-121     73-48*
RT$NAM    1-138     73-33*    73-34*
RT$SKP    1-120     73-39
RT$TOP    1-120     73-42*
RTFTCH   28-20      29-24     30-78     41-63     42-37     79-17#
RTMNVC    1-189#     5-167    23-13*    61-61
RTNKM    44-60      44-146#
RTTRP4    1-188#     5-166    23-12*
SAVBLK    1-190#
SB$$SZ    1-131     35-50
SCHED     1-88
SCPFHD    1-86      35-129*
SDANAM    1-119     30-30
SDCB      1-127     30-23
SDCBSZ    1-128     30-68
SDCHAN    1-128     30-47     31-23*    31-30*    31-31*
SDDVU     1-91      30-38*
SDNAME    1-128     30-28*    30-57*
SEGCHN    1-140     29-62
SETCHN   28-80      29-64     30-109    38-58     38-66     41-52     41-90     77-18#
SETJSZ   24-216     47-16#
SETLIN   14-49      14-64#
SETMIO   24-144     64-18#
SETMUX   14-48#
SETSY    24-110     78-14#
SETUMP    5-109     10-14#
SETVEC   11-36      11-50     12-21#
SFCB      1-113     35-81*
SFCBFH    1-113
SFCBND    1-113     35-83*
SFCBSZ    1-113     35-80
SG$ELG    1-104     23-172    57-20
SG$IOT    1-104     23-173    57-17
SG$MMU    1-105     23-173    52-67
SG$MTM    1-105     23-173
SG$MTS    1-105     23-172
SG$PAR    1-105     23-172
SG$TSX    1-141     23-173
SHRRCB    1-120     35-23*
SHRRCN    1-120     35-30*
SIZERR   47-59      79-69     80-13     80-20#
SKPDEV    1-228#    51-35
SLTSIZ    1-116     28-41*
SMRSIZ    1-149     67-15*    67-41     67-43*
SNMSHD    1-131     35-49*
SP$$SZ    1-86      35-130
SPLANM    1-127     30-25
SPLBLK    1-130     30-77     30-85     30-90     30-94     30-108    30-110
SPLCHN    1-130     30-107
SPLCLD   30-42      31-13#
```

Cross reference table (CREF V05.04)

```
SPLDEV    1-127     30-24     49-42
SPLDVN    1-128     30-70
SPLINI   24-130     30-7#
SPLNB     1-121     69-77
SPLND     1-127     30-7      49-40     69-75
SPNEED   28-97      28-108#   29-79     41-103    42-78
SROMMR    1-98       5-118*   23-62     44-62*    44-110*   56-74*    56-83*    60-24     60-47*    61-33*    61-56*    72-44*
         72-53*     73-62*    73-72*
SR3FLG    1-99       5-119    44-39*    44-63     44-107
SR3MMR    1-99       5-123*    5-126*   10-37*    44-37     44-65*    44-109*   56-77*    56-82*    60-23     60-50*    61-36*
         61-55*     72-47*    72-52*    73-65*    73-71*
SRTOVF    4-38#     73-105
SRTSIZ    1-149     24-185*
SS        1-116     23-44
SSEND     1-118     23-41
STA       1-172#    75-43
STDVTB   50-38      53-20#
STHNPV   54-41      56-100    57-11#
STK       1-173#
STKLVL    1-84       5-25*     5-157*
SVERR     4-35#
SWDBLK    1-88      28-19     28-50     28-58     28-62     28-79     28-84
SWPCHN    1-88      28-78
SWPJOB    1-115     35-127*
SWPPOS    1-115     35-125*
SYINDX    1-119     78-36*
SYNAME    1-140     24-126    41-21     78-43*
SYSDAT    1-123     24-223*
SYSGEN    1-106     23-174*
SYSMAP    1-136     75-117
SYSUPD    1-155     39-127    52-54
SYSVER    1-155     23-177*   39-125    40-13     52-51
SYTIMH    1-123     23-19     24-221
SYTIML    1-123     24-220    24-224
SYUNIT    1-118     78-20*    78-40
TAKOVR    5-9#      24-251
TIOBAS    1-109      7-32      9-41*
TIOVEC    1-106      7-34
TK1SEC    1-94      24-10*    24-19*
TK1VAL    1-108     24-13*    24-22*
TK3SVL    1-133     24-12*    24-21*
