TSINIT -- TSX startup initializ MACRO V05.04 Thursday 17-Dec-87 08:39
Table of contents

Table of contents

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

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

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

                ;
                ;  Global references
                ;
                        .GLOBL  HANDSK,MAXDEV,NDVRCB,HANRCB,HANRCO
                        .GLOBL  LXCL,VSYDMP,STKLVL,INTSSZ,INDFIL,NXIVMH,EXCBUF
                        .GLOBL  VNUIP,NSIP,INSTBL,INSTBN,II$$SZ,DCCSIZ,VNUMDC,NUMCDB
                        .GLOBL  NSCP,SCPFHD,SP$$SZ,CSHDEV,CSHDVN,VMXCSH,CD$$SZ
                        .GLOBL  LSTPL,LMXNUM,MXCSR,MXVEC,RSR,INVEC,VHIMEM,CXTPAG
                        .GLOBL  LSWPBK,LSTSL,SWDBLK,SWPCHN,NUMDEV,CS$NMX,SCHED
                        .GLOBL  H.DSTS,DVSTAT,HANENT,H.GEN,FORK,INTEN,PNAME
                        .GLOBL  $SXON,LSW10,LHIRBB,LHIRBE,LHIRBP,LHIRBG,LHIRBA
                        .GLOBL  LHIRBS,LHIRBC,VNCSLO,VNCXOF,VNCXON,SDDVU,VMSCHR,MAXSLO
                        .GLOBL  MPAR0,MPAR16,PARENL,MPARFL,TSEMT,VDBFLG,DX$EBA
                        .GLOBL  H.SIZ,HANSIZ,H.DVSZ,DEVSIZ,LOMAP,MMENBL,UPAR7
                        .GLOBL  PSW,HIMAP,FSTDL,LSTDL,LINBUF,LINSIZ,NUMCCB,TK1SEC
                        .GLOBL  FRKINI,FRKGEN,NUMFRK,FQ$$SZ,H.CSR,H.INS,VSWPSL,DMYDEV
                        .GLOBL  LINEND,LOTBUF,LOTSIZ,LOTEND,KMNTOP,KMNHI,NSL,NDL
                        .GLOBL  DX$MPH,DX$NHM,DX$IBH,HANPAR,HANXIT,MAPPAR,LINSPC
                        .GLOBL  KMNPGS,KMNSTK,KMNSTR,KMNCHN,SROMMR,KPAR0,PROFLG
                        .GLOBL  EMMAP,IOMAP,SR3MMR,IOPAGE,MAPSIZ,SR3FLG,NSPLDV
                        .GLOBL  UPAR0,KPDR0,UPDR0,KPAR7,BASMAP,PTWRD,PTBYT,LOKMEM
                        .GLOBL  GTBYT,MPPHY,RELOC,BRKPT,TSGEN,TSEXEC,VSWPFL
                        .GLOBL  CW$GDH,CW$BTH,CW$LGS,CW$FB,CW$FGJ,MSGBAS,RPRVEC
                        .GLOBL  CW$USR,CW$XM,CONFIG,CW$50H,JMP0,DTLX,USRBAS,WINBAS
                        .GLOBL  DATIML,DATIMH,RMON,CONFG2,SG$ELG,SG$IOT,CSHBAS
                        .GLOBL  SG$PAR,SG$MTS,SG$MMU,SG$MTM,LTTPAR,LOKBAS,CSHVEC,LOKVEC
                        .GLOBL  SYSGEN,AUTHAN,AHEND,CLKRTI,TRP4,CW$PRO,TIOVEC
                        .GLOBL  TRP10,TRP20,TRP24,EMTENT,TRP34,INIJMP,MHNSIZ
                        .GLOBL  TK1VAL,INRECV,OTRECV,INMXV,OTMXV,DHBFSZ,MXTYPE
                        .GLOBL  ZCLR,MXRBUF,MXDTR,INTMX1,$PHONE,LCDTYP,TIOBAS
                        .GLOBL  LDHB1B,LDHB1P,LDHB2B,CLVERS,CXTSIZ,CXTWDS,CXTPDR
                        .GLOBL  CLORSZ,TSXSIT,JM$$SZ,VMXMON,MONFQH,CXTRMN,CXTBAS
                        .GLOBL  ILSW2,$NOIN,LSW3,MXLPR,CW$ESP,CLTOTL,RMNPDR,MA$SYS
                        .GLOBL  SFCB,SFCBND,SFCBFH,SFCBSZ,NSPLFL,NSPLBL,INTSTK,INTSND
                        .GLOBL  NFRESB,PVSPBL,VMXWIN,DW$$SZ,LDVERS,CW$QBS
```

```
                        .GLOBL  FC$LBN,VMLBLK,VMXSF,VMXSFC,FF$$SZ,FW$$SZ,SWPJOB,SWPPOS
                        .GLOBL  TSR,RBR,RDINT,LSTMX,SS,CHAIN,JSWLOC,MU$TXT,SLTSIZ
                        .GLOBL  NUMIOQ,FREIOQ,UMODE,FPTRAP,MXLNT,DI$LD,DI$CL,CLSTS
                        .GLOBL  FREPGS,IOQSIZ,SYUNIT,UMSYTP,DI$TT,CXTBUF,SSEND
                        .GLOBL  SYINDX,MONVEC,KMNBAS,SDANAM,VBUSTP,MINCTR
                        .GLOBL  NUMRDB,RDB,RDBEND,RT$SKP,RT$TOP,NLINES,SHRRCB,SHRRCN
                        .GLOBL  RT$BAS,UPMODE,SPLNB,CSHALC,NIOL,CHNSIZ,RC$$SZ,VNGR
                        .GLOBL  UPAR6,UPDR6,RT$$SZ,VINABT,$DEAD,LSW6
                        .GLOBL  SYTIMH,SYSDAT,TRP250,ODTTRP,TRP14,SYTIML
                        .GLOBL  DS$ABT,CL$ORB,CL$ORE,CL$ORG,CL$ORP,CL$ORA
                        .GLOBL  $TAB,$FORM,CO$TAB,CO$FF,CO$DEF,CL$EPS,CLEOFS
                        .GLOBL  CL$OPT,CL$STA,CL$ORS,LSTLIN,VCSHNB,CL$EPP,CL$EPN
                        .GLOBL  CCLSAV,SPLND,SDCB,SPLDEV,SPLANM,MIODBG
                        .GLOBL  SDNAME,SDCHAN,SDCBSZ,SPLDVN,DTYPE
                        .GLOBL  DS$NRD,DX$NMT,$8BIT,CO$8BT,UEXRTN,VUXIFL
                        .GLOBL  SPLBLK,SPLCHN,MVSIZ,MEMPAR,UEXINT,DX$NRD
                        .GLOBL  NMSNMB,SNMSHD,SB$$SZ,PMSIZE,PMPAR
                        .GLOBL  NUMDCD,MEM256,LOKCSH,DC$$SZ
                        .GLOBL  JCXPGS,MXJMEM,VDFMEM,DFJMEM,TK5VAL,TK3SVL
                        .GLOBL  VPAR6,IOTIMR,ERRLOG,VNFCSH,FC$$SZ
                        .GLOBL  O.ADR,O.BLK,O.PAR,O.SIZ,VPAR5,KPAR5,DZOINT,DHOINT
                        .GLOBL  OVRADD,$OVRH,SYSMAP,MAPSYS,VSLEDT,LCLUNT
                        .GLOBL  UBUSMP,UMRADR,IOMAP,QBUS,UNIBUS,DX$NST
                        .GLOBL  DVFLAG,DX$DMA,RT$NAM,DS$DIR,LDDEVX,DS$VSZ
                        .GLOBL  INDSAV,INDDBL,INDTSV,INDDBS,DS$SFN
                        .GLOBL  SYNAME,UCLNAM,RSFBLK,VPLAS,SEGCHN
                        .GLOBL  MXJADR,$MEMSZ,PHYMEM,SG$TSX,CDX$DH
                        .GLOBL  CLHEAD,CLSIZE,CLDEVX,C.CSW,C.SBLK,C.DEVQ,C1DEVX
                        .GLOBL  VU$CL,VUCLMC,UK$$SZ,US$$SZ,UC$$SZ,UCLBLK,UCLDAT
                        .GLOBL  VLDSYS,VMXMSG,VMAXMC,MB$$SZ,MR$$SZ,CS$OPN,CS$ENT
                        .GLOBL  DX$MAP,MIOFLG,MI$SBP,MI$LNK,MIOBHD,VMIOSZ
                        .GLOBL  VMIOBF,MI$$SZ,MW$$SZ,MIONWB,MIOWHD,MW$LNK
                        .GLOBL  CSHSIZ,CSHBFP,CA$BLK,CA$DVU,CA$WCT,VMXMRB
                        .GLOBL  CA$UFL,CA$UBL,CA$HFL,CA$HBL,CA$HSH,NUMRDB
                        .GLOBL  SRTSIZ,SMRSIZ,CCBHD,CC$$SZ,CDX$DZ,MF$LIN
                        .GLOBL  CDX$DL,HF$TSB,MH$SCR,LMXLN,HF$LIN,HF$RIE,HF$TIE
                        .GLOBL  MH$LPR,DM$CSR,MF$LE,DM$LSR,HF$MC,MF$CS,MF$CM
                        .GLOBL  CDX$VH,VH$CSR,VH$LPR,MH$PBR,VF$TIE,VF$RIE,VF$MR
                        .GLOBL  VF$LIN,VF$SC,VF$RE,VH$LCR,VHOINT,TTINCP
                        .GLOBL  $HARD,LOUTIR,LINIR,NEDCHR,CLOTIR,CLINCP,FSTIOL,LSTHL
                        .GLOBL  SYSVER,SYSUPD,DI$DU,DI$XL,DI$MU,CL$LIX
                        .GLOBL  CL$LEN,DI$PI,GENTOP
                        .GLOBL  LSW5,DX$NCA,KPAR6,CLKVEC
                ;---------------------------------------------------------------------
                ;  Macros to enable and disable interrupts
                ;
                        .MACRO  DISABL          ;Disable interrupts
                        BIS     #340,@#PSW
                        .ENDM   DISABL

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

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

```
                        000040                  STA     =       40              ;PROGRAM START ADDRESS
                        000042                  STK     =       42              ;INITIAL STACK POINTER
                        000052                  RSZ     =       52              ;ROOT SIZE
                        000056                  OSZ     =       56              ;OVERLAY SIZE
                        000060                  RID     =       60              ;REL FILE ID
                        000062                  RBD     =       62              ;DISPLACEMENT TO 1ST REL BLOCK
                        001000                  ODTBAS  =       1000            ;BASE ADDRESS ODT WAS LINKED FOR
                                                ;
                                                ;  Data areas
                                                ;
000000                                  AREA:   .BLKW   8.
000020  000000  000000  000000          NFSBLK: .WORD   0,0,0,0,0,0             ;EXTENDED TO 6 WORDS FOR .CSTAT
000026  000000  000000  000000
000034  000000                          ODTFLG: .WORD   0
000036  000000                          ODTTOP: .WORD   0
000040  000000                          CCAFLG: .WORD   0
000042  000000                          CLK100: .WORD   0
000044  000000                          RTTRP4: .WORD   0
000046  000000                          RTMNVC: .WORD   0
000050                                  SAVBLK: .BLKW   5
000062  075250  100020  000000          TSXSAV: .RAD50  /SY TSX    SAV/
000070  073376
000072  075250  100003  051646          KMNNAM: .RAD50  /SY TSKMONSAV/
000100  073376
000102  075250  011504  000000          CCLNAM: .RAD50  /SY CCL    SAV/
000110  073376
000112  000000  000000  000000          DSTBLK: .WORD   0,0,0,0
000120  000000
000122  000000                          XMVBAS: .WORD   0
000124  000000                          NMXHAN: .WORD   0
000126  000000                          HMAP:   .WORD   0
000130  000000                          FETDEV: .WORD   0
000132  000000                          TOPMEM: .WORD   0
000134  000000                          FMEMHI: .WORD   0               ;64-byte block # below high alloc memory
000136  000000                          FMEMLO: .WORD   0               ;64-byte block # above top of low alloc memory
000140  000000                          OVLBAS: .WORD   0               ;Start loading overlays over TSINIT from here
000142  000000                          FILBLK: .WORD   0
000144  000000                          CURDEV: .WORD   0
000146  000000                          CURNAM: .WORD   0
000150  000000                          PROBUF: .WORD   0
000152  025556'                         WRKBUF: .WORD   INITOP
000154  004000                          WRKSIZ: .WORD   2048.
000156  052077                          R50MSG: .RAD50  /MSG/
000160  110466                          R50WIN: .RAD50  /WIN/
000162  046543                          R50LOK: .RAD50  /LOK/
000164  103112                          R50USR: .RAD50  /USR/
000166  012700                          R50CSH: .RAD50  /CSH/
000170  077167                          R50TIO: .RAD50  /TIO/
000172  100040                          R50TT:  .RAD50  /TT /
000174  075250                          R50SY:  .RAD50  /SY /
000176  045640                          R50LD:  .RAD50  /LD /
000200  062550                          R50PI:  .RAD50  /PI /
000202  012240                          R50CL:  .RAD50  /CL /
000204  012276                          R50CL0: .RAD50  /CL0/
000206  012305                          R50CL7: .RAD50  /CL7/
000210  013630                          R50C1:  .RAD50  /C1/
000212  013666                          R50C10: .RAD50  /C10/
```

```
                                R50C17: .RAD50  /C17/
                                R50VM:  .RAD50  /VM /
                                R50LS:  .RAD50  /LS /
                                R50ODT: .RAD50  /ODT/
                                SKPDEV: .RAD50  /TT DK SY CL C1 PI    /
                                GTLIN:  .BYTE   0,110
                                HANNAM: .RAD50  /SY XXX   TSX/
                                ODTBLK: .RAD50  /SY SYSODTREL/
                                INDNAM: .RAD50  /SY IND   SAV/
                                RLBF:   .WORD   0
                                RLBFND: .WORD   0
                                ODTSTA: .WORD   0
                                MEMLIM: .WORD   0
                                HGENFL: .WORD   0
                                ;
                                ;  Initialization configuration word
                                ;
                                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
                                ;
                                PISRT:  .RAD50  /SY PI    TSX/
                                        .WORD   0,0,0
                                ;
                                ;  Byte data cells
                                ;
                                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:
                                        DEFFLG  <CR>,<DX$MPH>
000342                                  DEFFLG  <CT>,<DX$MPH>
000346                                  DEFFLG  <DB>,<DX$DMA!DX$MPH>
000352                                  DEFFLG  <DD>,<DX$NHM>
000356                                  DEFFLG  <DL>,<DX$DMA!DX$MPH!DX$IBH>
000362                                  DEFFLG  <DM>,<DX$DMA!DX$NHM>
000366                                  DEFFLG  <DP>,<DX$DMA>
000372                                  DEFFLG  <DS>,<DX$DMA>
000376                                  DEFFLG  <DT>,<DX$DMA>
000402                                  DEFFLG  <DU>,<DX$DMA!DX$NHM!DX$NST>
000406                                  DEFFLG  <DW>,<DX$MPH>
000412                                  DEFFLG  <DX>,<DX$MPH>
000416                                  DEFFLG  <DY>,<DX$DMA!DX$NHM>
000422                                  DEFFLG  <DZ>,<DX$MPH>
000426                                  DEFFLG  <FW>,<DX$DMA>
000432                                  DEFFLG  <LP>,<DX$MPH>
000436                                  DEFFLG  <LS>,<DX$MPH>
000442                                  DEFFLG  <MM>,<DX$DMA!DX$MPH!DX$IBH>
000446                                  DEFFLG  <MS>,<DX$DMA!DX$MPH!DX$IBH>
000452                                  DEFFLG  <MT>,<DX$DMA!DX$MPH!DX$IBH>
000456                                  DEFFLG  <MU>,<DX$DMA!DX$NHM!DX$IBH!DX$NST>
000462                                  DEFFLG  <NL>,<DX$MPH>
000466                                  DEFFLG  <PC>,<DX$MPH>
000472                                  DEFFLG  <RF>,<DX$DMA>
000476                                  DEFFLG  <RK>,<DX$DMA!DX$MPH>
000502                                  DEFFLG  <VM>,<DX$EBA!DX$NCA!DX$NHM>
000506                                  DEFFLG  <XC>,<DX$MPH>
000512                                  DEFFLG  <XL>,<DX$MPH>
000516
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  004426'                 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
003560  105737  000000G                 TSTB    PROFLG          ;Are we running on a PRO?
003564  001403                          BEQ     7$              ;Br if not on a PRO
003566  004737  000000G                 CALL    PRONOP          ;Disable its interrupts
003572  000407                          BR      5$              ;Ignore unexpected interrupts on PRO
                                .ENDC   ;NE,PROASM
                                ;
                                ;  Set up vectors to catch unexpected interrupts
                                ;  Note: We encode the interrupt vector address in the PS --
                                ;  the low-order two bits of the address are dropped (they are
                                ;  always zero) and the remainder of the address is encoded in the
                                ;  PS fields priority (high-order 3 bits) and n-z-v-c (low-order 4 bits).
                                ;
003574  012702  000000G         7$:     MOV     #UEXINT,R2      ;SEND UNEXPECTED INTERRUPTS TO THIS ROUTINE
003600  012700  000044                  MOV     #44,R0          ;120 ENCODED IN PS FIELDS
003604  105737  000000G                 TSTB    VUXIFL          ;ARE WE TO IGNORE UNEXPECTED INTERRUPTS?
003610  001004                          BNE     10$             ;BR IF NOT
003612  012702  000000G         5$:     MOV     #UEXRTN,R2      ;ROUTINE TO GO TO TO IGNORE INTERRUPT
003616  012700  000340                  MOV     #340,R0         ;SET PRIO=7 IN PS
003622  012701  000120         10$:    MOV     #120,R1         ;INIT ALL VECTORS STARTING AT 120
003626  010221                  1$:     MOV     R2,(R1)+        ;SET PC FOR INTERRUPT
003630  010021                          MOV     R0,(R1)+        ;SET PS FOR INTERRUPT (ENCODED ADDRESS VALUE)
003632  105737  000000G         6$:     TSTB    VUXIFL          ;ARE WE TO IGNORE UNEXPECTED INTS?
003636  001411                          BEQ     2$              ;BR IF YES
003640  105737  000000G                 TSTB    PROFLG          ;IS THIS A PRO?
003644  001006                          BNE     2$              ;BR IF YES
003646  005200                          INC     R0              ;ADVANCE ENCODED ADDRESS
003650  032700  000020                  BIT     #20,R0          ;DID WE CARRY INTO "T"-FIELD?
003654  001402                          BEQ     2$              ;BR IF NOT
```

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

     58 003656  062700  000020                    ADD     #20,R0          ;FORCE CARRY OUT OF T-FIELD AND INTO PRIO FIELD
     59 003662  020127  000420          2$:       CMP     R1,#420         ;DONE ALL INTERRUPT VECTORS OF INTEREST?
     60 003666  103757                             BLO     1$              ;BR IF NOT
     61 003670  010237  000060                     MOV     R2,@#60         ;CATCH CONSOLE TERMINAL VECTOR TOO
     62 003674  012737  000014  000062             MOV     #14,@#62        ;ENCODED 60
     63 003702  010237  000064                     MOV     R2,@#64
     64 003706  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 003714  012700  000340          11$:      MOV     #340,R0         ;PRIORITY 7 PS
     71 003720  012737  000000G 000000G            MOV     #CLKRTI,@#CLKVEC;Send clock interrupt to RTI instruct for now
     72 003726  010037  000002G                    MOV     R0,@#CLKVEC+2
     73                                     ;
     74                                     ;  Take over traps, EMT, BPT, etc.
     75                                     ;
     76 003732  005001                             CLR     R1              ;Start at location 0
     77 003734  012721  000137                     MOV     #137,(R1)+      ;[JMP @#JMP0] ==> 0
     78 003740  012721  000000G                    MOV     #JMP0,(R1)+     ;CATCH JUMPS TO LOCATION 0
     79 003744  012721  000000G                    MOV     #TRP4,(R1)+     ;TRAP 4
     80 003750  005021                             CLR     (R1)+
     81 003752  012721  000000G                    MOV     #TRP10,(R1)+    ;TRAP 10
     82 003756  005021                             CLR     (R1)+
     83 003760  012721  000000G                    MOV     #TRP14,(R1)+    ;TRAP 14 (BREAKPOINTS)
     84 003764  010021                             MOV     R0,(R1)+
     85 003766  012721  000000G                    MOV     #TRP20,(R1)+    ;IOT TRAP
     86 003772  005021                             CLR     (R1)+
     87 003774  012721  000000G                    MOV     #TRP24,(R1)+    ;POWER FAIL
     88 004000  010021                             MOV     R0,(R1)+
     89 004002  012721  000000G                    MOV     #EMTENT,(R1)+   ;EMT
     90 004006  005021                             CLR     (R1)+
     91 004010  012721  000000G                    MOV     #TRP34,(R1)+    ;TRAP
     92 004014  005021                             CLR     (R1)+
     93 004016  012737  000000G 000114             MOV     #MEMPAR,@#114   ;MEMORY PARITY TRAP
     94 004024  010037  000116                     MOV     R0,@#116
     95 004030  012737  000000G 000244             MOV     #FPTRAP,@#244   ;TRAP 244 -- FLOATING POINT TRAP
     96 004036  010037  000246                     MOV     R0,@#246        ;Enter FPU trap at priority 7
     97 004042  012737  000000G 000250             MOV     #TRP250,@#250   ;TRAP 250 -- MEMORY MANAGEMENT TRAP
     98 004050  005037  000252                     CLR     @#252
     99                                     ;
    100                                     ;  Initialize the system mapped region.
    101                                     ;
    102 004054  010546                             MOV     R5,-(SP)        ;SAVE THE CURRENT CONTENTS OF R5
    103 004056  012705  000006                     MOV     #6,R5           ;INITIALIZE TO THE FIRST REGION
    104 004062  004737  000000G                    CALL    MAPSYS          ;CALL THE SYSTEM MAPPING ROUTINE
    105 004066  012605                             MOV     (SP)+,R5        ;RESTORE THE PREVIOUS CONTENTS OF R5
    106                                     ;
    107                                     ;  Set up Unibus mapping if needed
    108                                     ;
    109 004070  004737  004764'                    CALL    SETUMP          ;SET UP UNIBUS MAPPING
    110                                     ;
    111                                     ;  Initialize time-sharing lines.
    112                                     ;
    113 004074  004737  005376'                    CALL    LININI          ;INIT LINES & SET UP VECTORS
    114                                     ;
```

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

```
                                        ;  Enable memory management
                                        ;  (The kernel-mode mapping registers are already set up)
                                        ;
004100  052737  0000000G 0000000G               BIS     #MMENBL,@#SROMMR;Turn on memory management
004106  105737  0000000G                         TSTB    SR3FLG          ;Does machine have memory management reg 3?
004112  001415                                  BEQ     4$              ;Br if register does not exist (no ext. mem.)
004114  023727  0000000G 010000                 CMP     PHYMEM,#4096.   ;Does machine have at least 256Kb phys memory?
004122  103411                                  BLO     4$              ;Br if not
004124  052737  0000000G 0000000G               BIS     #EMMAP,@#SR3MMR ;Set extended memory on
004132  105737  0000000G                         TSTB    MEM256          ;Will TSX-Plus use at least 256Kb?
004136  001403                                  BEQ     4$              ;Br if not
004140  052737  0000000G 0000000G               BIS     #IOMAP,@#SR3MMR ;Turn on 22-bit memory management for I/O
                                        ;
                                        ;  Initialize the memory allocation table
                                        ;
004146  013737  0000000G 0000000G 4$:   MOV     MAPPAR,@#KPAR5  ;Map to memory allocation table
004154  013702  0000000G                MOV     LOMAP,R2        ;Point to 1st user-page entry
004160  105022                          8$:     CLRB    (R2)+           ;Say page is free
004162  020237  0000000G                         CMP     R2,HIMAP        ;Done all user pages?
004166  103774                                  BLO     8$              ;Loop if not
004170  112712  0000000G                         MOVB    #MA$SYS,(R2)    ;Set flag marking start of system pages
                                        ;
                                        ;  Set up I/O device interrupt vectors.
                                        ;
004174  004737  004766'                         CALL    DEVVEC          ;SET UP DEVICE INTERRUPT VECTORS
                                        ;
                                        ;  If we are running on a Professional, initiailize the PI handler
                                        ;
                                .IF     NE,PROASM
004200  105737  0000000G                         TSTB    PROFLG          ;Are we running on a Professional?
004204  001404                                  BEQ     3$              ;Br if not
004206  004737  0000000G                         CALL    PROHAN          ;Initialize the PI handler
004212  004737  005314'                         CALL    PIDVEN          ;Make device table entry for PI
                                .ENDC   ;NE,PROASM
                                        ;
                                        ;  Initialize interrupt stack area
                                        ;
004216  013702  0000000G                3$:     MOV     INTSND,R2       ;Point to base of stack area
004222  012700  123456                          MOV     #123456,R0      ;Get initialization value
004226  010022                          12$:    MOV     R0,(R2)+        ;Initialize the interrupt stack area
004230  020237  0000000G                         CMP     R2,INTSTK       ;Finished?
004234  103774                                  BLO     12$             ;Loop if not
004236  112737  177777  0000000G               MOVB    #-1,STKLVL      ;Say we are not running on interrupt stack
                                        ;
                                        ; Enter TSEXEC to complete initialization
                                        ;
004244  000137  0000000G                         JMP     INIJMP          ;ENTER INITIALIZATION ROUTINE IN TSEXEC
                                        ;
                                        ;  Abort the initialization
                                        ;
004250  013737  000042' 0000000G INISTP: MOV     CLK100,@#CLKVEC ;Restore RT-11 clock vector
004256  013737  000044' 000004           MOV     RTTRP4,@#4      ;Restore trap 4 vector
004264  013737  000046' 0000000G         MOV     RTMNVC,@#RMON   ;Restore RT-11 monitor pointer
004272                                  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.
                                ;
004274  010146          LODINI: MOV     R1,-(SP)
004276  010346                  MOV     R3,-(SP)
004300  010446                  MOV     R4,-(SP)
                                ;
                                ;  Get pointer to linker-built overlay entry
                                ;
004302  016201  000004          MOV     OS$OVL(R2),R1   ;Get pointer to linker-built table
                                ;
                                ;  Determine how much code to read from the segment
                                ;
004306  016204  000000          MOV     OS$SIZ(R2),R4   ;Get # 64-byte blks allocated for segment
004312  072427  000005          ASH     #5,R4           ;Convert to # words
004316  020461  000000G         CMP     R4,O.SIZ(R1)    ;Compare with original segment code size
004322  101402                  BLOS    1$              ;Br if segment was truncated by init
004324  016104  000000G         MOV     O.SIZ(R1),R4    ;Get code size
                                ;
                                ;  Read the segment into memory
                                ;
004330  010503          1$:     MOV     R5,R3           ;Get 64-byte block #
004332  072327  000006          ASH     #6,R3           ;Convert to byte address
004336                          .READW  #AREA,#17,R3,R4,O.BLK(R1)
004372  103406                  BCS     10$             ;Br if error on read
                                ;
                                ;  Store the physical address of the segment into the overlay descriptor
                                ;
004374  010561  000000G         MOV     R5,O.PAR(R1)    ;Remember physical address of segment
                                ;
                                ;  Finished
                                ;
004400  012604                  MOV     (SP)+,R4
004402  012603                  MOV     (SP)+,R3
004404  012601                  MOV     (SP)+,R1
004406  000207                  RETURN
                                ;
                                ;  Error on read
                                ;
004410                  10$:    .PRINT  #TSXHD
004416                          .PRINT  #RDERR
004424                          .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.
                                ;
004426  010246                  INIOVL: MOV     R2,-(SP)
004430  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize pointer to start of memory area for overlays
                                ;
004432  013705  000140'                 MOV     OVLBAS,R5       ;Start of area for overlays
004436  072527  177772                  ASH     #-6,R5          ;Convert to 64-byte #
004442  042705  176000                  BIC     #176000,R5      ;Clear possible propagated sign bits
                                ;
                                ;  Begin loop to load each overlay that goes over TSINIT
                                ;
004446  012702  000576'                 MOV     #OSTABL,R2      ;Point to 1st overlay segment entry
004452  005762  000002          1$:     TST     OS$FLG(R2)      ;Does this segment go over TSINIT?
004456  001406                          BEQ     2$              ;Br if not
004460  004737  004636'                 CALL    KEYSEG          ;Remember base of some segments
004464  004737  004274'                 CALL    LODINI          ;Load the segment
004470  066205  000000                  ADD     OS$SIZ(R2),R5   ;Advance memory pointer
004474  062702  000006          2$:     ADD     #OS$$SZ,R2      ;Point to entry for next segment
004500  020237  001024'                 CMP     R2,OSLAST       ;Finished all segments?
004504  103762                          BLO     1$              ;Loop if not
                                ;
                                ;  Initialize entry point vector for TSTIOX segment
                                ;
004506  013702  000000G                 MOV     TIOBAS,R2       ;Get addr of base of TSTIOX
004512  072227  000006                  ASH     #6,R2           ;Convert to byte address
004516  012705  000000G                 MOV     #TIOVEC,R5      ;Point to entry point vector
004522  004737  004600'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Initialize entry point vector for TSCASH segment
                                ;
004526  013702  000000G                 MOV     CSHBAS,R2       ;Get addr of base of TSCASH
004532  001406                          BEQ     3$              ;Br if TSCASH not loaded
004534  072227  000006                  ASH     #6,R2           ;Convert to byte address
004540  012705  000000G                 MOV     #CSHVEC,R5      ;Point to entry point vector
004544  004737  004600'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Initialize entry point vector for TSLOCK segment
                                ;
004550  013702  000000G         3$:     MOV     LOKBAS,R2       ;Get addr of base of TSLOCK
004554  001406                          BEQ     9$              ;Br if TSLOCK not loaded
004556  072227  000006                  ASH     #6,R2           ;Convert to byte address
004562  012705  000000G                 MOV     #LOKVEC,R5      ;Point to entry point vector
004566  004737  004600'                 CALL    ENTVEC          ;Set up entry point vector
                                ;
                                ;  Finished
                                ;
004572  012605                  9$:     MOV     (SP)+,R5
004574  012602                          MOV     (SP)+,R2
004576  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)
                                ;
004600  010246                  ENTVEC: MOV     R2,-(SP)
004602  010446                          MOV     R4,-(SP)
004604  010546                          MOV     R5,-(SP)
004606  010204                          MOV     R2,R4           ;Get addr of base of segment
004610  062704  000004                  ADD     #4,R4           ;Point to start of vector in segment
004614  005722                          TST     (R2)+           ;Get value to use to relocate offsets
004616  012415                  1$:     MOV     (R4)+,(R5)      ;Get offset to entry point within segment
004620  060225                          ADD     R2,(R5)+        ;Convert to absolute address
004622  005715                          TST     (R5)            ;Any more words to initialize?
004624  001774                          BEQ     1$              ;Br if yes
                                ;
                                ;  Finished
                                ;
004626  012605                          MOV     (SP)+,R5
004630  012604                          MOV     (SP)+,R4
004632  012602                          MOV     (SP)+,R2
004634  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.
                                ;
004636  010446                  KEYSEG: MOV     R4,-(SP)
                                ;
                                ;  Get the name of the segment out of the linker-built segment block
                                ;
004640  016200  000004                  MOV     OS$OVL(R2),R0   ;Point to linker-built entry
004644  016004  000000G                 MOV     O.ADR(R0),R4    ;Get the name of the segment
                                ;
                                ;  See if this is a segment whose address we want to remember
                                ;
004650  020437  000156'                 CMP     R4,R50MSG       ;Is this the TSMSG segment?
004654  001003                          BNE     1$              ;Br if not
004656  010537  000000G                 MOV     R5,MSGBAS       ;Remember base of TSMSG segment
004662  000436                          BR      8$
004664  020437  000160'          1$:     CMP     R4,R50WIN       ;Is this the TSWIN segment?
004670  001003                          BNE     3$              ;Br if not
004672  010537  000000G                 MOV     R5,WINBAS       ;Remember base of TSWIN segment
004676  000430                          BR      8$
004700  020437  000164'          3$:     CMP     R4,R50USR       ;Is this the TSUSR segment?
004704  001003                          BNE     4$              ;Br if not
004706  010537  000000G                 MOV     R5,USRBAS       ;Remember base of TSUSR segment
004712  000422                          BR      8$
004714  020437  000162'          4$:     CMP     R4,R50LOK       ;Is this the TSLOCK segment?
004720  001003                          BNE     5$              ;Br if not
004722  010537  000000G                 MOV     R5,LOKBAS       ;Remember base of TSLOCK segment
004726  000414                          BR      8$
004730  020437  000166'          5$:     CMP     R4,R50CSH       ;Is this the TSCASH segment?
004734  001003                          BNE     6$              ;Br if not
004736  010537  000000G                 MOV     R5,CSHBAS       ;Remember base of TSCASH segment
004742  000406                          BR      8$
004744  020437  000170'          6$:     CMP     R4,R50TIO       ;Is this the TSTIOX segment?
004750  001003                          BNE     8$              ;Br if not
004752  010537  000000G                 MOV     R5,TIOBAS       ;Remember base of module
004756  000400                          BR      8$
                                ;
                                ;  Finished
                                ;
004760  012604                  8$:     MOV     (SP)+,R4
004762  000207                          RETURN
```

```
                        .IF     NE,<PROASM-1>   ;Assemble for PDP-11
                        .SBTTL  SETUMP -- Set up Unibus mapping if needed
                ;-----------------------------------------------------------------------
                ;  SETUMP is called to set up the Unibus map registers for 11/44 and
                ;  11/70 systems which more than 256Kb of memory.
                ;  If Unibus mapping is needed, the Unibus map registers # 0-4 are
                ;  initialized for a 1-to-1 mapping with the low 40Kb of memory
                ;  so that I/O to system buffers in the low memory area can be done without
                ;  having to do Unibus mapping.
                ;
                ;  Outputs:
                ;   UBUSMP:   1==>Do Unibus mapping;  0==>Don't do Unibus mapping.
                ;
                SETUMP: MOV     R2,-(SP)
                        MOV     R3,-(SP)
                        MOV     @#4,-(SP)       ;SAVE TRAP VECTOR
                        MOV     #9$,@#4         ;CATCH TRAPS
                ;
                ;  See if this is a type of maching that needs unibus mapping
                ;
                        TSTB    UBUSMP          ;Is UNIBUS mapping needed?
                        BEQ     9$              ;Br if not
                ;
                ;  Unibus mapping is needed
                ;  Load unibus map registers # 0-4 to point to low 48Kb of memory.
                ;
                2$:     MOV     #UMRADR,R5      ;POINT TO CONTROL REGISTER FOR UNIBUS MAP 0
                        CLR     R4              ;START MAPPING TO BOTTOM OF MEMORY
                        MOV     #5,R0           ;LOAD 5 MAP REGISTERS
                1$:     MOV     R4,(R5)+        ;SET LOW-ORDER VALUE IN MAP REGISTER
                        CLR     (R5)+           ;CLEAR HIGH-ORDER VALUE IN MAP REGISTER
                        ADD     #8192.,R4       ;ADVANCE MEMORY ADDRESS
                        SOB     R0,1$           ;LOOP IF MORE MAP REGISTERS TO LOAD
                ;
                ;  Turn on Unibus mapping
                ;
                        BIS     #IOMAP,@#SR3MMR ;ENABLE UNIBUS MAPPING
                ;
                ;  Finished
                ;
                9$:     MOV     (SP)+,@#4       ;RESTORE TRAP VECTOR
                        MOV     (SP)+,R5
                        MOV     (SP)+,R4
                        RETURN
                        .IFF    ;NE,<PROASM-1>  ;Following code for Pro-only assembly
                ;
                ;  Define dummy SETUMP routine for Pro
                ;
004764  000207  SETUMP: RETURN
                        .ENDC   ;NE,<PROASM-1>
```

```
                                        .SBTTL  DEVVEC -- Set up device vectors
                                ;-----------------------------------------------------------------------
                                ;  DEVVEC is called to set up device interrupt vectors for handlers
                                ;  that have been loaded.
                                ;
004766  010146                  DEVVEC: MOV     R1,-(SP)
004770  010346                          MOV     R3,-(SP)
004772  010546                          MOV     R5,-(SP)
004774  013746  000000G                 MOV     @#KPAR5,-(SP)   ;Save PAR 5 mapping
                                ;
                                ;  Begin loop to set up vectors for each device
                                ;
005000  012701  000002                  MOV     #2,R1           ;Get index # of 1st device after TT
005004  016103  000000G         1$:     MOV     HANENT(R1),R3   ;Get handler entry point address
005010  020327  000006                  CMP     R3,#6           ;Is this a real device?
005014  101436                          BLOS    6$              ;Br if not
                                ;
                                ;  See if we need to map PAR 5 to this handler
                                ;
005016  016100  000000G                 MOV     HANPAR(R1),R0   ;Get PAR 5 base for this handler
005022  001402                          BEQ     2$              ;Br if this is not a mapped handler
005024  010037  000000G                 MOV     R0,@#KPAR5      ;Map PAR 5 to the handler
                                ;
                                ;  Clear CQE and LQE in handler header
                                ;
005030  005023                  2$:     CLR     (R3)+           ;Clear LQE (4th word in handler)
005032  005013                          CLR     (R3)            ;Clear CQE (5th word in handler)
005034  162703  000010                  SUB     #10,R3          ;Point to 1st word of handler
                                ;
                                ;  Set up interrupt vectors for this handler
                                ;
005040  005005                          CLR     R5              ;Assume vector base address is 0
005042  005713                          TST     (R3)            ;Any vectors to set up?
005044  001422                          BEQ     6$              ;Br if no vectors to set up
005046  002403                          BLT     5$              ;Br if multiple-vector
005050  004737  005140'                 CALL    SETVEC          ;Set up the vector
005054  000416                          BR      6$              ;Finished
                                ;
                                ;  Multiple vectors.
                                ;
005056  012300                  5$:     MOV     (R3)+,R0        ;Get offset to list of vector info
005060  006300                          ASL     R0              ;Get byte offset to vector list
005062  060003                          ADD     R0,R3           ;Get absolute address of vector table
005064  005713                          TST     (R3)            ;Is this a PRO device with floating vectors?
005066  002005                          BGE     7$              ;Br if not
005070  005723                          TST     (R3)+           ;Point to word with device ID
005072  012346                          MOV     (R3)+,-(SP)     ;Get device ID
005074  004777  000000G                 CALL    @RPRVEC         ;Get base vector location for device
005100  012605                          MOV     (SP)+,R5        ;This is base of vector locations
005102  004737  005140'         7$:     CALL    SETVEC          ;Set up the vector
005106  005713                          TST     (R3)            ;Any more vectors to set up?
005110  003374                          BGT     7$              ;Br if yes
                                ;
                                ;  See if there are more devices to set up.
                                ;
005112  062701  000002          6$:     ADD     #2,R1           ;Advance device table index
005116  020137  000000G                 CMP     R1,NUMDEV       ;More to do?
```

```
     58 005122  101730                          BLOS    1$              ;Br if yes
     59                                ;
     60                                ;  Finished
     61                                ;
     62 005124  012637  000000G                 MOV     (SP)+,@#KPAR5
     63 005130  012605                          MOV     (SP)+,R5
     64 005132  012603                          MOV     (SP)+,R3
     65 005134  012601                          MOV     (SP)+,R1
     66 005136  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
                                ;
005140  010446                  SETVEC: MOV     R4,-(SP)
                                ;
                                ;  See if this is a mapped handler
                                ;
005142  005761  0000006             TST     HANPAR(R1)      ;Is this a mapped handler
005146  001010                      BNE     1$              ;Br if yes
                                ;
                                ;  This is an unmapped handler.
                                ;  Vector interrupts directly to the handler.
                                ;
005150  012300                      MOV     (R3)+,R0        ;Get address of interrupt vector
005152  060500                      ADD     R5,R0           ;Add base address to vector location
005154  010310                      MOV     R3,(R0)         ;Store address of cell in handler
005156  062320                      ADD     (R3)+,(R0)+     ;Add offset to interrupt entry point
005160  012310                      MOV     (R3)+,(R0)      ;Set PS for interrupt
005162  052710  000340              BIS     #340,(R0)       ;Make sure priority = 7
005166  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.
                                ;
005170  013704  000122'         1$:     MOV     XMVBAS,R4       ;Point to area where we store interrupt rtn
005174  012300                          MOV     (R3)+,R0        ;Get address of interrupt vector
005176  060500                          ADD     R5,R0           ;Add base address to interrupt location
005200  010420                          MOV     R4,(R0)+        ;Direct interrupt to our routine
005202  012724  013746                  MOV     #013746,(R4)+   ; [ MOV @#KPAR5,-(SP) ]
005206  012724  0000006                 MOV     #KPAR5,(R4)+
005212  012724  012737                  MOV     #012737,(R4)+   ; [ MOV #par5val,@#KPAR5 ]
005216  016124  0000006                 MOV     HANPAR(R1),(R4)+
005222  012724  0000006                 MOV     #KPAR5,(R4)+
005226  012724  012746                  MOV     #012746,(R4)+   ; [ MOV #340,-(SP) ]
005232  012724  000340                  MOV     #340,(R4)+
```

```
                                .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.
                                ;
005314  010146                  PIDVEN: MOV     R1,-(SP)
                                ;
                                ;  Increase number of defined devices and get device table entry index
                                ;  to use for the PI device.
                                ;
005316  062737  000002  000000G       ADD     #2,NUMDEV       ;One more device
005324  013701  000000G               MOV     NUMDEV,R1       ;Get device table index
                                ;
                                ;  Set up information about the PI device
                                ;
005330  013761  000200' 000000G       MOV     R50PI,PNAME(R1) ;Set device name
005336  012761  000000C 000000G       MOV     #<DS$SFN!DI$PI>,DVSTAT(R1) ;Set device status flags
005344  005061  000000G               CLR     DVFLAG(R1)      ;Clear other flags
005350  005061  000000G               CLR     DEVSIZ(R1)      ;Clear device size
005354  012761  000006G 000000G       MOV     #PIHAN+6,HANENT(R1);Set handler entry point (4th word)
005362  012761  000000G 000000G       MOV     #PROSIZ,HANSIZ(R1) ;Set handler size
                                ;
                                ;  Finished
                                ;
005370  012601                       MOV     (SP)+,R1
005372  000207                       RETURN
                                .ENDC   ;NE,PROASM
```