TK5VAL    1-133     24-11*    24-20*
TOOBIG    4-30#     23-32     80-21
TOPMEM    1-199#    23-68*    79-47
TRCSET   44-34      44-141#   52-76     52-77
TRP10     1-107      5-81
TRP14     1-123      5-83
TRP20     1-107      5-85
TRP24     1-107      5-87
TRP250    1-123      5-97
TRP34     1-107      5-91
TRP4      1-106      5-79
TSEMT     1-92      75-116
TSEXEC    1-101     75-115
TSGEN     1-101     75-114
TSINIT    1-6#       1-57     23-77     23-101    28-70     29-55     30-99     75-112
```

Cross reference table (CREF V05.04)

```
TSR       1-116    14-99
TSTWRD   44-22#    44-86*    44-89*    44-91
TSXHD     4-5#      6-45     23-31     23-54     24-226    27-84     27-91     28-95     28-102    29-77     29-84     30-60
         30-131    30-138    38-75     38-81     41-101    41-108    42-76     42-83     44-151    46-76     58-6      66-51
         68-52     68-58     71-34     72-73     73-93     73-104    74-78     75-22     75-138    77-53     80-20
TSXPRO    1-33      1-34#     1-73      1-74#
TSXRUN    4-21#    23-55
TSXSAV    1-191#
TSXSIT    1-111
TSXSIZ    4-36#    68-59     71-35
TTINCP    1-153    22-29
UBUSMP    1-137    10-21     44-127*   50-50
UC$$SZ    1-143    42-25
UCLBLK    1-143    42-28*
UCLDAT    1-143    42-36     42-43     42-48     42-52     42-64*    42-66
UCLINI   24-122    42-10#
UCLNAM    1-140    24-126*
UCLOPN    4-41#    42-77     42-84
UEXINT    1-130     5-42
UEXRTN    1-129     5-46
UK$$SZ    1-143    42-22
UMODE     1-117    44-71     44-88     44-146
UMRADR    1-137    10-27
UMSYTP    1-118    24-154*
UNIBUS    1-137    44-123
UPAR0     1-100    43-15
UPAR6     1-122    73-57*
UPAR7     1-93     44-70*    44-87*
UPDR0     1-100    43-17
UPDR6     1-122    73-58*
UPMODE    1-121    73-59
US$$SZ    1-143    42-23
USRBAS    1-103     9-29*
VBUSTP    1-119    23-167    44-117    44-123
VCSHNB    1-126    74-17*
VDBFLG    1-92     69-155
VDFMEM    1-133    47-47     47-49
VF$LIN    1-153    16-13
VF$MR     1-152    20-21     20-22
VF$RE     1-153    16-15
VF$RIE    1-152
VF$SC     1-153
VF$TIE    1-152
VH$CSR    1-152    16-13*    16-14*    20-12     20-21*    20-22
VH$LCR    1-153    16-15*
VH$LPR    1-152
VHIMEM     1-87    28-38     47-35     47-37
VHINIT    14-54    20-8#
VHLPRM    14-87    16-8#
VHOINT    1-153    18-39
VINABT    1-122    27-67     30-58     52-30     52-56     73-31
VLDSYS    1-144    24-57     51-32
VMAXMC    1-144    69-97
VMIOBF    1-146    64-55     64-66
VMIOSZ    1-145    37-83
VMLBLK    1-115    37-59
```

Cross reference table (CREF V05.04)

```
VMSCHR    1-91      69-104
VMXCSH    1-86      35-65
VMXMON    1-111     35-41
VMXMRB    1-147     69-101
VMXMSG    1-144     69-108
VMXSF     1-115     37-51     37-57     69-88
VMXSFC    1-115     37-54     37-62
VMXWIN    1-114     35-25     35-27     69-135
VNCSLO    1-91      36-32
VNCXOF    1-91      36-56
VNCXON    1-91      36-63
VNFCSH    1-134     69-68
VNGR      1-121     35-24
VNUIP     1-85      35-55
VNUMDC    1-85      37-53     37-65     37-72     69-90
VPAR5     1-135     23-69     23-71     46-49     46-53     56-34     56-70     56-97     72-35
VPAR6     1-134     37-27     73-56
VPLAS     1-140     29-18     29-26     29-36     29-47     69-114