```
                        .IF     NE,<PROASM-1>   ;If assembling for PDP-11
                        .SBTTL  LININI -- Initialize time-sharing lines
                ;-----------------------------------------------------------------------
                ;  LININI is called to initialize the time-sharing lines.
                ;  This consists of setting up interrupt vectors and setting control
                ;  flags in the status registers.
                ;
                LININI: MOV     R1,-(SP)
                        MOV     R2,-(SP)
                        MOV     R3,-(SP)
                        MOV     R4,-(SP)
                        MOV     R5,-(SP)
                ;
                ;  Set up interrupt vectors for DL11 lines
                ;
                        MOV     #2,R1           ;Index for 1st line
                        MOV     #340,R4         ;Priority 7 PS
                1$:     BIT     #$DEAD,LSW3(R1) ;Is this line uninstalled?
                        BNE     8$              ;Br if yes
                        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
                        BEQ     8$              ;Br if not
                        CMP     LCDTYP(R1),#CDX$DL ;Is this a DL11 line?
                        BNE     8$              ;Br if not
                ;
                ;  DL-11 line
                ;
                        MOV     INVEC(R1),R5    ;GET ADDRESS OF INPUT VECTOR
                        MOV     #INRECV,R2      ;END OF RECEIVING VECTOR
                        MOV     #<OTRECV-6>,R3  ;START OF INPUT INTERRUPT ENTRY POINTS
                        MOV     R1,R0           ;GET LINE NUMBER
                        ASL     R0              ;4 BYTES PER INPUT INTERRUPT ENTRY POINT
                        SUB     R0,R2           ;GET ADDRESS OF INPUT INTERRUPT ENTRY POINT
                        ADD     R1,R0           ;6 BYTES PER OUTPUT INTERRUPT ENTRY POINT
                        ADD     R0,R3           ;GET ADDRESS OF OUTPUT INTERRUPT ENTRY POINT
                        MOV     R2,(R5)+        ;SET PC FOR INPUT INTERRUPT ENTRY POINT
                        MOV     R4,(R5)+        ;SET PS FOR INPUT INTERRUPT
                        MOV     R3,(R5)+        ;SET PC FOR OUTPUT INTERRUPT
                        MOV     R4,(R5)+        ;SET PS FOR OUTPUT INTERRUPT
                ;
                ;  Try next line
                ;
                8$:     ADD     #2,R1           ;ADVANCE LINE INDEX NUMBER
                        CMP     R1,#LSTHL       ;MORE TO DO?
                        BLOS    1$              ;BR IF YES
                ;
                ;  Initialize multiplexers.
                ;
                SETMUX: MOV     #LSTMX,R1       ;Get last mux index #
                        BEQ     SETLIN          ;Br if there are no mux lines
                3$:     CMP     MXTYPE(R1),#CDX$DZ ;Is this a DZ11, DH11, or DHV11?
                        BEQ     1$              ;Br if DZ11
                        CMP     MXTYPE(R1),#CDX$VH ;Is this a DHV11?
                        BNE     2$              ;Br if not
                        CALL    VHINIT          ;Initialize a DHV11
                        BR      4$
                2$:     CALL    DHINIT          ;Initialize a DH11
                        BR      4$
```

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

```
    115                                         MOV     (SP)+,R5
    116                                         MOV     (SP)+,R4
    117                                         MOV     (SP)+,R3
    118                                         MOV     (SP)+,R2
    119                                         MOV     (SP)+,R1
    120                                         RETURN
```

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

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

```
                        .SBTTL  DZINIT -- Initialize a DZ11 multiplexer
                ;-----------------------------------------------------------------------
                ;  DZINIT is called to initialize a DZ11 multiplexer.
                ;
                ;  Inputs:
                ;   R1 = Mux index number.
                ;
                DZINIT:
                ;
                ;  See if this DZ11 is installed
                ;
                        TST     MXCSR(R1)       ;Is this DZ-11 installed?
                        BEQ     4$              ;Br if not
                ;
                ;  Set up interrupt vector connections for this MUX
                ;
                        CALL    MUXVEC          ;Set up interrupt vectors for this DZ11
                ;
                ;  Start up the mux operation
                ;
                        BIS     #ZCLR,@MXCSR(R1);Do master clear on DZ-11
                1$:     BIT     #ZCLR,@MXCSR(R1);Wait for clear to finish
                        BNE     1$
                2$:     MOV     @MXRBUF(R1),R0  ;Clear silo
                        BMI     2$
                        CLRB    @MXDTR(R1)      ;Disable all data sets
                ;
                ;  Finished
                ;
                4$:     RETURN
```

```
                        .SBTTL  MUXVEC -- Set up interrupt vectors for a multiplexer
                ;-----------------------------------------------------------------------
                ;  MUXVEC is called to set up the interrupt vector connections for
                ;  a DZ11, DH11, or DHV11 multiplexer.
                ;
                ;  Inputs:
                ;   R1 = Mux index number.
                ;
MUXVEC: MOV     R2,-(SP)
        MOV     R3,-(SP)
        MOV     R5,-(SP)
                ;
                ;  Set interrupt vector for mux
                ;
        MOV     MXVEC(R1),R5    ;Get address of input interrupt vector
        MOV     #INMXV,R2       ;End of receiving vector
        MOV     #<OTMXV-6>,R3   ;Output interrupt table
        MOV     R1,R0           ;Get mux index number
        ASL     R0              ;4 bytes per line in input int table
        SUB     R0,R2           ;Get address of input entry point
        ADD     R1,R0           ;6 bytes per mux in output entry point table
        ADD     R0,R3           ;Get address of output int entry point
        MOV     R2,(R5)+        ;Set PC for input interrupt
        MOV     #340,(R5)+      ;Set PS for output interrupt
        MOV     R3,(R5)+        ;Set PC for output interrupt
        MOV     #340,(R5)       ;Set PS for output interrupt
                ;
                ;  Now store an instruction sequence of the form:
                ;
                ;       JSR     R5,@#interrupt_routine
                ;       .WORD   mux_index
                ;
                ;  to catch mux output interrupts and vector them to the interrupt routine.
                ;
        MOV     #004537,(R3)+   ;JSR R5,@#x
        MOV     #DZOINT,R0      ;Assume this is a DZ11
        CMP     MXTYPE(R1),#CDX$DZ ;Is this a DZ11?
        BEQ     1$              ;Br if yes
        MOV     #VHOINT,R0      ;Assume this is a DHV11
        CMP     MXTYPE(R1),#CDX$VH ;Is this a DHV11?
        BEQ     1$              ;Br if yes
        MOV     #DHOINT,R0      ;Get interrupt routine for DH11's
1$:     MOV     R0,(R3)+        ;Store address of interrupt routine
        MOV     R1,(R3)         ;Store mux index number
                ;
                ;  Finished
                ;
9$:     MOV     (SP)+,R5
        MOV     (SP)+,R3
        MOV     (SP)+,R2
        RETURN
```

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

```
                        .SBTTL  VHINIT -- Initialize a DHV11 multiplexer
                ;-----------------------------------------------------------------------
                ;  Perform initialization for a DHV11 mux.
                ;
                ;  Inputs:
                ;   R1 = Mux index number.
                ;
                VHINIT:
                ;
                ;  See if this DHV11 is installed
                ;
                        TST     VH$CSR(R1)      ;Is this DHV11 installed?
                        BEQ     9$              ;Br if not
                ;
                ;  Connect interrupt vector to DHV11
                ;
                        CALL    MUXVEC          ;Set up interrupt vectors
                ;
                ;  Clear the multiplexer
                ;
                        MOV     #VF$MR,@VH$CSR(R1) ;Reset the multiplexer
                1$:     BIT     #VF$MR,@VH$CSR(R1) ;Wait for reset to finish
                        BNE     1$
                ;
                ;  Clean out the FIFO buffer in the mux
                ;
                2$:     MOV     @MXRBUF(R1),R0  ;Get contents of receiver buffer register
                        BLT     2$              ;Loop until RBUF empty
                ;
                ;  Finished
                ;
                9$:     RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  End of code that can be omitted for Pro-only systems
                                ;
                                        .IFF    ;NE,<PROASM-1>  ;Begin code for Pro only
                                ;
                                ;  This code is assembled only for Pro systems.
                                ;  T/S line init routines for Pro only.
                                ;
005374                          LINCHK:
005374                          DHLPRM:
005374                          VHLPRM:
005374                          DZINIT:
005374                          MUXVEC:
005374                          DHINIT:
005374                          VHINIT:
005374  000207                          RETURN
                                ;
                                ;  LININI routine for Pro systems
                                ;
005376  010146                  LININI: MOV     R1,-(SP)
005400  012701  0000000                 MOV     #LSTLIN,R1      ;Get # of last line
                                ;
                                ;  Determine if this line is connected to hardware
                                ;
005404  004737  005454'         1$:     CALL    LINTYP          ;Determine the type of this line
005410  032761  0000000 0000000        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
005416  001411                          BEQ     2$              ;Br if not
                                ;
                                ;  Call Pro line initialization routine
                                ;
005420  004737  0000000               CALL    PROLIN          ;Initialize Pro line
                                ;
                                ;  Do some special init for phone lines
                                ;
005424  032761  0000000 0000000       BIT     #$PHONE,ILSW2(R1);Is this a dialup line?
005432  001403                        BEQ     2$              ;Br if not
005434  052761  0000000 0000000       BIS     #$NOIN,LSW3(R1) ;Ignore input till dial up occurs
                                ;
                                ;  Check next line
                                ;
005442  162701  000002          2$:    SUB     #2,R1           ;Get index # of next line
005446  003356                        BGT     1$              ;Loop if more lines to do
                                ;
                                ;  Finished
                                ;
005450  012601                        MOV     (SP)+,R1
005452  000207                        RETURN
                                ;
                                ;  End of Pro-only code
                                ;
                                        .ENDC   ;NE,<PROASM-1>
```

```
                                        .SBTTL  LINTYP -- Determine the type of a line
                                ;-----------------------------------------------------------------------
                                ;  LINTYP is called to determine if the current line is a time-sharing line
                                ;  a CL line, or a non-hardware connected line.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number
                                ;
005454  020127  000000G       LINTYP: CMP     R1,#LSTPL       ;Is this a time-sharing line?
005460  101422                        BLOS    1$              ;Br if yes
005462  020127  000000G               CMP     R1,#FSTIOL      ;Is this a CL line?
005466  103004                        BHIS    2$              ;Br if yes
005470  012761  177777  000000G       MOV     #-1,LCLUNT(R1)  ;Say line not in use as a CL line
005476  000432                        BR      9$
                                ;
                                ;  This is a CL line
                                ;
005500  016100  000000G       2$:     MOV     LCLUNT(R1),R0   ;Get the CL unit index number
005504  010160  000000G               MOV     R1,CL$LIX(R0)   ;Say which line is assoc with this CL unit
005510  012761  000000G 000000G       MOV     #CLOTIR,LOUTIR(R1) ;Set terminal output interrupt routine
005516  012761  000000G 000000G       MOV     #CLINCP,LINIR(R1)  ;Set terminal input interrupt routine
005524  000414                        BR      8$
                                ;
                                ;  This is a time-sharing line
                                ;
005526  012761  177777  000000G 1$:     MOV     #-1,LCLUNT(R1)  ;Say line is not in use as a CL unit
005534  012761  177777  000000G       MOV     #-1,LXCL(R1)    ;Line is not cross-connected to CL unit
005542  012761  000000G 000000G       MOV     #NEDCHR,LOUTIR(R1) ;Set terminal output interrupt routine
005550  012761  000000G 000000G       MOV     #TTINCP,LINIR(R1)  ;Set terminal input interrupt routine
005556  052761  000000G 000000G 8$:     BIS     #$HARD,LSW3(R1) ;This line is connected to hardware
                                ;
                                ;  Finished
                                ;
005564  000207                9$:     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.
                                ;
005566                          INITGO:
                                ;
                                ;  Save some RT-11 pointers in case we abort the initialization
                                ;
005566  013737  000004  000044'       MOV     @#4,RTTRP4      ;Save trap 4 vector
005574  013737  000000G 000046'       MOV     @#RMON,RTMNVC   ;RT-11 monitor pointer
005602  013737  000000G 000042'       MOV     @#CLKVEC,CLK100 ;Clock vector (defined in TSGEN at 100)
                                ;
                                ;  Get the current time of day which we will use later to make sure
                                ;  the line time clock is working.
                                ;
005610                                .GTIM   #AREA,#SYTIMH   ;Get the current time of day
                                ;
                                ;  Trap ^C for later test so we can restore clock vector
                                ;
005630                                .SCCA   #AREA,#CCAFLG   ;Catch control-C
                                ;
                                ;  Check for TSGEN size overflow
                                ;
005650  012700  000000G               MOV     #GENTOP,R0      ;Get top of TSGEN
005654  162700  037776                SUB     #<40000-2>,R0   ;Will TSKMON have problems?
005660  003422                        BLE     15$             ;Continue if not
005662  010046                        MOV     R0,-(SP)        ;Save overflow size
005664                                .PRINT  #TSXHD          ;Print error message
005672                                .PRINT  #TOOBIG
005700                                .PRINT  #REDUCE
005706  012600                        MOV     (SP)+,R0        ;Recover amount of overflow
005710  004737  025436'               CALL    PRTDEC
005714                                .PRINT  #BYTES
005722  000137  004250'               JMP     INISTP
                                ;
                                ;  Initialize the system stack (below 1000)
                                ;
005726  012701  000000G       15$:    MOV     #SSEND,R1       ;Point to bottom of stack
005732  012700  123456                MOV     #123456,R0      ;Get initialization value
005736  010021                13$:    MOV     R0,(R1)+        ;Initialize the stack
005740  020127  000000G               CMP     R1,#SS          ;Reached top of the stack area?
005744  103774                        BLO     13$             ;Loop if not
005746  010106                        MOV     R1,SP           ;Run on system stack
                                ;
                                ;  Make sure we are not already running under TSX.
                                ;
005750                                .SERR                   ;DON'T DIE ON ERRORS
005756  012700  000242'               MOV     #GTLIN,R0       ;TSX EMT TO GET LINE NUMBER
005762  104375                        EMT     375             ;TRY A TSX EMT
005764  103410                        BCS     1$              ;BR IF NOT UNDER TSX
005766                                .PRINT  #TSXHD          ;ALREADY UNDER TSX
005774                                .PRINT  #TSXRUN
006002  000137  004250'               JMP     INISTP
006006                          1$:   .HERR                   ;RENABLE FATAL ERRORS
```

```
                                        ;
                                        ;  Make sure this machine has memory management facilities.
                                        ;
006014                                          .TRPSET #AREA,#NOXM     ;CATCH TRAPS
006034  005737  000000G                        TST     @#SROMMR        ;TRY TO ACCESS MEMORY MANAGEMENT REGISTER
006040                                          .TRPSET #AREA,#0        ;Release trap control
                                        ;
                                        ;  Request all available memory from RT-11.
                                        ;
006056                                          .SETTOP #-2             ;REQUEST ALL AVAILABLE MEMORY
006064  010037  000132'                         MOV     R0,TOPMEM       ;REMEMBER WHERE TOP OF MEMORY IS
006070  020027  000000G                        CMP     R0,#VPAR5       ;TSX CANNOT EXTEND ABOVE PAR5 BASE ADDRESS
006074  101402                                 BLOS    3$              ;BR IF RT-11 IS BELOW THAT
006076  012700  000000G                        MOV     #VPAR5,R0       ;SET PAR5 BASE AS UPPER LIMIT ON TSX SIZE
006102  010037  000302'                3$:     MOV     R0,MEMLIM       ;TSX MAY NOT EXCEED THIS UPPER LIMIT
                                        ;
                                        ;  Lock USR in memory for speed
                                        ;  (Set USR to swap over TSEMT to get out of the way)
                                        ;
006106  012705  177776'                         MOV     #TSINIT-2,R5    ;GET THE BASE OF TSINIT
006112                                          .GVAL   #AREA,#374      ;GET SIZE OF RT-11 USR MODULE
006132  160005                                  SUB     R0,R5           ;ALLOCATE SPACE BELOW TSINIT FOR USR
006134  010537  000046                         MOV     R5,@#46         ;SET USR TO SWAP OVER TSEMT
006140                                  5$:     .LOCK                   ;LOCK USR IN MEMORY
                                        ;
                                        ;  Determine if we are to run system with the system debugger
                                        ;
006142  032737  000000G 000000G                 BIT     #CHAIN,@#JSWLOC ;WERE WE CHAINED TO?
006150  001406                                 BEQ     10$             ;BR IF NOT
006152  023737  000510  000222'                CMP     @#510,R50ODT    ;SHOULD WE RUN UNDER ODT?
006160  001002                                 BNE     10$             ;BR IF NOT
006162  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.
                                        ;
006166                                  10$:
                                        .IF     NE,PROCID               ;** Do if assembling for pro only **
006166  004737  025562'                         CALL    INSCHK          ;PERFORM VERIFICATION AND DECRYPTION FOR PRO
                                        .ENDC   ;NE,PROCID
                                        ;
                                        ;  Allocate non-initialized buffer space over TSINIT.
                                        ;
006172  012705  000000'                         MOV     #TSINIT,R5      ;Allocate buffer space over TSINIT
006176  004737  011652'                         CALL    ALOCBF          ;Do allocation
006202  004737  012316'                         CALL    ALCSLO          ;Allocate silo buffers for lines
006206  020527  005566'                         CMP     R5,#INITGO      ;Are we beyond code that takes over control?
006212  103002                                 BHIS    12$             ;Br if yes
006214  012705  005566'                         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
006220  010537  000000G         12$:    MOV     R5,INTSND               ;Ptr to base of interrupt stack
006224  062705  000002                         ADD     #2,R5           ;Always leave last word of stack for flag val
```

```
    115 006230  013701  000000G                 MOV     @#RMON,R1       ;Get pointer to RT-11 RMON base
    116 006234  032761  000000G 000370          BIT     #CW$PRO,370(R1) ;Are we running on a PRO?
    117 006242  001407                          BEQ     11$             ;Br if not
    118 006244  105237  000000G                 INCB    PROFLG          ;Set flag saying this is a PRO-350
    119 006250  010537  000150'                 MOV     R5,PROBUF       ;Save pointer to buffer area
    120 006254  062705  001274                  ADD     #PIINSZ,R5      ;Allocate space for buffer
    121 006260  000402                          BR      14$
    122 006262  062705  000000G         11$:    ADD     #INTSSZ,R5      ;Allocate space for interrupt stack
    123 006266  010537  000000G         14$:    MOV     R5,INTSTK       ;Address of top of interrupt stack
    124                                 ;
    125                                 ;  Allocate space for those overlays that go over TSINIT
    126                                 ;
    127 006272  004737  021624'                 CALL    OVLPOS          ;Determine how much space to alloc for overlay
    128                                 ;
    129                                 ;  Note: from this point onward we are carrying the address of the
    130                                 ;  base of the free memory area in R5.
    131                                 ;
    132 006276  020527  025556'                 CMP     R5,#INITOP      ;Have we allocated up to top of TSINIT?
    133 006302  103002                          BHIS    4$              ;Br if yes
    134 006304  012705  025556'                 MOV     #INITOP,R5      ;Advance to top of TSINIT
    135                                 ;
    136                                 ;  Allocate a 2048. byte work buffer
    137                                 ;
    138 006310  004737  007454'         4$:     CALL    ALCWRK          ;Allocate work buffer
    139                                 ;
    140                                 ;  Allocate empty Region Control Blocks for use by handlers
    141                                 ;
    142 006314  004737  007510'                 CALL    ALCHRB
    143                                 ;
    144                                 ;  If we were started in debug mode, load ODT.
    145                                 ;
    146                                         .IF     EQ,PROCID       ;Don't allow ODT for production PRO version
    147                                         TST     ODTFLG          ;Are we to load system debugger?
    148                                         BEQ     2$              ;Br if not
    149                                         CALL    GETODT          ;Load ODT and start it
    150                                         .ENDC   ;EQ,PROCID
    151                                 ;
    152                                 ;  Initialize memory management registers for 1-to-1 mapping but
    153                                 ;  leave memory management turned off
    154                                 ;
    155 006320  004737  015356'         2$:     CALL    MEMINI          ;Initialize memory management
    156                                 ;
    157                                 ;  Extract information from RT-11 configuration and sysgen words.
    158                                 ;
    159 006324  013701  000000G                 MOV     @#RMON,R1       ;GET POINTER TO BASE OF RMON
    160 006330  016102  000300                  MOV     300(R1),R2      ;GET RT-11 CONFIGURATION WORD
    161 006334  042702  000000C                 BIC     #CW$GDH+CW$BTH+CW$LGS,R2 ;RESET A FEW FLAGS
    162 006340  052702  000000C                 BIS     #CW$FB+CW$FQJ+CW$USR+CW$XM,R2 ;SET A FEW FLAGS
    163 006344  010237  000000G                 MOV     R2,CONFIG       ;INITIALIZE OUR CONFIGURATION WORD
    164                                 ;  Now get extended configutation word.
    165 006350  016137  000370  000000G         MOV     370(R1),CONFG2  ;EXTENDED CONFIGURATION WORD
    166 006356  052737  000000G 000000G         BIS     #CW$ESP,CONFG2  ;SET EXIT NO SWAP FLAG
    167 006364  123727  000000G 000000G         CMPB    VBUSTP,#QBUS    ;Is this a Q-bus machine?
    168 006372  001003                          BNE     25$             ;Br if not
    169 006374  052737  000000G 000000G         BIS     #CW$QBS,CONFG2  ;Set QBUS flag
    170                                 ;  And sysgen option word.
    171 006402  016102  000372          25$:    MOV     372(R1),R2
```

```
    172 006406  042702  000000C                 BIC     #SG$ELG+SG$PAR+SG$MTS,R2
    173 006412  052702  000000C                 BIS     #SG$MMU+SG$MTM+SG$IOT+SG$TSX,R2
    174 006416  010237  0000000                 MOV     R2,SYSGEN       ;INITIALIZE OUR SYSGEN WORD
    175                                 ;  Get system version number
    176 006422                                  .GVAL   #AREA,#276      ;GET RT-11 SYSTEM VERSION NUMBER
    177 006442  010037  0000000                 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
                                        ;
006446  032737  0000000G 0000000G INICLK: BIT     #CW$50H,CONFIG  ;50 or 60 Hz clock?
006454  001017                           BNE     2$              ;Br if 50 Hz
                                        ;
                                        ;  60 Hz clock
                                        ;
006456  012737  000074  0000000G         MOV     #60.,TK1SEC     ;Clock ticks per 1 second
006464  012737  000036  0000000G         MOV     #30.,TK5VAL     ;Clock ticks per 0.5 seconds
006472  012737  000264  0000000G         MOV     #180.,TK3SVL    ;Clock ticks per 3 seconds
006500  012737  000006  0000000G         MOV     #6.,TK1VAL      ;Clock ticks per 0.1 seconds
006506  012700  001130                   MOV     #600.,R0        ;Get # clock ticks per 10 seconds
006512  000416                           BR      8$
                                        ;
                                        ;  50 Hz clock
                                        ;
006514  012737  000062  0000000G 2$:     MOV     #50.,TK1SEC     ;Clock ticks per 1 second
006522  012737  000031  0000000G         MOV     #25.,TK5VAL     ;Clock ticks per 0.5 seconds
006530  012737  000226  0000000G         MOV     #150.,TK3SVL    ;Clock ticks per 3 seconds
006536  012737  000005  0000000G         MOV     #5.,TK1VAL      ;Clock ticks per 0.1 seconds
006544  012700  000764                   MOV     #500.,R0        ;Get # clock ticks per 10 seconds
                                        ;
                                        ;  Set number of clock ticks per day
                                        ;
006550  012702  020700           8$:     MOV     #8640.,R2       ;(# seconds per day) / 10.
006554  070200                           MUL     R0,R2           ;Get # clock ticks per day
006556  010237  0000000G                 MOV     R2,DATIMH       ;High-order value
006562  010337  0000000G                 MOV     R3,DATIML       ;Low-order value
                                        ;
                                        ;  Do a fast check to make sure specified T/S line addresses are ok.
                                        ;
006566  004737  005374'          CKLIN:  CALL    LINCHK          ;CHECK T/S LINE ADDRESSES
                                        ;
                                        ;  Do PRO-350 system initialization
                                        ;
                                        .IF     NE,PROASM
006572  032737  0000000G 0000000G         BIT     #CW$PRO,CONFG2  ;Are we running on a PRO-350?
006600  001405                           BEQ     INIDEV          ;Br if not
006602  004737  0000000G                 CALL    PROINI          ;Do PRO-350 initialization
006606  012737  0000000G 0000000G         MOV     #PIDRIV,PIDPTR  ;Set up pointer to clock driven PI routine
                                        .ENDC   ;NE,PROASM
                                        ;
                                        ;  Make entry in device handler table for TT device.
                                        ;
006614  013737  000172' 0000000G INIDEV: MOV     R50TT,PNAME     ;PERMANENT NAME "TT"
006622  012737  0000000G 0000000G         MOV     #DI$TT,DVSTAT   ;SET DEVICE STATUS FLAGS FOR TT
006630  005037  0000000G                 CLR     DVFLAG
006634  005037  0000000G                 CLR     DEVSIZ
006640  012737  000002  0000000G         MOV     #2,HANENT       ;SET UP HANENT SO HANDLER LOOKS RESIDENT
006646  005037  0000000G                 CLR     NUMDEV          ;IT IS DEVICE # 0
                                        ;
                                        ;  Make device table entry for LD (logical disk) device
                                        ;
006652  012737  177777  0000000G         MOV     #-1,LDDEVX      ;ASSUME LD SUPPORT NOT WANTED
006660  105737  0000000G                 TSTB    VLDSYS          ;IS LD SUPPORT GENNED IN?
```

```
006664  001427                          BEQ     6$                  ;BR IF NOT
006666  062737  000002  000000G         ADD     #2,NUMDEV           ;ONE MORE DEVICE
006674  013701  000000G                 MOV     NUMDEV,R1           ;GET DEVICE TABLE INDEX
006700  010137  000000G                 MOV     R1,LDDEVX           ;REMEMBER INDEX NUMBER FOR LD DEVICE
006704  013761  000176'  000000G        MOV     R50LD,PNAME(R1)     ;SET DEVICE NAME ("LD")
006712  012761  000000C  000000G        MOV     #<DS$DIR+DS$SFN+DS$VSZ+DI$LD>,DVSTAT(R1);SET DEV STATUS FLAGS
006720  012761  000000G  000000G        MOV     #DX$EBA,DVFLAG(R1);Say buffers must be on even byte boundaries
006726  005061  000000G                 CLR     DEVSIZ(R1)
006732  012761  000002  000000G        MOV     #2,HANENT(R1)       ;SAY HANDLER IS RESIDENT
006740  004737  014156'                 CALL    LDINIT              ;DETERMINE LD TRANSLATION TABLE FORMAT
                                ;
                                ;  Make device table entry for CL (communications line) device
                                ;
006744  005727  000000G         6$:     TST     #CLTOTL             ;Are there any communications lines?
006750  001402                          BEQ     8$                  ;Br if not
006752  004737  013436'                 CALL    CLINIT              ;Initialize CL handler
                                ;
                                ;  Disable clock interrupts.
                                ;
006756  012737  000002  000000  8$:     MOV     #2,@#0              ;LOAD RTI IN LOCATION 0
006764  005037  000000G                 CLR     @#CLKVEC            ;ATTACH CLOCK INTERRUPT TO 0
006770  032737  000000G  000000G        BIT     #CW$PRO,CONFG2      ;ARE WE RUNNING ON A PRO?
006776  001402                          BEQ     1$                  ;BR IF NOT
007000  005037  000230                 CLR     @#230               ;380 CLOCK INTERRUPT VECTOR
                                ;
                                ;  Set up memory parity control
                                ;
007004  004737  016426'         1$:     CALL    PARSET              ;SET UP MEMORY PARITY CONTROL
                                ;
                                ;  Determine how much memory is installed on machine
                                ;
007010  004737  015462'                 CALL    MEMTST              ;FIND OUT HOW MUCH PHYSICAL MEMORY THERE IS
                                ;
                                ;  Set up information about the size of the job context area
                                ;
007014  004737  015756'                 CALL    CXTALC              ;Determine size of job context area
                                ;
                                ;  Load TSX-Plus device handlers that go in low memory
                                ;
007020  004737  016430'                 CALL    GETHNL              ;Load low memory handlers
                                ;
                                ;  Reserve space for interrupt vector intercept routines for mapped handlers
                                ;
007024  010537  000122'                 MOV     R5,XMVBAS           ;Save address of base of area for XM vectors
007030  013701  000124'                 MOV     NMXHAN,R1           ;Get # mapped handlers
007034  006301                          ASL     R1                  ;Reserve room for 2 interrupts per handler
007036  062701  000000G                 ADD     #NXIVMH,R1          ;Add # requested extra interrupt vectors
007042  070127  000032                  MUL     #MPIVSZ,R1          ;Calc space needed for interrupt entry code
007046  060105                          ADD     R1,R5               ;Advance the address of free memory
                                ;
                                ;  Set up device index number and unit number for "SY:" device.
                                ;
007050  004737  025006'                 CALL    SETSY               ;SET UP INFO ABOUT SY DEVICE
                                ;
                                ;  Open channel to TSKMON and set up information about it.
                                ;
007054  004737  013134'                 CALL    OPNKMN              ;OPEN CHANNEL TO TSKMON
```

```
                                        ;
                                        ;  Set up information about the IND program
                                        ;
007060  004737  014236'                       CALL    INDINI          ;INITIALIZE FOR IND PROGRAM
                                        ;
                                        ;  Initialize the TSXUCL data file
                                        ;
007064  004737  015000'                       CALL    UCLINI
                                        ;
                                        ;  Set name of device that UCL program is to be run from
                                        ;
007070  013737  000000G 000000G               MOV     SYNAME,UCLNAM   ;SET DEVICE NAME FOR UCL PROGRAM
                                        ;
                                        ;  Initialize spooling system
                                        ;
007076  004737  010522'                       CALL    SPLINI          ;INITIALIZE SPOOLING SYSTEM
                                        ;
                                        ;  Open system swap file
                                        ;
007102  105737  000000G                       TSTB    VSWPFL          ;IS JOB SWAPPING ALLOWED?
007106  001402                               BEQ     3$              ;BR IF NOT
007110  004737  007564'                       CALL    OPNSWP          ;OPEN THE SYSTEM SWAP FILE
                                        ;
                                        ;  Open swap file used for PLAS regions
                                        ;
007114  004737  010216'                3$:    CALL    OPNRSF          ;Open PLAS region swap file
                                        ;
                                        ;  Set up information about which devices need to have their I/O mapped
                                        ;
007120  004737  021622'                       CALL    SETMIO          ;Set up information about mapped I/O
                                        ;
                                        ;  We are finished allocating low-memory buffer space.
                                        ;
007124  010500                               MOV     R5,R0           ;ENSURE WE DON'T OVERFLOW 40KB
007126  004737  025312'                       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.
                                        ;
007132  010537  000000G                       MOV     R5,UMSYTP       ;SAVE ADDRESS OF NON-EXTENDED SYSTEM TOP
007136  062705  000077                       ADD     #77,R5          ;ROUND UP TO 64-BYTE BOUNDARY
007142  072527  177772                       ASH     #-6,R5          ;CONVERT TO 64-BYTE BLOCK #
007146  042705  176000                       BIC     #176000,R5      ;KILL SIGN EXTENSION
                                        ;
                                        ;  Allocate buffer space that is not contrained to 40Kb TSX-Plus region.
                                        ;
007152  004737  012536'                       CALL    ALBFX           ;ALLOCATE EXTENDED BUFFERS
                                        ;
                                        ;  We will now do some allocation from the top of physical memory downward.
                                        ;  Save the base of free memory in R4 and get the top of free memory to R5.
                                        ;
007156  010504                               MOV     R5,R4           ;Save the base of free memory in R4
007160  010437  000136'                       MOV     R4,FMEMLO       ;Save pointer above top of alloc low memory
007164  013705  000134'                       MOV     FMEMHI,R5       ;Get 64-byte block # of free high memory
                                        ;
                                        ;  Load any mapped system code
                                        ;
```

```
                                        CALL    GETMAP          ;LOAD USR, EMT, MSG, LOCK, SPOOL, etc.
                                ;
                                ;  Load any shared run-time systems
                                ;
                                        TST     #NUMRDB         ;Do we need to load any shared run-times?
                                        BEQ     4$              ;Br if not
                                        MOV     #RDB,R1         ;Point to 1st run-time descriptor block
                                        MOV     R5,R2           ;Save initial memory pointer
                                5$:     CALL    GETSRT          ;Load a shared run-time system
                                        ADD     #RT$$SZ,R1      ;Point to next shared run-time descriptor
                                        CMP     R1,#RDBEND      ;Are there more to load?
                                        BLO     5$              ;Br if yes
                                        SUB     R5,R2           ;Compute amt of space used by run-times
                                        MOV     R2,SRTSIZ       ;Save total run-time size
                                4$:
                                .IF     NE,PROASM
                                ;
                                ;  If we are running on a Pro, load the PI handler like a shared run-time
                                ;
                                        TSTB    PROFLG          ;Are we running on a Pro?
                                        BEQ     10$             ;Br if not
                                        MOV     #PISRT,R1       ;Point to dummy shared run-time block for PI
                                        MOV     R5,R2           ;Save current memory pointer
                                        CALL    GETSRT          ;Load PI handler like a shared run-time
                                        SUB     R5,R2           ;Calculate amt of space used by PI handler
                                        ADD     R2,MHNSIZ       ;Count in mapped-handler size
                                .ENDC   ;NE,PROASM
                                ;
                                ;  Load any mapped handlers
                                ;
                                10$:    CALL    GETHNH          ;Load mapped handlers
                                ;
                                ;  Allocate space for data cache buffers and control tables
                                ;
                                        CALL    CSHBUF          ;Allocate space for data cache
                                ;
                                ;  We have finished allocating all of the memory used by the system.
                                ;  Allocate and initialize a memory map table that will be used to
                                ;  show which pages are available for user jobs.
                                ;
                                        CALL    MAPALC          ;Allocate memory map table
                                ;
                                ;  Set up info about maximum memory space available to jobs
                                ;
                                        CALL    SETJSZ          ;SET JOB SIZE INFO
                                ;
                                ;  Set up date and time
                                ;
                                        MOV     SYTIML,R2       ;Save time we got at start of init
                                        .GTIM   #AREA,#SYTIMH   ;SET TIME OF DAY
                                        .DATE                   ;GET DATE
                                        MOV     R0,SYSDAT       ;SET SYSTEM DATE
                                        CMP     R2,SYTIML       ;Make sure some time has elapsed
                                        BNE     11$             ;Br if clock is running
                                        .PRINT  #TSXHD          ;Print error message heading
                                        .PRINT  #NOCLOK         ;Print clock-not-working message
                                        JMP     INISTP          ;Abort initialization
```