VSLEDT    1-136     69-129
VSWPFL    1-101     24-134    29-16     35-121    69-123
VSWPSL    1-95      28-32     28-34*    28-45     35-123
VSYDMP    1-84      69-167
VU$CL     1-143     42-15
VUCLMC    1-143     42-17
VUXIFL    1-129      5-44      5-51
WINBAS    1-103      9-25*
WRKBUF    1-207#    24-239    24-243    25-20*    38-26     41-28     41-82     42-55     52-39     52-47     52-66     52-78
          53-43     56-28     56-62     59-35     59-50     59-57     66-16     68-23     72-29     73-53     75-64     79-38
WRKSIZ     1-208#   24-237    25-21*    79-40
XMVBAS     1-195#   12-47     12-72*    24-101*
ZCLR      1-109     17-21     17-22
```

Cross reference table (CREF V05.04)

```
...CM0   28-84   29-68   30-110  41-91   42-66   79-27   79-53
...CM1    6-29   24-243  28-50   28-62   28-70   29-30   29-47   29-55   30-85   30-94   30-99   38-21
         38-27   38-62   41-22   41-29   41-31   41-68   41-78   41-82   41-84   42-43   42-52   42-55
         42-60   52-25   52-39   54-25   56-21   56-29   56-63   59-36   59-50   60-35   66-17   68-24
         72-30   73-25   73-54   75-20   75-32   75-51   75-70   77-25
...CM2    6-29    6-29    6-29    6-29   23-19   23-23   23-61   23-63   23-78   23-176  24-221  24-243
         24-243  24-243  24-243  28-50   28-50   28-58   28-58   28-62   28-62   28-62   28-70   28-70
         28-70   28-70   29-30   29-30   29-43   29-43   29-47   29-47   29-47   29-55   29-55   29-55
         29-55   30-85   30-85   30-90   30-90   30-94   30-94   30-94   30-99   30-99   30-99   30-99
         38-21   38-21   38-27   38-27   38-27   38-27   38-62   38-62   41-22   41-22   41-29   41-29
         41-29   41-29   41-31   41-31   41-31   41-31   41-68   41-68   41-73   41-73   41-78   41-78
         41-78   41-82   41-82   41-82   41-82   41-84   41-84   42-43   42-43   42-48   42-48   42-52
         42-52   42-52   42-55   42-55   42-55   42-55   42-60   52-25   52-25   52-39   52-39   52-39
         52-39   54-25   54-25   54-25   54-25   56-21   56-21   56-29   56-29   56-29   56-29   56-63
         56-63   56-63   56-63   59-36   59-36   59-36   59-36   59-50   59-50   59-50   59-50   60-35
         60-35   60-35   60-35   66-17   66-17   66-17   66-17   68-24   68-24   68-24   68-24   72-30
         72-30   72-30   72-30   73-25   73-25   73-54   73-54   73-54   73-54   75-20   75-20   75-32
         75-32   75-32   75-32   75-51   75-51   75-51   75-51   75-70   75-70   75-70   75-70   77-25
         77-32   77-34   78-19   78-24   78-26   78-28
...CM3   28-57   28-72   29-42   29-56   30-89   30-101  41-72   41-83   42-47   42-65   50-76   56-111
         73-81   75-95
...CM5    6-29    6-45    6-46   23-19   23-23   23-31   23-32   23-33   23-36   23-54   23-55   23-61
         23-63   23-67   23-78   23-176  24-221  24-226  24-227  24-243  27-84   27-85   27-91   27-92
         27-96   27-97   27-101  28-50   28-58   28-62   28-70   28-84   28-95   28-96   28-102  28-103
         28-108  28-111  28-116  28-119  29-30   29-43   29-47   29-55   29-68   29-77   29-78   29-84
         29-85   30-60   30-61   30-85   30-90   30-94   30-99   30-110  30-131  30-132  30-138  30-139
         38-21   38-27   38-62   38-75   38-76   38-81   38-82   41-22   41-29   41-31   41-68   41-73