```
                                        ;
                                        ;  Unlock the USR so that TSEMT will be swapped back in.
                                        ;
007366                                  11$:    .UNLOCK                 ;RELEASE USR
                                        ;
                                        ;  Read back into memory that part of the resident portion of TSX
                                        ;  that we overlayed with our work buffer.
                                        ;
007370  013702  000154'                         MOV     WRKSIZ,R2       ;Get size of work buffer
007374  006202                          ASR     R2                      ;Convert to # words
007376  013703  000152'                 MOV     WRKBUF,R3               ;Get address of work buffer area
007402  000241                          CLC                             ;Convert to block # in TSX.SAV file
007404  006003                          ROR     R3
007406  000303                          SWAB    R3
007410                                  .READW  #AREA,#17,WRKBUF,R2,R3 ;Read back TSX over work buffer
                                        ;
                                        ;  See if user requested control-C abort
                                        ;
007444  004737  025352'                 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
                                        ;
007450  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).
                                ;
007454  010246                  ALCWRK: MOV     R2,-(SP)
                                ;
                                ;  Get address of start of area where buffer can go and then bound
                                ;  up to a block boundary.
                                ;
007456  012702  000000G                 MOV     #EXCBUF,R2      ;Get address of base of buffer area
007462  062702  000777                  ADD     #777,R2         ;Bound up to block boundary
007466  042702  000777                  BIC     #777,R2         ;Set to block boundary
007472  010237  000152'                 MOV     R2,WRKBUF
007476  012737  004000  000154'         MOV     #2048.,WRKSIZ
                                ;
                                ;  Finished
                                ;
007504  012602                          MOV     (SP)+,R2
007506  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.
                                ;
007510  010246                  ALCHRB: MOV     R2,-(SP)
                                ;
                                ;  Get count of # RCB's to build
                                ;
007512  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
                                ;
007516  010537  000000G                 MOV     R5,HANRCB       ;Start of RCB area
007522  010537  000000G                 MOV     R5,HANRCO       ;Store offset relative to MONVEC
007526  162737  000000G 000000G        SUB     #MONVEC,HANRCO  ;Convert address to offset
007534  012725  177777                  MOV     #-1,(R5)+       ;Store -1 at start of area
                                ;
                                ;  Allocate and initialize to zero the RCB's
                                ;
007540  012702  000005          1$:     MOV     #5.,R2          ;Each RCB has 5 words
007544  005025                  2$:     CLR     (R5)+           ;Zero the RCB
007546  077202                          SOB     R2,2$
                                ;
                                ;  See if there are more RCB's to build
                                ;
007550  005300                          DEC     R0              ;More RCB's to initialize?
007552  003372                          BGT     1$              ;Loop if yes
                                ;
                                ;  Store -1 at end of RCB area
                                ;
007554  012725  177777                  MOV     #-1,(R5)+       ;Mark end of RCB list
                                ;
                                ;  Finished
                                ;
007560  012602                          MOV     (SP)+,R2
007562  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 $DEAD flag for the line if INIABT=0.
                ;
                LINCHK: MOV     R1,-(SP)
                        MOV     R2,-(SP)
                        MOV     R3,-(SP)
                        MOV     R4,-(SP)
                        MOV     @#4,-(SP)       ;SAVE ORIGINAL TRAP VECTOR
                ;
                ;  Take over trap control
                ;
                        MOV     #6$,@#4         ;CATCH TRAPS
                ;
                ;  Loop through the test for each line.
                ;
                        MOV     #LSTLIN,R1      ;NUMBER OF LAST LINE
                ;
                ;  Determine if this is a primary line or an I/O line and set the
                ;  addresses of the interrupt service routines.
                ;
                1$:     CALL    LINTYP          ;Determine the type of this line
                        BIT     #$HARD,LSW3(R1) ;Is this line connected to hardware?
                        BEQ     31$             ;Br if not
                        MOV     LMXNUM(R1),R3   ;IS THIS A DL-11 OR MULTIPLEXER LINE?
                        BEQ     2$              ;BR IF DL-11
                        MOV     MXCSR(R3),R2    ;GET DZ11 OR DH11 STATUS REGISTER ADDRESS
                        BEQ     11$             ;BR IF ALREADY MARKED AS DEAD
                        MOV     MXVEC(R3),R4    ;GET MUX INTERRUPT VECTOR ADDRESS
                        BR      3$
                11$:    CALL    4$              ;MARK LINE AS DEAD
                        BR      31$             ;CONTINUE CHECKING TERMINALS
                2$:     MOV     RSR(R1),R2      ;GET DL-11 STATUS REGISTER ADDRESS
                        MOV     INVEC(R1),R4    ;GET DL-11 INTERRUPT VECTOR ADDRESS
                ;  Check validity of status register address.
                3$:     CMP     R2,#160000      ;IS IT IN I/O PAGE?
                        BLO     LINTRP          ;ERROR IF NOT
                        BIT     #7,R2           ;IS IT ON 8-BYTE BOUNDARY?
                        BNE     LINTRP          ;ERROR IF NOT
                        TST     @R2             ;TRY TO ACCESS IT AND SEE IF WE TRAP
                ;  Check validty of interrupt vector address.
                        CMP     R4,#60          ;CAN'T BE BELOW 60
                        BLO     BADVEC
                        CMP     R4,#500         ;OR ABOVE 500
                        BHIS    BADVEC
                        BIT     #7,R4           ;MUST BE ON 8-BYTE BOUNDARY
                        BNE     BADVEC
                ;  This line looks good.  Check next.
                31$:    SUB     #2,R1           ;MORE TO CHECK?
                        BGT     1$              ;BR IF YES
                ;
                ;  Finished -- all lines look ok.
                ;
```

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

                ;
                ;  Trap occured while trying to access status register.
                ;
6$:             CALL    4$              ;REPORT ERROR OR MARK AS DEAD LINE
                RTI                     ;RETURN TO LINE CHECKING
                ;
                ;  Error: Invalid status register address.
                ;  R1 = Line number,  R2 = status register address
                ;
LINTRP:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
                .PRINT  #BADLIN
                BR      ERP
                ;
                ;  Error: Invalid interrupt vector address.
                ;  R1 = Line number,  R4 = interrupt vector address
                ;
BADVEC:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
                .PRINT  #BDVMSG
                MOV     R4,R2           ;GET VECTOR ADDRESS TO R2
ERP:            MOV     R2,R0           ;GET ADDRESS TO R0
                CALL    PRTOCT          ;PRINT OCTAL VALUE
                .PRINT  #CRLF
                .PRINT  #BDLMSG         ;LINE # =
                MOV     R1,R0           ;GET LINE NUMBER
                ASR     R0              ;# 1
                CALL    PRTDEC          ;PRINT LINE NUMBER
                .PRINT  #CRLF
                JMP     INISTP          ;ABORT INITIALIZATION
                .ENDC   ;NE,<PROASM-1>
```

```
                                        .SBTTL  OPNSWP -- Open system swap file
                                ;-----------------------------------------------------------------------
                                ;  OPNSWP is called to open the TSX job swap file.
                                ;  It also assigns swap file slots for each line.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of base of free memory region
                                ;
                                ;  Outputs:
                                ;   SWPCHN = Set up for access to swap file.
                                ;   LSWPBK(i) = Starting block number in swap file for swap area for line.
                                ;
007564  010146                  OPNSWP: MOV     R1,-(SP)
007566  010246                          MOV     R2,-(SP)
007570  010346                          MOV     R3,-(SP)
                                ;
                                ;  Load RT-11 handler for swap device.
                                ;
007572  013700  000000G                 MOV     SWDBLK,R0       ;Get name of device
007576  004737  025176'                 CALL    RTFTCH          ;Fetch the RT-11 handler
007602  103546                          BCS     11$             ;Br if invalid device
                                ;
                                ;  Compute the maximum number of slots in swap file that we could need
                                ;
007604  012703  000000C                 MOV     #NSL+NDL,R3     ;Get # virtual lines and detached jobs
007610  012701  000000G                 MOV     #LSTPL,R1       ;Get index to last primary line
007614  032761  000000G 000000G 1$:     BIT     #$DEAD,LSW3(R1) ;Is this line installed?
007622  001001                          BNE     5$              ;Br if not
007624  005203                          INC     R3              ;Count another primary line
007626  162701  000002          5$:     SUB     #2,R1           ;Get next line index
007632  003370                          BGT     1$              ;Loop if more lines to check
007634  020337  0000000                 CMP     R3,VSWPSL       ;Compare with # slots specified
007640  002002                          BGE     6$              ;Br if VSWPSL value is ok
007642  010337  0000000                 MOV     R3,VSWPSL       ;Reduce number of slots in swap file
                                ;
                                ;  Determine how many blocks are needed for each slot in swap file.
                                ;
007646  013703  0000000         6$:     MOV     VHIMEM,R3       ;GET # BLOCKS NEEDED FOR LARGEST JOB SIZE
007652  006303                          ASL     R3
007654  063703  0000000                 ADD     CXTPAG,R3       ;ADD # BLOCKS NEEDED FOR JOB CONTEXT AREA
007660  010337  0000000                 MOV     R3,SLTSIZ       ;Save size of swap file slot
                                ;
                                ;  Compute the total number of blocks needed for the swap file.
                                ;
007664  070337  0000000                 MUL     VSWPSL,R3       ;Multiply by # slots in swap file
                                ;
                                ;  R3 now contains total number of blocks needed in swap file.
                                ;  See if swap file already exists on disk.
                                ;
007670                          4$:     .LOOKUP #AREA,#1,#SWDBLK;DOES SWAP FILE EXIST NOW?
007710  103415                          BCS     2$              ;BR IF NOT
                                ;  Swap file exists.  See if it is the right size.
007712  020003                          CMP     R0,R3           ;IS SWAP FILE THE RIGHT SIZE?
007714  001453                          BEQ     3$              ;BR IF YES
                                ;  Old swap file is not of correct size.
                                ;  Delete it and open a new swap file.
007716                                  .CLOSE  #1
```

```
                                        .DELETE #AREA,#1,#SWDBLK;DELETE THE OLD SWAP FILE
                                ;
                                ;  Create a new swap file.
                                ;
                        2$:     .ENTER  #AREA,#1,#SWDBLK,R3 ;CREATE A NEW SWAP FILE
                                BCS     9$                  ;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.
                                ;
                                DEC     R3                  ;GET # OF LAST BLOCK IN FILE
                                .WRITW  #AREA,#1,#TSINIT,#256.,R3 ;WRITE TO LAST BLOCK IN FILE
                                INC     R3                  ;GET BACK # BLOCKS IN FILE
                                .CLOSE  #1                  ;CLOSE FILE WE CREATED
                                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.
                                ;
                        3$:     MOV     #SWPCHN,R0          ;POINT TO SWAP CHANNEL BLOCK
                                MOV     SWDBLK,R2           ;GET DEVICE NAME
                                CALL    SETCHN              ;SET UP SWAP CHANNEL INFO
                                ;
                                ;  Release the RT-11 device handler
                                ;
                                .RELEAS #SWDBLK             ;Release RT-11 device handler
                                ;
                                ;  Finished
                                ;
                                MOV     (SP)+,R3
                                MOV     (SP)+,R2
                                MOV     (SP)+,R1
                                RETURN
                                ;
                                ;  Error: Cannot open swap file
                                ;
                        9$:     .PRINT  #TSXHD              ;PRINT ERROR MESSAGE
                                .PRINT  #BADOPN
                                CALL    SPNEED              ;Print info about number of blocks needed
                                ;
                                ;  Error: Invalid device specification.
                                ;
                        11$:    MOV     R0,R1               ;Save device name
                                .PRINT  #TSXHD              ;Print error message
                                .PRINT  #BADOPN
                                CALL    BADDEV              ;Print invalid device specification
                                ;
                                ;  Error: Number of contiguous blocks required.
                                ;
                        SPNEED: .PRINT  #CONSPC             ;Print contiquous blocks needed
                                MOV     R3,R0               ;GET # BLOCKS NEEDED FOR FILE
                                CALL    PRTDEC              ;DISPLAY # BLOCKS NEEDED
                                .PRINT  #CRLF
                                JMP     INISTP              ;ABORT INITIALIZATION
                                ;
                                ;  Bad file specification.
```

```
                                                ;
010170                                  BADDEV: .PRINT  #CFHMSG         ;Print invalid device specification
010176  010100                                  MOV     R1,R0           ;Get the rad50 device name
010200  004737  025502'                         CALL    PRTR50          ;Print rad50 device name
010204                                          .PRINT  #CRLF           ;Print carriage return/line feed
010212  000137  004250'                         JMP     INISTP          ;Abort initialization
```

```
                        .SBTTL  OPNRSF -- Open PLAS region swap file
                ;-----------------------------------------------------------------------
                ;  OPNRSF is called to open the swap file used for PLAS regions.
                ;
                ;  Inputs:
                ;   R5 = Address of base of free memory area.
                ;
                ;  Outputs:
                ;   SEGCHN = Set up to access swap file.
                ;
010216  010346  OPNRSF: MOV     R3,-(SP)
                ;
                ;  Return if this is a non-swapping system or if region swap file is
                ;  not wanted.
                ;
010220  105737  0000000         TSTB    VSWPFL          ;Is this a non-swapping system?
010224  001513                  BEQ     9$              ;Br if yes
010226  005737  0000000         TST     VPLAS           ;Is a PLAS swap file wanted?
010232  001510                  BEQ     9$              ;Br if not
                ;
                ;  Load RT-11 device handler for swap device
                ;
010234  013700  0000000         MOV     RSFBLK,R0       ;Get name of device
010240  004737  025176'         CALL    RTFTCH          ;Try to fetch the RT-11 device handler
010244  103515                  BCS     11$             ;Br if error on handler fetch
010246  013703  0000000         MOV     VPLAS,R3        ;Get # blocks in PLAS swap file
                ;
                ;  See if PLAS swap file already exists on disk
                ;
010252          4$:     .LOOKUP #AREA,#1,#RSFBLK ;Try to find existing PLAS swap file
010272  103416          BCS     2$              ;Br if file does not now exist
                ;
                ;  PLAS swap file exists.
                ;  See if it is the correct size.
                ;
010274  020037  0000000         CMP     R0,VPLAS        ;Is swap file of the correct size?
010300  001453                  BEQ     3$              ;Br if yes
                ;
                ;  Old PLAS swap file is not of correct size.
                ;  Delete it and open a new swap file.
                ;
010302                  .CLOSE  #1              ;Close and delete the old file
010310                  .DELETE #AREA,#1,#RSFBLK;Delete the old file
                ;
                ;  Create new swap file
                ;
010330          2$:     .ENTER  #AREA,#1,#RSFBLK,VPLAS ;Create a new PLAS swap file
010356  103440          BCS     10$             ;Br if cannot create new file
                ;
                ;  New swap file has been created.
                ;  Write to last block to reserve full file size
                ;  and then close and reopen with a lookup.
                ;
010360  005303          DEC     R3              ;Get # of last block in file
010362                  .WRITW  #AREA,#1,#TSINIT,#256.,R3
010420                  .CLOSE  #1
010426  000711          BR      4$              ;Go back and lookup file
```

```
                                ;
                                ;  Swap file has been successfully opened using lookup.
                                ;  Copy channel status to TSX channel block.
                                ;
010430  012700  000000G        3$:     MOV     #SEGCHN,R0      ;Point to TSX PLAS swap channel
010434  013703  000000G                MOV     RSFBLK,R3       ;Get device name
010440  004737  024616'                CALL    SETCHN          ;Set up TSX channel block
                                ;
                                ;  Release the RT-11 device handler
                                ;
010444                                  .RELEAS #RSFBLK         ;Release RT-11 device handler
                                ;
                                ;  Finished
                                ;
010454  012603                  9$:     MOV     (SP)+,R3
010456  000207                          RETURN
                                ;
                                ;  Error -- Cannot open PLAS swap file
                                ;
010460                          10$:    .PRINT  #TSXHD          ;Print error prefix
010466                                  .PRINT  #RSFERR         ;Print error message
010474  004737  010142'                CALL    SPNEED          ;Print information about amt of space needed
                                ;
                                ;  Error: Invalid device specification.
                                ;
010500  010001                  11$:    MOV     R0,R1           ;Save device name
010502                                  .PRINT  #TSXHD          ;Print error message
010510                                  .PRINT  #RSFERR
010516  004737  010170'                CALL    BADDEV          ;Print invalid device specification
```

```
                                        .SBTTL  SPLINI -- Initialize spooling system
                                ;-----------------------------------------------------------------------
                                ;  SPLINI performs the initialization of the spooling system.
                                ;  Inputs:
                                ;   R5 = Current base of free memory area.
                                ;
010522  005727  0000000G        SPLINI: TST     #SPLND          ;Are there any spooled devices?
010526  001401                          BEQ     13$             ;Br if not
010530  000401                          BR      10$             ;Initialize the spooled devices
010532  000207                13$:    RETURN
                                ;
                                ;  There are some spooled devices
                                ;
010534  010146                10$:    MOV     R1,-(SP)
010536  010246                        MOV     R2,-(SP)
010540  010346                        MOV     R3,-(SP)
010542  010446                        MOV     R4,-(SP)
010544  010546                        MOV     R5,-(SP)
                                ;
                                ;  Open each spooled device
                                ;
010546  105037  0000000G                CLRB    NSPLDV          ;INIT COUNT OF # ACTUAL SPOOLED DEVICES
010552  012701  0000000G                MOV     #SDCB,R1        ;POINT TO 1ST SDCB
010556  012703  0000000G                MOV     #SPLDEV,R3      ;POINT TO TABLE OF RAD50 DEV NAMES
010562  012704  0000000G                MOV     #SPLANM,R4      ;POINT TO TABLE OF ASCII DEV NAMES
010566  004737  011574'        2$:     CALL    FORCE0          ;FORCE UNIDENTIFIED UNIT #S TO 0
010572  011302                        MOV     (R3),R2         ;GET RAD50 NAME OF SPOOLED DEVICE
010574  010261  0000000G                MOV     R2,SDNAME(R1)   ;SET NAME IN SDCB
010600  010100                        MOV     R1,R0           ;GET ADDRESS OF SDCB
010602  062700  0000000G                ADD     #SDANAM,R0      ;POINT TO CELL FOR ASCII NAME
010606  112420                        MOVB    (R4)+,(R0)+     ;MOVE IN ASCII DEVICE NAME
010610  112420                        MOVB    (R4)+,(R0)+
010612  112420                        MOVB    (R4)+,(R0)+
010614  020227  0000000G                CMP     R2,#DMYDEV      ;Is this a dummy entry for later patching?
010620  001451                        BEQ     1$              ;Br if yes -- Ignore it
010622  010200                        MOV     R2,R0           ;Get name to R0
010624  004737  011476'                CALL    CVTDVU          ;Convert name to device # and unit #
010630  010061  0000000G                MOV     R0,SDDVU(R1)    ;Store device # and unit # in SDCB
010634  010200                        MOV     R2,R0           ;Get device name
010636  004737  011372'                CALL    CHKCLD          ;See if this is a CL device?
010642  103406                        BCS     14$             ;Br if not
010644  004737  011306'                CALL    SPLCLD          ;Set up for spooling to CL device
010650  103414                        BCS     3$              ;Br if invalid unit
010652  105237  0000000G                INCB    NSPLDV          ;Count # of actual spooled devices
010656  000432                        BR      1$              ;Process next device
010660  010100                14$:    MOV     R1,R0           ;Get address of SDCB
010662  062700  0000000G                ADD     #SDCHAN,R0      ;Point to channel block within SDCB
010666  004737  024530'                CALL    OPNCHN          ;Set TSX-Plus channel block open to device
010672  103403                        BCS     3$              ;Br if did not recognize device
010674  105237  0000000G                INCB    NSPLDV          ;Count # actual spooled devices
010700  000421                        BR      1$              ;GO PROCESS NEXT DEVICE
                                ;
                                ;  Error on opening spooled device
                                ;  Determine if we should print an error message or simply
                                ;  mark the spooled device as unavailable.
                                ;
010702  012761  0000000G 0000000G 3$:     MOV     #DMYDEV,SDNAME(R1);SAY THIS DEVICE IS NOT SPOOLED
```

```
     58 010710  105737  000000G                 TSTB    VINABT          ;ABORT OR CONTINUE ON ERRORS?
     59 010714  001413                          BEQ     1$              ;BR TO IGNORE DEVICE AND CONTINUE INIT
     60 010716                                  .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     61 010724                                  .PRINT  #BDSPOP
     62 010732  010200                          MOV     R2,R0           ;GET RAD50 DEVICE NAME
     63 010734  004737  025502'                 CALL    PRTR50          ;PRINT DEVICE NAME
     64 010740  000137  004250'                 JMP     INISTP          ;ABORT INITIALIZATION
     65                                 ;
     66                                 ;  Process next spooled device
     67                                 ;
     68 010744  062701  000000G         1$:     ADD     #SDCBSZ,R1      ;POINT TO NEXT SDCB
     69 010750  005723                          TST     (R3)+           ;POINT TO NEXT DEVICE NAME
     70 010752  020327  000000G                 CMP     R3,#SPLDVN      ;OPENED ALL SPOOLED DEVICES?
     71 010756  103703                          BLO     2$              ;BR IF MORE TO DO
     72                                 ;
     73                                 ;  Open the spool file
     74                                 ;
     75 010760  105737  000000G                 TSTB    NSPLDV          ;ARE THERE ANY ACTUAL SPOOLED DEVICES?
     76 010764  001521                          BEQ     12$             ;BR IF THERE ARE NO ACTUAL SPOOLED DEVICES
     77 010766  013700  000000G                 MOV     SPLBLK,R0       ;Get name of device for spool file
     78 010772  004737  025176'                 CALL    RTFTCH          ;Fetch the RT-11 device handler
     79 010776  103532                          BCS     11$             ;Br if cannot fetch handler
     80 011000  013702  000000G                 MOV     NSPLBL,R2       ;GET # BLOCKS TO ALLOCATE FOR FILE
     81 011004  005202                          INC     R2              ;Add 1 extra block
     82                                 ;
     83                                 ;  See if spool file already exists
     84                                 ;
     85 011006                          5$:     .LOOKUP #AREA,#1,#SPLBLK;SEE IF SPOOL FILE ALREADY EXISTS
     86 011026  103415                          BCS     6$              ;BR IF IT DOES NOT EXIST
     87 011030  020002                          CMP     R0,R2           ;IS IT THE RIGHT SIZE?
     88 011032  001453                          BEQ     7$              ;BR IF YES
     89 011034                                  .CLOSE  #1              ;IT EXISTS BUT IS OF WRONG SIZE
     90 011042                                  .DELETE #AREA,#1,#SPLBLK;DELETE CURRENT FILE AND OPEN NEW ONE
     91                                 ;
     92                                 ;  Open new spool file
     93                                 ;
     94 011062                          6$:     .ENTER  #AREA,#1,#SPLBLK,R2;CREATE A NEW SPOOL FILE
     95 011106  103456                          BCS     8$              ;BR IF ERROR ON ENTER
     96                                 ;  Write to last block in file to reserve full file space
     97 011110  010203                          MOV     R2,R3           ;Get # of blocks in file
     98 011112  005303                          DEC     R3              ;Get # of last block in file
     99 011114                                  .WRITW  #AREA,#1,#TSINIT,#256.,R3
    100                                 ;  Now close and reopen using a lookup
    101 011152                                  .CLOSE  #1              ;CLOSE SPOOL FILE
    102 011160  000712                          BR      5$              ;GO BACK AND REOPEN USING LOOKUP
    103                                 ;
    104                                 ;  Spool file has been successfully opened with a lookup.
    105                                 ;  Save the channel status.
    106                                 ;
    107 011162  012700  000000G         7$:     MOV     #SPLCHN,R0      ;SAVE CHANNEL STATUS HERE
    108 011166  013702  000000G                 MOV     SPLBLK,R2       ;GET DEVICE NAME
    109 011172  004737  024616'                 CALL    SETCHN          ;SAVE CHANNEL STATUS
    110 011176                                  .RELEAS #SPLBLK         ;Release the RT-11 device handler
    111                                 ;