         41-78   41-82   41-84   41-91   41-101  41-102  41-108  41-109  42-43   42-48   42-52   42-55
         42-60   42-66   42-76   42-77   42-83   42-84   44-151  44-152  46-76   46-77   52-25   52-39
         54-25   56-21   56-29   56-63   58-6    58-7    58-10   59-36   59-50   60-35   66-17   66-51
         66-52   68-24   68-52   68-53   68-58   68-59   71-34   71-35   72-30   72-73   72-74   73-25
         73-54   73-93   73-94   73-99   73-104  73-105  74-78   74-79   75-20   75-22   75-23   75-32
         75-51   75-70   75-138  75-139  77-25   77-32   77-34   77-53   77-54   77-57   78-19   78-24
         78-26   78-28   79-27   79-53   80-20   80-21   81-18   82-26   83-26
...CM6   23-19   23-23   23-61   23-63   23-78   23-176  24-221  77-32   77-34   78-19   78-24   78-26
         78-28
...CM7    6-29   24-243  28-70   29-55   30-99   38-27   41-29   41-31   41-82   42-55   52-39   54-25
         56-29   56-63   59-36   59-50   60-35   66-17   68-24   72-30   73-54   75-32   75-51   75-70
.CLOSE    1-50#  28-57   28-72   29-42   29-56   30-89   30-101  41-83   42-65   50-76   56-111  73-81
         75-95
.CSTAT    1-53#  42-60
.DATE     1-52#  24-222
.DELET    1-51#  28-58   29-43   30-90   41-73   42-48
.DSTAT    1-52#  79-27
.ENTER    1-49#  28-62   29-47   30-94   41-78   42-52
.EXIT     1-51#   5-168   6-47
.FETCH    1-52#  79-53
.GTIM     1-52#  23-19   24-221
.GVAL     1-49#  23-78   23-176  77-32   77-34   78-19   78-24   78-26   78-28
.HERR     1-51#  23-57
.LOCK     1-52#  23-81
.LOOKU    1-49#  28-50   29-30   30-85   38-21   38-62   41-22   41-68   41-84   42-43   52-25   56-21
         73-25   75-20
.PRINT    1-50#   6-45    6-46   23-31   23-32   23-33   23-36   23-54   23-55   24-226  24-227  27-84
         27-85   27-91   27-92   27-96   27-97   27-101  28-95   28-96   28-102  28-103  28-108  28-111
         28-116  28-119  29-77   29-78   29-84   29-85   30-60   30-61   30-131  30-132  30-138  30-139
```

Cross reference table (CREF V05.04)

	38-75	38-76	38-81	38-82	41-101	41-102	41-108	41-109	42-76	42-77	42-83	42-84
	44-151	44-152	46-76	46-77	58-6	58-7	58-10	66-51	66-52	68-52	68-53	68-58
	68-59	71-34	71-35	72-73	72-74	73-93	73-94	73-99	73-104	73-105	74-78	74-79
	75-22	75-23	75-138	75-139	77-53	77-54	77-57	80-20	80-21			
.PURGE	1-50#	41-72	42-47									
.READW	1-49#	6-29	24-243	38-27	41-29	41-31	52-39	54-25	56-29	56-63	59-36	59-50
	60-35	66-17	68-24	72-30	73-54	75-32	75-51	75-70				
.RELEA	1-52#	28-84	29-68	30-110	41-91	42-66						
.SAVES	1-49#	77-25										
.SCCA	1-53#	23-23										
.SERR	1-51#	23-50										
.SETTO	1-50#	23-67										
.TRPSE	1-50#	23-61	23-63									
.TTYOU	1-50#	81-18	82-26	83-26								
.UNLOC	1-51#	24-232										
.WRITW	1-51#	28-70	29-55	30-99	41-82	42-55						
CLVX	3-53#	3-61	3-62	3-63	3-64	3-65	3-66	3-67				
DEFFLG	3-8#	3-14	3-15	3-16	3-17	3-18	3-19	3-20	3-21	3-22	3-23	3-24
	3-25	3-26	3-27	3-28	3-29	3-30	3-31	3-32	3-33	3-34	3-35	3-36
	3-37	3-38	3-39	3-40	3-41							
DISABL	1-161#	56-71	61-18	72-41	73-61							
ENABL	1-165#	56-85	61-65	72-55	73-73							
HANVER	2-9#	2-30	2-31	2-32								
OVLTBL	69-45#	69-51	69-52	69-53	69-54	69-55	69-56	69-57	69-58	69-59	69-60	69-61
	69-62	69-63										