    112                                 ;  Set number of free public blocks in spool file
    113                                 ;
    114 011206  113703  000000G                 MOVB    NSPLDV,R3       ;Get # spooled devices
```

```
    115 011212  070327  0000000                 MUL     #PVSPBL,R3      ;Times number of private blocks per dev
    116 011216  005403                          NEG     R3
    117 011220  063703  0000000                 ADD     NSPLBL,R3               ;Get # public spool blocks
    118 011224  010337  0000000                 MOV     R3,NFRESB       ;This is number of public free spool blocks
    119                             ;
    120                             ;  Finished
    121                             ;
    122 011230  012605              12$:        MOV     (SP)+,R5
    123 011232  012604                          MOV     (SP)+,R4
    124 011234  012603                          MOV     (SP)+,R3
    125 011236  012602                          MOV     (SP)+,R2
    126 011240  012601                          MOV     (SP)+,R1
    127 011242  000207              9$:         RETURN
    128                             ;
    129                             ;  Error: Cannot open spool file.
    130                             ;
    131 011244                      8$:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
    132 011252                                  .PRINT  #BOSF
    133 011260  000137  004250'                 JMP     INISTP          ;ABORT INITIALIZATION
    134                             ;
    135                             ;  Error: Invalid device specification.
    136                             ;
    137 011264  010001              11$:        MOV     R0,R1           ;Save device name
    138 011266                                  .PRINT  #TSXHD          ;Print error message
    139 011274                                  .PRINT  #BOSF
    140 011302  004737  010170'                 CALL    BADDEV          ;Print invalid device specification
```

```
                                        .SBTTL  SPLCLD -- Set up spooling to a CL device
                                ;-----------------------------------------------------------------------
                                ;  SPLCLD is called to set up a spool device control block when
                                ;  spooling is being directed to a Communication Line (CL) device.
                                ;
                                ;  Inputs:
                                ;   R0 = CL unit number
                                ;   R1 = Address of SDCB
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Invalid CL unit
                                ;
011306  010546                  SPLCLD: MOV     R5,-(SP)
                                ;
                                ;  Make sure CL unit number is valie
                                ;
011310  010005                          MOV     R0,R5           ;Get CL unit number
011312  020527  0000000                 CMP     R5,#CLTOTL      ;Is this a valid unit #
011316  103022                          BHIS    8$              ;Br if invalid
                                ;
                                ;  Set up channel control block in SDCB
                                ;
011320  005061  000000C                 CLR     SDCHAN+C.SBLK(R1);Starting block # = 0
011324  020527  000007                  CMP     R5,#7           ;Is this a CL or C1 unit?
011330  101405                          BLOS    1$              ;Br if CL unit
011332  162705  000010                  SUB     #8.,R5          ;Remove C1 unit # bias
011336  013700  0000000                 MOV     C1DEVX,R0       ;Get C1 device index number
011342  000402                          BR      2$
011344  013700  0000000         1$:     MOV     CLDEVX,R0       ;Get CL device index number
011350  010061  000000C         2$:     MOV     R0,SDCHAN+C.CSW(R1);Set device index number
011354  110561  000000C                 MOVB    R5,SDCHAN+C.DEVQ(R1);Set unit #
                                ;
                                ;  We successfully set up a CL unit
                                ;
011360  000241                          CLC                     ;Signal success on error
011362  000401                          BR      9$
                                ;
                                ;  We cannot open this CL unit
                                ;
011364  000261                  8$:     SEC                     ;Signal error
                                ;
                                ;  Finished
                                ;
011366  012605                  9$:     MOV     (SP)+,R5
011370  000207                          RETURN
```

```
                                        .SBTTL  CHKCLD -- See if a device name is a CL or C1 unit
                                ;-----------------------------------------------------------------------
                                ;  Determine if a rad50 device and unit name is a CL or C1 device.
                                ;
                                ;  Inputs:
                                ;   R0 = Rad50 device spec
                                ;
                                ;  Outputs:
                                ;   C-flag set      ==> Not a CL or C1 unit
                                ;   C-flag cleared ==> This is a CL or C1 unit
                                ;   R0 = CL unit number (0-15)
                                ;
011372                          CHKCLD:
                                ;
                                ;  See if this is a CL unit
                                ;
011372  020037  000202'                 CMP     R0,R50CL        ;Is name "CL"?
011376  001411                          BEQ     1$              ;Br if yes
011400  020037  000204'                 CMP     R0,R50CL0       ;Is name in the range CL0 to CL7?
011404  103432                          BLO     8$              ;Br if not
011406  020037  000206'                 CMP     R0,R50CL7
011412  101005                          BHI     2$
011414  163700  000204'                 SUB     R50CL0,R0       ;Get CL unit number
011420  000422                          BR      7$
011422  005000                  1$:     CLR     R0              ;CL = CL0
011424  000420                          BR      7$
                                ;
                                ;  See if this is a C1 unit
                                ;
011426  020037  000210'         2$:     CMP     R0,R50C1        ;Is name "C1"?
011432  001413                          BEQ     3$              ;Br if yes
011434  020037  000212'                 CMP     R0,R50C10       ;Is name in the range C10 to C17?
011440  103414                          BLO     8$              ;Br if not
011442  020037  000214'                 CMP     R0,R50C17
011446  101011                          BHI     8$
011450  163700  000212'                 SUB     R50C10,R0       ;Get unit number
011454  062700  000010                  ADD     #8.,R0          ;Bias by 8 for C1 units
011460  000402                          BR      7$
011462  012700  000010          3$:     MOV     #8.,R0          ;C1 = CL8
                                ;
                                ;  This is a CL or C1 unit
                                ;
011466  000241                  7$:     CLC                     ;Signal success on return
011470  000401                          BR      9$
                                ;
                                ;  This is not a CL or C1 unit
                                ;
011472  000261                  8$:     SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
011474  000207                  9$:     RETURN
```

```
                                        .SBTTL  CVTDVU -- Convert device name to dev index and unit #
                                ;-----------------------------------------------------------------------
                                ;  CVTDVU is called to convert a RAD50 device name into the corresponding
                                ;  device index number and unit number.
                                ;
                                ;  Inputs:
                                ;   R0 = RAD50 device name.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Conversion successful.
                                ;   C-flag set     ==> Unable to find device name in tables.
                                ;   R0 = Device index number (low byte), device unit number (high byte).
                                ;
011476  010246                  CVTDVU: MOV     R2,-(SP)
011500  010346                          MOV     R3,-(SP)
                                ;
                                ;  Split the unit number off of the full device name
                                ;
011502  010003                          MOV     R0,R3           ;Get full device name
011504  005002                          CLR     R2              ;Set up for divide
011506  071227  000050                  DIV     #50,R2          ;Split name and unit (R0=name, R1=unit)
011512  005703                          TST     R3              ;Was a unit number specified?
011514  001402                          BEQ     1$              ;Br if not
011516  162703  000036                  SUB     #36,R3          ;Convert unit number to binary value
011522  010300                  1$:     MOV     R3,R0           ;Get unit number
011524  000300                          SWAB    R0              ;Position to high-order byte
                                ;
                                ;  Look up the device name to get the device index
                                ;
011526  070227  000050                  MUL     #50,R2          ;Now get the device name without unit number
011532  013702  000000G                 MOV     NUMDEV,R2       ;Get index number of last device
011536  020362  000000G         2$:     CMP     R3,PNAME(R2)    ;Search for device in name table
011542  001407                          BEQ     3$              ;Br if found it
011544  162702  000002                  SUB     #2,R2           ;Try next device
011550  002372                          BGE     2$              ;Loop if more to check
                                ;
                                ;  Error, cannot find device name in tables
                                ;
011552  012700  177777                  MOV     #-1,R0          ;Set device # = unit # = -1
011556  000261                          SEC                     ;Signal error on return
011560  000402                          BR      9$
                                ;
                                ;  Found the device in the tables
                                ;
011562  050200                  3$:     BIS     R2,R0           ;Combine device # and unit #
011564  000241                          CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
011566  012603                  9$:     MOV     (SP)+,R3
011570  012602                          MOV     (SP)+,R2
011572  000207                          RETURN
```

```
                                        .SBTTL  FORCE0 -- Force a 2-char dev name to unit 0
                                ;-----------------------------------------------------------------------
                                ;  Inputs: R3 points to a RAD50 device name
                                ;
                                ;  Outputs: If the 3rd char of the device name pointed to by R3 is
                                ;       blank, then it is changed to 0
                                ;
011574  010346                  FORCE0: MOV     R3,-(SP)
011576  010446                          MOV     R4,-(SP)
011600  010546                          MOV     R5,-(SP)
011602  011305                          MOV     (R3),R5         ;MOVE CURRENT DEV NAME TO R5
011604  005004                          CLR     R4              ;SET UP FOR DIVIDE
011606  071427  000050                  DIV     #50,R4          ;SEPARATE INTO NAME AND UNIT
011612  005705                          TST     R5              ;WAS 3RD CHAR BLANK?
011614  001012                          BNE     9$              ;RETURN IF NOT
011616  010405                          MOV     R4,R5           ;GET HIGH 2 CHARS
011620  005004                          CLR     R4              ;SET UP FOR ANOTHER DIVIDE
011622  071427  000050                  DIV     #50,R4          ;SEPARATE 1 & 2 CHARS
011626  005704                          TST     R4              ;WAS CHAR 1 BLANK?
011630  001404                          BEQ     9$              ;EMPTY OR INVALID DEV NAME!
011632  005705                          TST     R5              ;WAS CHAR 2 BLANK?
011634  001402                          BEQ     9$              ;1-CHAR DEV NAME SHOULD BE INVALID???
011636  062713  000036                  ADD     #^R  0,(R3)     ;FORCE TO UNIT 0
011642  012605                  9$:     MOV     (SP)+,R5
011644  012604                          MOV     (SP)+,R4
011646  012603                          MOV     (SP)+,R3
011650  000207                          RETURN
```

```
                                        .SBTTL  ALOCBF -- Allocate buffer space
                                ;-----------------------------------------------------------------------
                                ;  ALOCBF is called to allocate space for buffers.  The allocated space
                                ;  is not initialized but simply reserved.
                                ;
                                ;  Inputs:
                                ;   R5 = Start of area to allocate buffer space in.
                                ;
                                ;  Outputs:
                                ;   R5 = Address beyond end of buffer area.
                                ;   CHNBAS = Address of base of I/O channel space.
                                ;   CHNEND = Address past end of I/O channel space.
                                ;
011652  010146                  ALOCBF: MOV     R1,-(SP)
                                ;
                                ;  Assign space for I/O queue elements.
                                ;
011654  010537  0000000G                MOV     R5,FREIOQ       ;START OF I/O QUEUE SPACE
011660  062705  000000C                 ADD     #IOQSIZ*NUMIOQ,R5;RESERVE SPACE FOR I/O QUEUE ELEMENTS
                                ;
                                ;  Assign space for shared PLAS region control blocks
                                ;
011664  010537  0000000G                MOV     R5,SHRRCB       ;Start of area for RCB's
011670  013701  0000000G                MOV     VNGR,R1         ;Get number of RCB's wanted
011674  020137  0000000G                CMP     R1,VMXWIN       ;Must have one for each display window
011700  103002                          BHIS    13$             ;Br if ok
011702  013701  0000000G                MOV     VMXWIN,R1       ;Force one for each window
011706  070127  0000000G        13$:    MUL     #RC$$SZ,R1      ;Multiply by size of each block
011712  060105                          ADD     R1,R5           ;Allocate space for RCB's
011714  010537  0000000G                MOV     R5,SHRRCN       ;Address of end of region
                                ;
                                ;  Assign space for fork blocks
                                ;
011720  012700  000000C                 MOV     #<NUMFRK-FRKGEN>,R0 ;Get # fork blocks to allocate
011724  003404                          BLE     11$             ;Br if none to allocate
011726  010537  0000000G                MOV     R5,FRKINI       ;Set pointer to start of area
011732  062705  000000C                 ADD     #<<NUMFRK-FRKGEN>*FQ$$SZ>,R5 ;Reserve space for fork blocks
                                ;
                                ;  Assign space for job monitoring control blocks
                                ;
011736  013701  0000000G        11$:    MOV     VMXMON,R1       ;Any job monitoring blocks wanted?
011742  001405                          BEQ     10$             ;Br if not
011744  010537  0000000G                MOV     R5,MONFQH       ;Start of job monitoring control blocks
011750  070127  0000000G                MUL     #JM$$SZ,R1      ;Compute space needed for control blocks
011754  060105                          ADD     R1,R5           ;Allocate the space
                                ;
                                ;  Allocate space for system message buffers.
                                ;
011756  010537  0000000G        10$:    MOV     R5,SNMSHD       ;HEAD OF SYSTEM MESSAGE BUFFER AREA
011762  062705  000000C                 ADD     #<NMSNMB*SB$$SZ>,R5;RESERVE ROOM FOR MESSAGE BUFFERS
                                ;
                                ;  Allocate space for INSTALLed program table
                                ;
011766  010537  0000000G                MOV     R5,INSTBL       ;Base of table
011772  013701  0000000G                MOV     VNUIP,R1        ;# slots for user installed programs
011776  062701  0000000G                ADD     #NSIP,R1        ;Add # slots for system programs
012002  070127  0000000G                MUL     #II$$SZ,R1      ;Multiply by size of each slot
```

```
                                ;
                                ;  Allocate space for device mount entries
                                ;
012014  010537  0000000G                MOV     R5,CSHDEV       ;Point to start of area
012020  012701  0000000G                MOV     #CD$$SZ,R1      ;Get size of each entry
012024  070137  0000000G                MUL     VMXCSH,R1       ;Multiply by number of entries
012030  060105                          ADD     R1,R5           ;Reserve space for table
012032  010537  0000000G                MOV     R5,CSHDVN       ;Save pointer past end of table
                                ;
                                ;  Allocate space for data cache control blocks
                                ;
012036  005737  0000000G                TST     CSHALC          ;Is data caching wanted?
012042  001404                          BEQ     12$             ;Br if not
012044  010537  0000000G                MOV     R5,CCBHD        ;Head of free list area
012050  062705  000000C                 ADD     #NUMCCB*CC$$SZ,R5 ;Allocate space for control blocks
                                ;
                                ;  Allocate space for spool file control blocks
                                ;
012054  013701  0000000G        12$:    MOV     NSPLFL,R1       ;Get # spool file control blocks needed
012060  001407                          BEQ     1$              ;Br if none needed
012062  070127  0000000G                MUL     #SFCBSZ,R1      ;Compute space needed by control blocks
012066  010537  0000000G                MOV     R5,SFCB         ;Base of control block area
012072  060105                          ADD     R1,R5           ;Allocate space for control blocks
012074  010537  0000000G                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.
                                ;
012100  012701  000002          1$:     MOV     #2,R1           ;START WITH FIRST MUX
012104  020127  0000000G        2$:     CMP     R1,#LSTMX       ;HAVE WE DONE ALL MUX'S?
012110  101007                          BHI     5$              ;BR IF YES
012112  010561  0000000G                MOV     R5,MXLNT(R1)    ;SET ADDRESS OF START OF VECTOR
012116  062705  000020                  ADD     #16.,R5         ;RESERVE SPACE FOR VECTOR
012122  062701  000002                  ADD     #2,R1           ;ADVANCE TO NEXT MUX
012126  000766                          BR      2$
                                ;
                                ;  Allocate buffers to hold characters for DMA transfers to DH11 multiplexers
                                ;
012130  012701  000002          5$:     MOV     #2,R1           ;Start with first line
012134  020127  0000000G        6$:     CMP     R1,#LSTPL       ;Is this a primary time-sharing line?
012140  101403                          BLOS    3$              ;Br if yes
012142  020127  0000000G                CMP     R1,#FSTIOL      ;Is this a CL line?
012146  103422                          BLO     7$              ;Br if not
012150  026127  0000000G 0000000G 3$:    CMP     LCDTYP(R1),#CDX$DH ;Is this line connected to a DH11?
012156  001404                          BEQ     8$              ;Br if yes
012160  026127  0000000G 0000000G        CMP     LCDTYP(R1),#CDX$VH ;Is this line connected to a DHV11?
012166  001012                          BNE     7$              ;Br if not
012170  010561  0000000G        8$:     MOV     R5,LDHB1B(R1)   ;Set address of start of buffer 1
012174  010561  0000000G                MOV     R5,LDHB1P(R1)   ;Initialize pointer into buffer 1
012200  062705  0000000G                ADD     #DHBFSZ,R5      ;Reserve space for buffer
012204  010561  0000000G                MOV     R5,LDHB2B(R1)   ;Set address of start of buffer 2
012210  062705  0000000G                ADD     #DHBFSZ,R5      ;Reserve space for buffer
012214  062701  000002          7$:     ADD     #2,R1           ;Get # of next line
012220  020127  0000000G                CMP     R1,#LSTHL       ;Have we checked all lines?
```

```
    115 012224  101743                          BLOS    6$              ;Br if not
    116 012226  005205                          INC     R5              ;Bound up to next word
    117 012230  042705  000001                  BIC     #1,R5
    118                                 ;
    119                                 ;  Allocate space for tables that keep track of space in swap file
    120                                 ;
    121 012234  105737  0000000G                TSTB    VSWPFL          ;Is this a swapping system?
    122 012240  001415                          BEQ     14$             ;Br if not
    123 012242  013700  0000000G                MOV     VSWPSL,R0       ;Get # slots in swap file
    124 012246  006300                          ASL     R0              ;Allocate 2 bytes per slot
    125 012250  010537  0000000G                MOV     R5,SWPPOS       ;Start of table with starting block #'s
    126 012254  060005                          ADD     R0,R5           ;Allocate space
    127 012256  010537  0000000G                MOV     R5,SWPJOB       ;Start of table with job #'s
    128 012262  060005                          ADD     R0,R5           ;Allocate space
    129 012264  010537  0000000G                MOV     R5,SCPFHD       ;Pointer to area with command packets
    130 012270  062705  000000C                 ADD     #NSCP*SP$$SZ,R5 ;Allocate space for swap command packets
    131                                 ;
    132                                 ;  Allocate a 512-byte buffer to use to access job context blocks
    133                                 ;
    134 012274  010537  0000000G        14$:    MOV     R5,CXTBUF       ;Set address of buffer
    135 012300  062705  001400                  ADD     #1400,R5        ;Reserve space for the buffer
    136                                 ;
    137                                 ;  Make sure TSX is not too big.
    138                                 ;
    139 012304  010500                          MOV     R5,R0           ;GET CURRENT MEMORY ADDRESS
    140 012306  004737  025312'                 CALL    CHKMEM          ;CHECK FOR SPACE OVERFLOW
    141                                 ;
    142                                 ;  Finished
    143                                 ;
    144 012312  012601                          MOV     (SP)+,R1
    145 012314  000207                          RETURN
```

```
                                        .SBTTL  ALCSLO -- Allocate silo buffers for lines
                                ;-----------------------------------------------------------------------
                                ;  Allocate the silo buffers that are used to hold characters as they
                                ;  are received from serial lines.
                                ;
                                ;  Inputs:
                                ;   R5 = Current pointer to start of free memory.
                                ;
                                ;  Outputs:
                                ;   R5 = New pointer to start of free memory.
                                ;
012316  010146                  ALCSLO: MOV     R1,-(SP)
012320  010246                          MOV     R2,-(SP)
                                ;
                                ;  Begin loop to check each line
                                ;
012322  012701  000000G                 MOV     #LSTHL,R1       ;Get index to last hardware line
                                ;
                                ;  Only allocate silo buffers for real lines
                                ;
012326  012702  000040          1$:     MOV     #32.,R2         ;Set minimum size for time-sharing lines
012332  020127  000000G                 CMP     R1,#LSTPL       ;Is this a primary line?
012336  101405                          BLOS    2$              ;Br if yes
012340  020127  000000G                 CMP     R1,#FSTIOL      ;Is this a CL line?
012344  103463                          BLO     9$              ;Br if not
012346  012702  000020                  MOV     #16.,R2         ;Set minimum size for CL lines
                                ;
                                ;  Determine how much space to allocate
                                ;
012352  016100  000000G         2$:     MOV     LHIRBA(R1),R0   ;Get requested size
012356  001002                          BNE     8$              ;Br if a size was specified
012360  013700  000000G                 MOV     VNCSLO,R0       ;Use default value
012364  020027  000000G         8$:     CMP     R0,#MAXSLO      ;Constrain size to 255 bytes
012370  101402                          BLOS    3$              ;Br if ok
012372  012700  000000G                 MOV     #MAXSLO,R0      ;Reduce size
012376  020002                  3$:     CMP     R0,R2           ;Compare with acceptable minimum
012400  103001                          BHIS    4$              ;Br if ok
012402  010200                          MOV     R2,R0           ;Use minimum size allowed
012404  010061  000000G         4$:     MOV     R0,LHIRBA(R1)   ;Set # bytes to be allocated for silo
                                ;
                                ;  Allocate the space
                                ;
012410  010561  000000G                 MOV     R5,LHIRBB(R1)   ;Set base address of buffer
012414  010561  000000G                 MOV     R5,LHIRBP(R1)   ;Init pointer where to store next char
012420  010561  000000G                 MOV     R5,LHIRBG(R1)   ;Init pointer where to get next char
012424  010061  000000G                 MOV     R0,LHIRBS(R1)   ;Set # free bytes in buffer
012430  060005                          ADD     R0,R5           ;Allocate space for buffer
012432  010561  000000G                 MOV     R5,LHIRBE(R1)   ;Set address beyond end of buffer
                                ;
                                ;  Set up control information about when to send XON and XOFF
                                ;
012436  006200                          ASR     R0              ;Get 1/2 of buffer size
012440  162700  000002                  SUB     #2,R0           ;Minus two characters
012444  116102  000000G                 MOVB    LHIRBC(R1),R2   ;Get specified size for XOFF point
012450  001002                          BNE     5$              ;Br if value specified
012452  113702  000000G                 MOVB    VNCXOF,R2       ;Try default
012456  020200                  5$:     CMP     R2,R0           ;Is specified value ok?
```

```
     58 012460  101401                          BLOS    6$                ;Br if yes
     59 012462  010002                          MOV     R0,R2             ;No, use size/2-2
     60 012464  110261  000000G          6$:    MOVB    R2,LHIRBC(R1)     ;Set # of chars when XOFF sent
     61 012470  116102  000001G                 MOVB    LHIRBC+1(R1),R2 ;Get specified size for XON point
     62 012474  001002                          BNE     7$                ;Br if a value was specified
     63 012476  113702  000000G                 MOVB    VNCXON,R2         ;Try default
     64 012502  020200                   7$:    CMP     R2,R0             ;Is specified value ok?
     65 012504  101401                          BLOS    10$               ;Br if ok
     66 012506  010002                          MOV     R0,R2             ;No, use size/2-2
     67 012510  110261  000001G          10$:   MOVB    R2,LHIRBC+1(R1) ;Set # of chars when XON sent
     68                                     ;
     69                                     ;  Do the next line
     70                                     ;
     71 012514  162701  000002           9$:    SUB     #2,R1             ;Get next line index number
     72 012520  003302                          BGT     1$                ;Loop if more to do
     73                                     ;
     74                                     ;  Finished
     75                                     ;
     76 012522  005205                          INC     R5                ;Force R5 to be even
     77 012524  042705  000001                  BIC     #1,R5
     78 012530  012602                          MOV     (SP)+,R2
     79 012532  012601                          MOV     (SP)+,R1
     80 012534  000207                          RETURN
```

```
                                        .SBTTL  ALBFX  -- Allocate buffers in extended memory region
                                ;-----------------------------------------------------------------------
                                ;  ALBFX is called to allocate space for buffers that are not constrained
                                ;  to fit in the 40Kb region that TSX-Plus occupies.
                                ;
                                ;  Inputs:
                                ;   R5 = 64-Byte address of base of free memory region.
                                ;
                                ;  Outputs:
                                ;   R5 = Address above top of buffers allocated.
                                ;
012536  010146                  ALBFX:  MOV     R1,-(SP)
012540  010246                          MOV     R2,-(SP)
012542  010346                          MOV     R3,-(SP)
                                ;
                                ;  Allocate character buffers for all lines
                                ;  Note: Character buffer space will be accessed by mapping through PAR 6.
                                ;
012544  012701  000002                  MOV     #2,R1           ;GET 1ST JOB INDEX NUMBER
012550  032761  000000G 000000G 3$:      BIT     #$DEAD,LSW3(R1) ;IS THIS LINE INSTALLED?
012556  001047                          BNE     2$              ;BR IF NOT -- DON'T ALLOCATE ANY BUFFER SPACE
012560  020127  000000G                 CMP     R1,#FSTDL       ;IS THIS A DETACHED JOB LINE?
012564  103403                          BLO     1$              ;BR IF NOT
012566  020127  000000G                 CMP     R1,#LSTDL       ;DETACHED JOB LINE?
012572  101441                          BLOS    2$              ;BR IF DETACHED JOB -- DON'T ALLOCATE  BUFFERS
012574  010561  000000G         1$:     MOV     R5,LTTPAR(R1)   ;SET PHYSICAL MEMORY PAR OFFSET FOR BUFFER
012600  012702  000000G                 MOV     #VPAR6,R2       ;GET VIRTUAL MEMORY ADDRESS FOR BASE OF PAR6
012604  010261  000000G                 MOV     R2,LINBUF(R1)   ;INPUT BUFFER STARTS AT BASE OF PAR6 REGION
012610  016100  000000G                 MOV     LINSIZ(R1),R0   ;GET # BYTES FOR INPUT BUFFER
012614  010061  000000G                 MOV     R0,LINSPC(R1)   ;SET # FREE BYTES IN INPUT BUFFER
012620  060002                          ADD     R0,R2           ;ADVANCE VIRTUAL ADDRESS
012622  010261  000000G                 MOV     R2,LINEND(R1)   ;POINTS PAST END OF INPUT BUFFER
012626  010003                          MOV     R0,R3           ;Get # bytes in input buffer
012630  062703  000007                  ADD     #7,R3           ;Bound up to multiple of 8
012634  072327  177775                  ASH     #-3,R3          ;Get # bytes needed in activation-flag buffer
012640  060302                          ADD     R3,R2           ;Reserve space for activation-flag buffer
012642  060300                          ADD     R3,R0           ;Accumulate total buffer space
012644  010261  000000G                 MOV     R2,LOTBUF(R1)   ;POINTS TO START OF OUTPUT BUFFER AREA
012650  066100  000000G                 ADD     LOTSIZ(R1),R0   ;ACCUMULATE # BYTES IN BOTH BUFFERS
012654  066102  000000G                 ADD     LOTSIZ(R1),R2   ;ADVANCE VIRTUAL ADDRESS COUNTER
012660  010261  000000G                 MOV     R2,LOTEND(R1)   ;SAVE ADDRESS OF END OF OUTPUT BUFFER
012664  062700  000077                  ADD     #77,R0          ;BOUND UP TO MULTIPLE OF 64 BYTES
012670  072027  177772                  ASH     #-6,R0          ;CONVERT TO # 64-BYTE BLOCKS ALLOCATED
012674  060005                          ADD     R0,R5           ;ADVANCE PHYSICAL MEMORY PAR ADDRESS
012676  062701  000002          2$:     ADD     #2,R1           ;ADVANCE LINE NUMBER
012702  020127  000000G                 CMP     R1,#LSTSL       ;HAVE WE DONE ALL LINES YET?
012706  101720                          BLOS    3$              ;BR IF MORE TO DO
                                ;
                                ;  Allocate space for shared file record locking data structures
                                ;
012710  005737  000000G                 TST     VMXSF           ;Shared file support wanted?
012714  001451                          BEQ     5$              ;Br if not
012716  013737  000000G 000000G         MOV     VNUMDC,NUMDCD   ;Set number of data cache blocks
012724  013737  000000G 000000G         MOV     VMXSFC,NUMCDB   ;Set number of free CDB's
012732  010537  000000G                 MOV     R5,LOKMEM       ;Set phys address of base of area
012736  012701  000000G                 MOV     #FF$$SZ,R1      ;Size of an FDB
012742  070137  000000G                 MUL     VMXSF,R1        ;Times number of FDB's
```

```
     58 012746  062701  000000G             ADD     #<NLINES*FW$$SZ>,R1 ;Space needed for wait blocks
     59 012752  013703  0000000             MOV     VMLBLK,R3           ;Max blocks a CDB may hold locked
     60 012756  006303                      ASL     R3                  ;Two bytes per entry
     61 012760  062703  0000000             ADD     #FC$LBN,R3          ;Add base size of a CDB
     62 012764  070337  0000000             MUL     VMXSFC,R3           ;Times number of shared file channels
     63 012770  060301                      ADD     R3,R1               ;Accumulate space needed
     64 012772  012703  0000000             MOV     #DC$$SZ,R3          ;Size of a data cache descriptor
     65 012776  070337  0000000             MUL     VNUMDC,R3           ;Times number of data cache entries
     66 013002  060301                      ADD     R3,R1               ;Reserve space for data cache descriptors
     67 013004  062701  000100              ADD     #64.,R1             ;Bound up to 64 byte unit
     68 013010  072127  177772              ASH     #-6,R1              ;Convert to # 64 byte units
     69 013014  042701  176000              BIC     #176000,R1          ;Clear sign extension
     70 013020  060105                      ADD     R1,R5               ;Reserve space for data structures
     71 013022  010537  0000000             MOV     R5,LOKCSH           ;Save pointer to start of cache buffer area
     72 013026  013701  0000000             MOV     VNUMDC,R1           ;# shared-file data cache blocks wanted
     73 013032  070127  000010              MUL     #8.,R1              ;8 64-byte blocks each (512 bytes each)
     74 013036  060105                      ADD     R1,R5               ;Reserve space for data cache buffers
     75                                 ;
     76                                 ;  Allocate space for mapped I/O buffers
     77                                 ;
     78 013040  105737  0000000         5$:     TSTB    MIOFLG              ;Are any mapped I/O buffers needed?
     79 013044  001415                          BEQ     7$                  ;Br if not
     80 013046  013701  0000000             MOV     MIOBHD,R1           ;Point to 1st mapped I/O control block
     81 013052  001412                      BEQ     7$                  ;Br if no more buffers needed
     82 013054  010561  0000000         6$:     MOV     R5,MI$SBP(R1)       ;Set address of base of buffer
     83 013060  113703  0000000             MOVB    VMIOSZ,R3           ;Get # blocks for buffer
     84 013064  072327  000003              ASH     #3,R3               ;Convert to # 64-byte pages
     85 013070  060305                      ADD     R3,R5               ;Allocate space for buffer
     86 013072  016101  0000000             MOV     MI$LNK(R1),R1       ;Get address of next control block
     87 013076  001366                      BNE     6$                  ;Loop if more to allocate
     88                                 ;
     89                                 ;  Allocate space for performance monitor data buffer if it is wanted.
     90                                 ;
     91 013100  012701  0000000         7$:     MOV     #PMSIZE,R1          ;DID USER GEN IN PERFORMANCE MONITOR FEATURE?
     92 013104  001407                          BEQ     9$                  ;BR IF NOT
     93 013106  010537  0000000             MOV     R5,PMPAR            ;SET BASE ADDRESS OF PM BUFFER
     94 013112  062701  000077              ADD     #77,R1              ;BOUND SIZE UP TO 64-BYTE MULTIPLE
     95 013116  072127  177772              ASH     #-6,R1              ;CONVERT TO # 64-BYTE BLOCKS
     96 013122  060105                      ADD     R1,R5               ;ADVANCE FREE MEMORY POINTER
     97                                 ;
     98                                 ;  Finished
     99                                 ;
    100 013124  012603                 9$:     MOV     (SP)+,R3
    101 013126  012602                         MOV     (SP)+,R2
    102 013130  012601                         MOV     (SP)+,R1
    103 013132  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.
                                ;
013134  010246                  OPNKMN: MOV     R2,-(SP)
                                ;
                                ;  Lookup TSKMON file.
                                ;
013136                                  .LOOKUP #AREA,#1,#KMNNAM ;TRY TO FILE KMON SAV FILE
013156  103517                         BCS     9$              ;BR IF NOT THERE
                                ;
                                ;  Read block 0 of save file and extract some information.
                                ;
013160  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
013164                                  .READW  #AREA,#1,R2,#256.,#0 ;READ BLOCK 0 OF SAV FILE
                                ;  Determine size of kmon
013220  016200  000050                  MOV     50(R2),R0       ;GET TOP ADDRESS OF KMON
013224  062700  000003                  ADD     #3,R0           ;BOUND UP TO NEXT WORD
013230  042700  000001                  BIC     #1,R0           ;FORCE EVEN
013234  010037  000000G                 MOV     R0,KMNTOP
                                ;  Determine number of 256-word memory pages needed while kmon running.
013240  162700  000000G                 SUB     #KMNBAS,R0      ;BASE ADDRESS OF KMON
013244  010037  000000G                 MOV     R0,KMNHI        ;TOP OF TSKMON - KMNBAS
013250  062700  000777                  ADD     #511.,R0        ;BOUND UP TO PAGE SIZE
013254  000241                         CLC
013256  006000                         ROR     R0              ;CVT TO # WORDS
013260  000300                         SWAB    R0              ;CVT TO # PAGES
013262  042700  177400                 BIC     #^C377,R0
013266  063700  000000G                 ADD     CXTPAG,R0       ;# PAGES NEEDED FOR JOB CONTEXT AREA
013272  010037  000000G                 MOV     R0,KMNPGS       ;# 256-wd pages needed for kmon + context area
                                ;  Determine Kmon stack pointer.
013276  016237  000042  000000G         MOV     42(R2),KMNSTK   ;INITIAL STACK POINTER FOR KMON
                                ;  Determine Kmon starting address.
013304  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)
                                ;
013312  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.
                                ;
013320  012700  000000G                 MOV     #KMNCHN,R0      ;GET KMON CHANNEL SAVE AREA
013324  013702  000072'                 MOV     KMNNAM,R2       ;GET KMON DEVICE NAME
```

```
     58 013330  004737  024616'                    CALL    SETCHN          ;SAVE CHANNEL STATUS
     59                                ;
     60                                ;  Lookup CCL.SAV and save channel status for it.
     61                                ;
     62 013334                                     .LOOKUP #AREA,#1,#CCLNAM;LOOKUP SY:CCL.SAV
     63 013354  103410                             BCS     8$              ;BR IF CAN'T FIND CCL
     64 013356  012700  000000G                    MOV     #CCLSAV,R0      ;CHANNEL SAVE AREA
     65 013362  013702  000102'                    MOV     CCLNAM,R2       ;DEVICE NAME
     66 013366  004737  024616'                    CALL    SETCHN          ;SAVE CHANNEL STATUS
     67                                ;
     68                                ;  Finished
     69                                ;
     70 013372  012602                 10$:        MOV     (SP)+,R2
     71 013374  000207                             RETURN
     72                                ;
     73                                ;  Error:  We could not find SY:CCL.SAV
     74                                ;
     75 013376                         8$:         .PRINT  #TSXHD
     76 013404                                     .PRINT  #NOCCL          ;PRINT ERROR MESSAGE
     77 013412  000137  004250'                    JMP     INISTP          ;ABORT INITIALIZATION
     78                                ;
     79                                ;  Error: We could not locate TSKMON SAV file.
     80                                ;
     81 013416                         9$:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
     82 013424                                     .PRINT  #NOKMON
     83 013432  000137  004250'                    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.
                                ;
013436  010146                  CLINIT: MOV     R1,-(SP)
013440  010246                          MOV     R2,-(SP)
013442  010346                          MOV     R3,-(SP)
                                ;
                                ;  Initialize tables for each CL unit
                                ;
013444  005003                          CLR     R3              ;Accumulate ring buffer sizes in R3
013446  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.
                                ;
013452  016102  0000000G        1$:     MOV     CL$LIX(R1),R2   ;Is this CL unit associated with a line?
013456  001416                          BEQ     5$              ;Br if not
013460  012762  0000000G 0000000G       MOV     #$SXON,LSW10(R2);Send XON when we start the line
013466  010162  0000000G                MOV     R1,LCLUNT(R2)   ;Associate the CL unit with this line
013472  005762  0000000G                TST     RSR(R2)         ;Does this unit have a specified RSR addr?
013476  001006                          BNE     5$              ;Br if yes
013500  005762  0000000G                TST     LMXNUM(R2)      ;Is this a mux line?
013504  001003                          BNE     5$              ;Br if yes
013506  005061  0000000G                CLR     CL$EPS(R1)      ;Say no endstring buffer
013512  000472                          BR      4$              ;Line is not genned in
                                ;
                                ;  Allocate and set up pointers for the output ring buffers
                                ;
013514  010561  0000000G        5$:     MOV     R5,CL$ORB(R1)   ;Start of output ring buffer
013520  010561  0000000G                MOV     R5,CL$ORP(R1)   ;Input character pointer
013524  010561  0000000G                MOV     R5,CL$ORG(R1)   ;Next available character pointer
013530  012700  0000000G                MOV     #CLORSZ,R0      ;Get default output ring buffer size
013534  005702                          TST     R2              ;Is this CL unit connected to a line?
013536  001402                          BEQ     6$              ;Br if not
013540  016200  0000000G                MOV     LOTSIZ(R2),R0   ;Get size of output ring buffer
013544  010061  0000000G        6$:     MOV     R0,CL$ORA(R1)   ;Set size of output ring buffer
013550  010061  0000000G                MOV     R0,CL$ORS(R1)   ;Available space in ring buffer
013554  060005                          ADD     R0,R5           ;Point beyond end of ring buffer
013556  010561  0000000G                MOV     R5,CL$ORE(R1)   ;Address past end of ring buffer
013562  060003                          ADD     R0,R3           ;Accumulate size of output ring buffers
                                ;
                                ;  Allocate space for end-of-file string buffer
                                ;
013564  010561  0000000G                MOV     R5,CL$EPS(R1)   ;Set pointer to end-of-file string buffer
013570  005061  0000000G                CLR     CL$EPP(R1)      ;No string to print yet
013574  062705  0000010G                ADD     #<CLEOFS+1>,R5  ;Reserve space for buffer
013600  062703  0000010G                ADD     #<CLEOFS+1>,R3  ;Accumulate buffer space
                                ;
                                ;  Initialize end-of-file form-feed count
                                ;
```

```
     58 013604  005061  000000G                    CLR     CL$EPN(R1)      ;Init ENDPAGE=0
     59                                ;
     60                                ;  Initialize option word
     61                                ;
     62 013610  012700  000000G                    MOV     #<CO$DEF>,R0    ;Get default option flags
     63 013614  005702                             TST     R2              ;Is this CL unit connected with a line?
     64 013616  001421                             BEQ     7$              ;Br if not
     65 013620  016202  000000G                    MOV     ILSW2(R2),R2    ;Get line options
     66 013624  032702  000000G                    BIT     #$TAB,R2        ;Does hardware support tabs?
     67 013630  001402                             BEQ     2$              ;Br if not
     68 013632  052700  000000G                    BIS     #CO$TAB,R0      ;Set hardware-tab flag
     69 013636  032702  000000G          2$:       BIT     #$FORM,R2       ;Does hardware support form feeds?
     70 013642  001402                             BEQ     3$              ;Br if not
     71 013644  052700  000000G                    BIS     #CO$FF,R0       ;Set hardware-form-feed flag
     72 013650  032702  000000G          3$:       BIT     #$8BIT,R2       ;Does hardware want 8 bit support?
     73 013654  001402                             BEQ     7$              ;Br if not
     74 013656  052700  000000G                    BIS     #CO$8BT,R0      ;Enable 8 bit support for CL line
     75 013662  010061  000000G          7$:       MOV     R0,CL$OPT(R1)   ;Set options for this CL line
     76                                ;
     77                                ;  Initialize page length
     78                                ;
     79 013666  012761  000102  000000G            MOV     #66.,CL$LEN(R1) ;Say page length = 66 lines
     80                                ;
     81                                ;  Initialize status flags
     82                                ;
     83 013674  005061  000000G                    CLR     CL$STA(R1)      ;Initialize status flags
     84                                ;
     85                                ;  Do next line
     86                                ;
     87 013700  162701  000002          4$:       SUB     #2,R1           ;Get index # of next unit
     88 013704  002262                             BGE     1$              ;Loop if more units to initialize
     89                                ;
     90                                ;  Make a device table entry for "CL" device
     91                                ;
     92 013706  062737  000002  000000G            ADD     #2,NUMDEV       ;One more device
     93 013714  013701  000000G                    MOV     NUMDEV,R1       ;Get device table index
     94 013720  010137  000000G                    MOV     R1,CLDEVX       ;Remember index number of CL device
     95 013724  013761  000202' 000000G            MOV     R50CL,PNAME(R1) ;Set device name ("CL")
     96 013732  012761  000000G 000000G            MOV     #CLSTS,DVSTAT(R1) ;Set dev status flags
     97 013740  012761  000000C 000000G            MOV     #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ;Device info flags
     98 013746  005061  000000G                    CLR     DEVSIZ(R1)      ;Clear device size
     99 013752  012761  000006G 000000G            MOV     #CLHEAD+6,HANENT(R1) ;Set handler entry point (4th word)
    100 013760  062703  000000G                    ADD     #CLSIZE,R3      ;Get size of handler
    101 013764  062703  000000C                    ADD     #<<CLTOTL*46.>+<NIOL*32.>>,R3 ;Add size of tables in TSGEN
    102 013770  010361  000000G                    MOV     R3,HANSIZ(R1)   ;Set size of handler
    103 013774  005205                             INC     R5              ;Make sure free-memory pointer is even
    104 013776  042705  000001                     BIC     #1,R5
    105                               ;
    106                               ;  Make a device table entry for C1 if there are more than 8 CL units
    107                               ;
    108 014002  022727  000000G 000010             CMP     #CLTOTL,#8.     ;Are there more than 8 CL units?
    109 014010  101430                             BLOS    13$             ;Br if not -- Don't need C1
    110 014012  062737  000002  000000G            ADD     #2,NUMDEV       ;One more device
    111 014020  013701  000000G                    MOV     NUMDEV,R1       ;Get device table index
    112 014024  010137  000000G                    MOV     R1,C1DEVX       ;Remember index number of CL device
    113 014030  013761  000210' 000000G            MOV     R50C1,PNAME(R1) ;Set device name ("C1")
    114 014036  012761  000000G 000000G            MOV     #CLSTS,DVSTAT(R1) ;Set dev status flags
```

```
     115 014044  012761  000000C 000000G          MOV     #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ;Device info flags
     116 014052  005061  000000G                  CLR     DEVSIZ(R1)      ;Clear device size
     117 014056  012761  000006G 000000G          MOV     #CLHEAD+6,HANENT(R1) ;Set handler entry point (4th word)
     118 014064  012761  000004  000000G          MOV     #4.,HANSIZ(R1)  ;Set size of handler
     119                                          ;
     120                                          ;  Set the version number to be returned by the .SPFUN used by
     121                                          ;  VTCOM to see if it is matched to the correct version of XL/XC.
     122                                          ;
     123 014072  105737  000000G          13$:    TSTB    CLVERS          ;Was a version specified in TSGEN?
     124 014076  001023                           BNE     9$              ;Br if yes
     125 014100  113703  000000G                  MOVB    SYSVER,R3       ;Get current RT-11 version number
     126 014104  000303                           SWAB    R3              ;Put in high order byte
     127 014106  153703  000000G                  BISB    SYSUPD,R3       ;Put update number in low-order byte
     128 014112  012702  000522'                  MOV     #CLVTBL,R2      ;Point to table with CL version numbers
     129 014116  020322                   10$:    CMP     R3,(R2)+        ;Is this entry for our version?
     130 014120  001407                           BEQ     12$             ;Br if yes
     131 014122  005722                           TST     (R2)+           ;Skip cell with CL version
     132 014124  020227  000556'                  CMP     R2,#CLVEND      ;Checked all table entries?
     133 014130  103772                           BLO     10$             ;Loop if not
     134 014132  012701  000021                   MOV     #17.,R1         ;Set default CL version if not found
     135 014136  000401                           BR      11$
     136 014140  011201                   12$:    MOV     (R2),R1         ;Get correct CL version
     137 014142  110137  000000G          11$:    MOVB    R1,CLVERS       ;Set CL version number
     138                                          ;
     139                                          ;  Finished
     140                                          ;
     141 014146  012603                   9$:     MOV     (SP)+,R3
     142 014150  012602                           MOV     (SP)+,R2
     143 014152  012601                           MOV     (SP)+,R1
     144 014154  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
                                ;
014156  010146                  LDINIT: MOV     R1,-(SP)
014160  113701  000000G                 MOVB    LDVERS,R1       ;See if value was specified in TSGEN
014164  001022                          BNE     9$              ;If value specified in TSGEN, use it
014166  010246                          MOV     R2,-(SP)
014170  010346                          MOV     R3,-(SP)
014172  013703  000000G                 MOV     SYSVER,R3       ;Get current RT version and update
014176  012701  000001                  MOV     #1,R1           ;Assume version format RT5.3 or earlier
014202  012702  000556'                 MOV     #LD1TBL,R2      ;Point to table of versions using format 1
014206  022203                  1$:     CMP     (R2)+,R3        ;Does this entry match actual version?
014210  001404                          BEQ     5$              ;Exit if so
014212  005712                          TST     @R2             ;End of table?
014214  001374                          BNE     1$              ;Keep looking if more entries
014216  012701  000002                  MOV     #2,R1           ;If not in format 1 table, use format 2
014222  110137  000000G         5$:     MOVB    R1,LDVERS       ;Remember for LD SPFUN 372
014226  012603                          MOV     (SP)+,R3
014230  012602                          MOV     (SP)+,R2
014232  012601                  9$:     MOV     (SP)+,R1
014234  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
                                ;
014236  010246                  INDINI: MOV     R2,-(SP)
014240  010346                          MOV     R3,-(SP)
                                ;
                                ;  Determine if IND support is wanted
                                ;
014242  005037  000000G                 CLR     INDSAV          ;ASSUME IND SUPPORT NOT WANTED
                                ;
                                ;  Lookup SY:IND.SAV file
                                ;
014246  013737  000000G 000264'         MOV     SYNAME,INDNAM   ;LOOK UP IND ON BOOT DEVICE
014254                                  .LOOKUP #AREA,#1,#INDNAM ;TRY TO FIND SY:IND.SAV
014274  103002                          BCC     4$              ;BR IF FOUND IND
014276  000137  014730'                 JMP     9$              ;IF CAN'T FIND IND, THEN NO IND SUPPORT
                                ;
                                ;  Set up information about IND overlay data segment
                                ;
014302  013703  000152'         4$:     MOV     WRKBUF,R3       ;Get pointer to work buffer
014306                                  .READW  #AREA,#1,R3,#256.,#0 ;READ IN BLOCK 0 OF SAV FILE
014342  016302  000064                  MOV     64(R3),R2       ;GET POINTER TO OVERLAY TABLE
014346                                  .READW  #AREA,#1,R3,#256.,#1 ;READ IN BLOCK 1 WITH OVERLAY TABLE
014404  162702  001000                  SUB     #1000,R2        ;GET ADDRESS OF OVERLAY TABLE REL TO BLOCK 1
014410  060302                          ADD     R3,R2           ;ADD BASE ADDRESS WHERE BLOCK 1 DATA IS
014412  011203                          MOV     (R2),R3         ;Get virtual address of segment 0
014414  020312                  5$:     CMP     R3,(R2)         ;Search for 1st segment with different addr
014416  001003                          BNE     6$              ;Br if found it (this is the data segment)
014420  062702  000006                  ADD     #6,R2           ;Point to overlay table entry for next seg
014424  000773                          BR      5$
014426  005722                  6$:     TST     (R2)+           ;Point to word with block # if SAV file
014430  012237  000000G                 MOV     (R2)+,INDDBL    ;GET BLOCK # IN SAV FILE OF DATA OVERLAY
014434  011202                          MOV     (R2),R2         ;GET # OF WORDS IN OVERLAY SEGMENT
014436  062702  000377                  ADD     #255.,R2        ;ROUND UP TO NEXT BLOCK
014442  000302                          SWAB    R2              ;CONVERT # WORDS TO # BLOCKS
014444  042702  177400                  BIC     #^C<377>,R2
014450  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.
                                ;
014454  012700  000000G                 MOV     #INDSAV,R0      ;GET ADDRESS OF SAVESTATUS BLOCK
014460  013702  000264'                 MOV     INDNAM,R2       ;GET RAD50 DEVICE NAME
014464  004737  024616'                 CALL    SETCHN          ;SAVE FILE STATUS
                                ;
                                ;  Determine how much space is needed for SY:TSXIND.TSX swap file
                                ;
014470  013703  000000G                 MOV     INDDBS,R3       ;GET # BLOCKS NEEDED PER JOB
014474  070327  000000C                 MUL     #<LSTSL/2>,R3   ;TIMES TOTAL NUMBER OF JOBS
```

```
                                        ;
                                        ;  Load Rt-11 device handler for ind swap file.
                                        ;
014500  013700  0000000                         MOV     INDFIL,R0       ;Get name of the device
014504  004737  025176'                         CALL    RTFTCH          ;Try to fetch the RT-11 device handler
014510  103522                                  BCS     11$             ;Br if error on handler fetch
                                        ;
                                        ;  See if TSXIND file already exists
                                        ;
014512                                          .LOOKUP #AREA,#1,#INDFIL ;DOES SY:TSXIND.TSX FILE EXIST NOW?
014532  103415                                  BCS     1$              ;BR IF NOT
014534  020003                                  CMP     R0,R3           ;IS IT OF THE CORRECT SIZE?
014536  001462                                  BEQ     2$              ;BR IF YES
014540                                          .PURGE  #1              ;FILE IS OF WRONG SIZE
014546                                          .DELETE #AREA,#1,#INDFIL;DELETE OLD FILE
                                        ;
                                        ;  File does not now exist
                                        ;  Create new file
                                        ;
014566                                  1$:     .ENTER  #AREA,#1,#INDFIL,R3 ;CREATE NEW TSXIND FILE
014612  103451                                  BCS     10$             ;BR IF ERROR ON CREATE
014614  010302                                  MOV     R3,R2           ;# BLOCKS IN FILE
014616  005302                                  DEC     R2              ;GET # OF LAST BLOCK IN FILE
014620                                          .WRITW  #AREA,#1,WRKBUF,#256.,R2 ;WRITE TO LAST BLOCK OF FILE
014656                                          .CLOSE  #1              ;NOW CLOSE THE FILE
014664                                          .LOOKUP #AREA,#1,#INDFIL ;REOPEN TSXIND FILE WITH LOOKUP
                                        ;
                                        ;  Do .SAVESTATUS for SY:TSXIND.TSX file
                                        ;
014704  012700  0000000                 2$:     MOV     #INDTSV,R0      ;POINT TO SAVESTATUS BLOCK
014710  013702  0000000                         MOV     INDFIL,R2       ;GET RAD50 DEVICE NAME
014714  004737  024616'                         CALL    SETCHN          ;SAVE FILE INFO
014720                                          .RELEAS #INDFIL         ;Release device handler
                                        ;
                                        ;  Finished
                                        ;
014730  012603                          9$:     MOV     (SP)+,R3
014732  012602                                  MOV     (SP)+,R2
014734  000207                                  RETURN
                                        ;
                                        ;  Error occurred while opening SY:TSXIND.TSX file
                                        ;
014736                                  10$:    .PRINT  #TSXHD          ;Print error message
014744                                          .PRINT  #INDOPN
014752  004737  010142'                         CALL    SPNEED          ;Print info about number of blocks needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
014756  010001                          11$:    MOV     R0,R1           ;Save device name
014760                                          .PRINT  #TSXHD          ;Print error message
014766                                          .PRINT  #INDOPN
014774  004737  010170'                         CALL    BADDEV          ;Print invalid device specification
```

```
                                        .SBTTL  UCLINI -- Initialize TSXUCL data file
                                ;-------------------------------------------------------------------------
                                ;  UCLINI is called to initialize the TSXUCL data file which is used
                                ;  to store user-defined commands.
                                ;
                                ;  Outputs:
                                ;   TSXUCL data file is initialized.
                                ;   UCLBLK = Number of blocks in data file for each job.
                                ;
015000  010246                  UCLINI: MOV     R2,-(SP)
015002  010346                          MOV     R3,-(SP)
                                ;
                                ;  Determine if TSXUCL data file is needed
                                ;
015004  105737  0000000G                TSTB    VU$CL           ;Is TSXUCL being used at all?
015010  001536                          BEQ     9$              ;Br if not
015012  013702  0000000                 MOV     VUCLMC,R2       ;Get maximum number of commands
015016  001533                          BEQ     9$              ;Br if none allowed
                                ;
                                ;  Determine number of blocks needed in data file for each job
                                ;
015020  012700  0000000G                MOV     #UK$$SZ,R0      ;Size of each keyword descriptor
015024  062700  0000000G                ADD     #US$$SZ,R0      ;Size of each command string descriptor
015030  070200                          MUL     R0,R2           ;Compute total # bytes for keywords+commands
015032  062703  0007770G                ADD     #UC$$SZ+511.,R3 ;Add space for control information & round up
015036  005502                          ADC     R2              ;Propogate carry
015040  071227  001000                  DIV     #512.,R2        ;Convert to # of blocks needed
015044  010237  0000000G                MOV     R2,UCLBLK       ;Save number of blocks needed per job
                                ;
                                ;  Multiply by number of jobs to get total file size
                                ;
015050  070227  000000C                 MUL     #<LSTSL/2>,R2   ;Times total number of jobs
                                ;
                                ;  Load Rt-11 device handler for ind swap file.
                                ;
015054  013700  0000000G                MOV     UCLDAT,R0       ;Get name of the device
015060  004737  025176'                 CALL    RTFTCH          ;Try to fetch the RT-11 device handler
015064  103523                          BCS     11$             ;Br if error on handler fetch
                                ;
                                ;  The total required file size is now in R3.
                                ;  See if the file already exists.
                                ;
015066                                  .LOOKUP #AREA,#1,#UCLDAT ;See if the file exists now
015106  103415                          BCS     1$              ;Br if file does not exist
015110  020003                          CMP     R0,R3           ;Is existing file of correct size?
015112  001446                          BEQ     2$              ;Br if yes -- use the old file
015114                                  .PURGE  #1              ;Purge the channel
015122                                  .DELETE #AREA,#1,#UCLDAT;Delete the old file
                                ;
                                ;  Create a new data file
                                ;
015142                          1$:     .ENTER  #AREA,#1,#UCLDAT,R3 ;Create new data file
015166  103452                          BCS     10$             ;Br if error creating the file
015170  005303                          DEC     R3              ;Get # of last block in the file
015172                                  .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.)
                                        ;
015230                          2$:     .CSTAT  #AREA,#1,#NFSBLK        ;GET CHANNEL STATUS INFORMATION
015250  013702  000032'                 MOV     <NFSBLK+12>,R2  ;FETCH DEVICE NAME IN RAD50
015254  063702  000030'                 ADD     <NFSBLK+10>,R2  ;ADD IN DEVICE UNIT NUMBER
015260  062702  000036                  ADD     #^R  0,R2       ;CONVERT UNIT NUMBER TO RAD50
015264  010237  0000000                 MOV     R2,UCLDAT       ;SET PHYSICAL NAME BACK INTO TSGEN CELL
015270                                  .CLOSE  #1              ;Close the file
015276                                  .RELEAS #UCLDAT         ;Release device handler
                                        ;
                                        ;  Finished
                                        ;
015306  012603                  9$:     MOV     (SP)+,R3
015310  012602                          MOV     (SP)+,R2
015312  000207                          RETURN
                                        ;
                                        ;  Error creating the data file
                                        ;
015314                          10$:    .PRINT  #TSXHD          ;Print error message
015322                                  .PRINT  #UCLOPN
015330  004737  010142'                 CALL    SPNEED          ;Print info about number of blocks needed
                                        ;
                                        ;  Error: Invalid device specification.
                                        ;
015334  010001                  11$:    MOV     R0,R1           ;Save device name
015336                                  .PRINT  #TSXHD          ;Print error message
015344                                  .PRINT  #UCLOPN
015352  004737  010170'                 CALL    BADDEV          ;Print invalid device specification
```

```
                                        .SBTTL  MEMINI -- Initialize memory management
                                ;-------------------------------------------------------------------
                                ;  Initialize memory management registers for a 1-to-1 mapping.
                                ;  But leave memory management turned off.
                                ;
015356  010146                  MEMINI: MOV     R1,-(SP)
015360  010246                          MOV     R2,-(SP)
015362  010346                          MOV     R3,-(SP)
015364  010446                          MOV     R4,-(SP)
015366  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize all pages for a 1-to-1 mapping.
                                ;
015370  012700  0000000         12$:    MOV     #KPAR0,R0       ;Kernel mode PAR 0
015374  012701  0000000                 MOV     #UPAR0,R1       ;User mode PAR 0
015400  012702  0000000                 MOV     #KPDR0,R2       ;Kernel mode PDR 0
015404  012703  0000000                 MOV     #UPDR0,R3       ;User mode PDR 0
015410  012704  000010                  MOV     #8.,R4          ;Initialize 8 pages
015414  005005                          CLR     R5              ;Set initial PAR value
015416  010520                  2$:     MOV     R5,(R0)+        ;Set kernel PAR
015420  010521                          MOV     R5,(R1)+        ;Set user PAR value
015422  012722  077406                  MOV     #077406,(R2)+   ;Set kernel PDR
015426  012723  077406                  MOV     #077406,(R3)+   ;Set user PDR value
015432  062705  000200                  ADD     #200,R5         ;Advance block number
015436  077411                          SOB     R4,2$           ;Init all pages
                                ;
                                ;  Map kernel mode I/O page (160000) to 17760000.
                                ;
015440  012737  0000000 0000000         MOV     #IOPAGE,@#KPAR7 ;Map I/O page
                                ;
                                ;  Finished
                                ;
015446  012605                          MOV     (SP)+,R5
015450  012604                          MOV     (SP)+,R4
015452  012603                          MOV     (SP)+,R3
015454  012602                          MOV     (SP)+,R2
015456  012601                          MOV     (SP)+,R1
015460  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

015462  010146                  MEMTST: MOV     R1,-(SP)
015464  010246                          MOV     R2,-(SP)
015466  010346                          MOV     R3,-(SP)
015470  010446                          MOV     R4,-(SP)
015472  010546                          MOV     R5,-(SP)
015474  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.
                                ;
015500  012737  015716' 000004          MOV     #TRCSET,@#4     ;Catch trap
015506  000240                          NOP                     ;Clean out 11/73 pipeline
015510  000240                          NOP                     ;Before attempting trap
015512  005737  000000G                 TST     @#SR3MMR        ;Try to access status register 3
015516  103402                          BCS     22$             ;Br if MMU 3 status register is non-existent
015520  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.
                                ;
015524                          22$:
                                . IF    NE,PROASM
015524  105737  000000G                 TSTB    PROFLG          ;Are we running on a Professional?
015530  001410                          BEQ     26$             ;Br if not
015532  005005                          CLR     R5              ;Load byte without sign extension
015534  153705  173050                  BISB    @#173050,R5     ;Get 32Kb top of system RAM boundary
015540  072527  000011                  ASH     #9.,R5          ;Convert to # 64 byte blocks
015544  162705  000010                  SUB     #10,R5          ;Don't use the last 512 bytes of memory
015550  000400                          BR      7$
015552                          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.
                                ;
                                        MOV     #RTNKM,@#20     ;Use IOT instruction to get out of user mode
                                        CLR     @#22
                                        BIS     #MMENBL,@#SROMMR;Enable memory management
                                        TSTB    SR3FLG          ;Does maching have mem management reg # 3?
                                        BEQ     4$              ;Br if non-existent
                                        BIS     #EMMAP,@#SR3MMR ;Enable 22-bit extended memory
                                ;
                                ;  Map user page 7 to each successive 256-word block and attempt to access.
                                ;
                                4$:     MOV     #1024.,R5       ;Start checking at 64Kb
                                5$:     MOV     R5,@#UPAR7      ;Map user page 7 to page to be tested
                                        BIS     #UMODE,@#PSW    ;Go into user mode
                                        TST     @#160000        ;Can we access the page?
                                ;
                                ;  Use IOT to get back into kernel mode.
                                ;
                                        IOT                     ;Return to kernel mode
                                        BCS     6$              ;Br if memory is non-existent
                                        ADD     #10,R5          ;Go try next page
                                        CMP     R5,#177600      ;Don't enter I/O page
                                        BLO     5$
                                ;
                                ;  Check for potential memory wrap (on 18-bit 256K byte computers).
                                ;
                                6$:     CMP     R5,#10000       ;Is physical memory above 256K bytes
                                        BLOS    7$              ;Br if below 256K bytes
                                        CLR     @#TSTWRD        ;Clear physical location
                                        MOV     #10000,@#UPAR7  ;Map to 256K byte boundary
                                        BIS     #UMODE,@#PSW    ;Go into user mode
                                        MOV     #-1,@#160000+TSTWRD     ;Store -1 at 256K physical location
                                        IOT                     ;Return to kernel mode
                                        TST     @#TSTWRD        ;Test physical location
                                        BEQ     7$              ;Br if physical location is clear
                                        MOV     #10000,R5       ;Constrain memory to 256K byte total
                                .ENDC   ;NE,<PROASM-1>
                                ;
                                ;  Reached end of available memory.
                                ;
015552  010537  000000G         7$:     MOV     R5,PHYMEM       ;set physical memory size
015556  020537  000000G                 CMP     R5,MAPSIZ       ;Constrain kernel to user specified cutoff
015562  101402                         BLOS    8$              ;Br if below user specified
015564  013705  000000G                 MOV     MAPSIZ,R5       ;Only use this much memory
015570  010537  000000G         8$:     MOV     R5,$MEMSZ       ;Set # 64-byte blocks of total memory
015574  010537  000134'                 MOV     R5,FMEMHI       ;Save base 64-byte block # of top of free mem
                                ;
                                ;  Turn off memory management
                                ;
015600  105737  000000G                 TSTB    SR3FLG          ;Do we have memory management reg # 3?
015604  001403                         BEQ     9$              ;Br if non-existent
015606  042737  000000G 000000G        BIC     #EMMAP,@#SR3MMR ;Disable extended memory management
015614  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
                                ;
015622  023727  000134' 010000         CMP     FMEMHI,#4096.   ;Does machine have at least 256Kb?
```

```
                                        BLO     25$             ;Br if not
                                        INCB    MEM256          ;Remember machine has at least 256kb
                                        CMPB    VBUSTP,#QBUS    ;Is this a Q-bus machine?
                                        BNE     25$             ;Br if not
                                        BIS     #EXTLSI,ICONFG  ;Set extended-LSI flag in ICONFG
                                ;
                                ;  See if this machine needs UNIBUS mapping
                                ;
                                25$:    CMPB    VBUSTP,#UNIBUS  ;Is this a UNIBUS machine?
                                        BNE     29$             ;Br if not
                                        TSTB    MEM256          ;Does machine have at least 256kb of memory?
                                        BEQ     29$             ;Br if not
                                        INCB    UBUSMP          ;Say UNIBUS mapping is needed
                                ;
                                ;  Finished
                                ;
                                29$:    MOV     (SP)+,@#4       ;Reset trap vector
                                        MOV     (SP)+,R5
                                        MOV     (SP)+,R4
                                        MOV     (SP)+,R3
                                        MOV     (SP)+,R2
                                        MOV     (SP)+,R1
                                        RETURN
                                ;
                                ;  Trap - return with C-bit set.
                                ;
                                TRCSET: BIS     #1,2(SP)        ;Set c-bit for return
                                        RTI                     ;Return from trap
                                ;
                                ;  IOT - return at kernel mode with c-bit preserved.
                                ;
                                RTNKM:  BIC     #UMODE,2(SP)    ;Clear user mode - return to kernel
                                        RTI                     ;Return from trap
                                ;
                                ;  Error: System does not have memory management hardware.
                                ;
                                NOXM:   .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
                                        .PRINT  #NXMMSG
                                        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.
                                ;
015756                          CXTALC:
                                ;
                                ;  Get size of base portion of job context area
                                ;
015756  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.
                                ;
015762  062700  000077                 ADD     #63.,R0         ;Bound up to 64 byte boundary
015766  042700  000077                 BIC     #77,R0
015772  010037  000000G                MOV     R0,CXTRMN       ;Offset to start of simulated RMON
015776  062737  000000G 000000G        ADD     #CXTBAS,CXTRMN  ;Add base virtual address of context area
016004  062700  000010G                ADD     #MVSIZ+1,R0     ;Add space for simulated RMON & channels
                                ;
                                ;  Save number of words needed for context area
                                ;
016010  006200                         ASR     R0              ;Convert to # words
016012  010037  000000G                MOV     R0,CXTWDS       ;This is # words for whole job context area
                                ;
                                ;  Compute PDR value to use to map to job context area
                                ;
016016  062700  000037                 ADD     #31.,R0         ;Bound up to # 32 word units
016022  072027  177773                 ASH     #-5.,R0         ;Get # 32-word units for context area
016026  000300                         SWAB    R0              ;Put # 32-word units in high-order byte
016030  052700  000006                 BIS     #6,R0           ;Set PDR control flags
016034  010037  000000G                MOV     R0,CXTPDR       ;This is the PDR value
                                ;
                                ;  Compute # 512-byte pages needed for job context block
                                ;
016040  013700  000000G                MOV     CXTWDS,R0       ;Get back # words for context area
016044  062700  000377                 ADD     #255.,R0        ;Bound up to # 256-word blocks
016050  072027  177770                 ASH     #-8.,R0         ;Get # 256-word pages for context area
016054  010037  000000G                MOV     R0,CXTPAG       ;# pages for job context area
                                ;
                                ;  Set up PDR value used when mapping to simulated RMON
                                ;
016060  012700  000000G                MOV     #MVSIZ,R0       ;Get size of monitor vector table
016064  062700  000077                 ADD     #63.,R0         ;Round up to # 32 word blocks
016070  072027  177772                 ASH     #-6,R0          ;Cvt to # 32-word blocks
016074  005300                         DEC     R0              ;Get # blocks - 1
016076  000300                         SWAB    R0              ;Put # blocks in left byte
016100  052700  000006                 BIS     #6,R0           ;Allow read and write access
016104  042700  100261                 BIC     #100261,R0      ;Make sure unused PDR bits are zero
016110  010037  000000G                MOV     R0,RMNPDR       ;This is PDR value to map to sim. RMON
                                ;
```

```
      58                                ;  Finished
      59                                ;
      60 016114  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.
                                ;
016116  010246                  MAPALC: MOV     R2,-(SP)
016120  010346                          MOV     R3,-(SP)
016122  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.
                                ;
016124  010503                          MOV     R5,R3           ;Get 64-byte block # of top of free mem
016126  072327  177775                  ASH     #-3,R3          ;Convert to 512-byte page #
016132  042703  160000                  BIC     #160000,R3      ;Kill possible sign extension
016136  013702  000136'                 MOV     FMEMLO,R2       ;Get 64-byte block # of base of free memory
016142  062702  000007                  ADD     #7,R2           ;Round up
016146  072227  177775                  ASH     #-3,R2          ;Convert to 512-byte page #
016152  042702  160000                  BIC     #160000,R2      ;Kill possible sign extension
016156  160203                          SUB     R2,R3           ;Get # bytes needed for allocation table
016160  003440                          BLE     10$             ;Br if memory overflow
016162  010304                          MOV     R3,R4           ;Get # bytes for allocation table
016164  062704  001000                  ADD     #512.,R4        ;Add 1 extra byte and round up to 512-byte
016170  072427  177767                  ASH     #-9.,R4         ;Get # 512-byte units needed for alloc table
                                ;
                                ;  Set up virtual address pointers for the allocation table
                                ;
016174  012700  000000G                 MOV     #VPAR5,R0       ;We will map to alloc table through PAR 5
016200  010037  000000G                 MOV     R0,LOMAP        ;Pointer to 1st entry in alloc table
016204  160200                          SUB     R2,R0           ;Get pseudo virtual address for page # 0
016206  010037  000000G                 MOV     R0,BASMAP       ;This would point to alloc entry for page 0
016212  012700  000000G                 MOV     #VPAR5,R0       ;Get back base address of table
016216  060300                          ADD     R3,R0           ;Add # bytes used by table
016220  160400                          SUB     R4,R0           ;Subtract space used by table itself
016222  010037  000000G                 MOV     R0,HIMAP        ;Virtual address of 1st entry for system page
                                ;
```

```
                                        ;  Allocate space for the allocation table
                                        ;
016226  072427  000003                        ASH     #3,R4           ;Get # 64-byte units for alloc table
016232  160405                                SUB     R4,R5           ;Compute physical 64-byte base for table
016234  020537  000136'                      CMP     R5,FMEMLO       ;Did we run out of memory space?
016240  101410                                BLOS    10$             ;Br if memory overflow
016242  010537  000000G                      MOV     R5,MAPPAR       ;Use this value to map PAR 5 to alloc table
016246  010537  000134'                      MOV     R5,FMEMHI       ;Save new top of free memory area
                                        ;
                                        ;  Finished
                                        ;
016252  012604                                MOV     (SP)+,R4
016254  012603                                MOV     (SP)+,R3
016256  012602                                MOV     (SP)+,R2
016260  000207                                RETURN
                                        ;
                                        ;  Error: Generated system is too large
                                        ;
016262                                  10$:    .PRINT  #TSXHD          ;Print error message heading
016270                                          .PRINT  #PHSOVF         ;Physical memory overflow
016276  000137  004250'                         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)
                                ;
016302  010546                  SETJSZ: MOV     R5,-(SP)
                                ;
                                ;  Determine total number of pages of memory available to user jobs
                                ;
016304  013705  0000000                 MOV     HIMAP,R5        ;POINTER ABOVE LAST FREE PAGE ENTRY
016310  163705  0000000                 SUB     LOMAP,R5        ;GET TOTAL # OF FREE PAGES
016314  010537  0000000                 MOV     R5,FREPGS       ;# FREE PAGES AVAILABLE TO USER JOBS
                                ;
                                ;  Make sure there is enough free space to run TSKMON.
                                ;
016320  020537  0000000                 CMP     R5,KMNPGS       ;COMPARE # FREE PAGES TO # PAGES FOR TSKMON
016324  103436                          BLO     1$              ;BR IF INSUFFICIENT MEMORY TO RUN TSKMON
                                ;
                                ;  Set up max memory limit for jobs
                                ;
016326  013700  0000000                 MOV     CXTPAG,R0       ;# PAGES NEEDED FOR JOB CONTEXT AREA
016332  010037  0000000                 MOV     R0,JCXPGS
016336  160005                          SUB     R0,R5           ;LEAVE ROOM FOR JOB CONTEXT AREA
016340  006205                          ASR     R5              ;Convert # pages to # KB
016342  020537  0000000                 CMP     R5,VHIMEM       ;COMPARE WITH TSGEN SPECIFIED MAX SIZE
016346  101402                          BLOS    10$             ;BR IF CONSTRAINED BY PHYSICAL SIZE
016350  013705  0000000                 MOV     VHIMEM,R5       ;LIMIT BY VALUE SPECIFIED IN TSGEN
016354  010537  0000000          10$:   MOV     R5,MXJMEM       ;MAX # K BYTES AVAILABLE TO A JOB
016360  010500                          MOV     R5,R0
016362  072027  000012                  ASH     #10.,R0         ;CONVERT # KB TO BYTE ADDRESS
016366  001002                          BNE     2$              ;BR IF DIDN'T OVERFLOW 64KB
016370  012700  177774                  MOV     #177774,R0      ;GET 64KB TOP ADDRESS
016374  010037  0000000          2$:    MOV     R0,MXJADR       ;ADDRESS ABOVE TOP OF JOB
                                ;
                                ;  Set default memory size of jobs
                                ;
016400  020537  0000000                 CMP     R5,VDFMEM       ;COMPARE TO DEFAULT SPECIFIED IN TSGEN
016404  101402                          BLOS    11$             ;BR IF CONSTRAINED BY PHYSICAL LIMIT
016406  013705  0000000                 MOV     VDFMEM,R5       ;CONSTRAIN BY TSGEN PARAMETER
016412  010537  0000000          11$:   MOV     R5,DFJMEM       ;SET DEFAULT JOB MEMORY SIZE
                                ;
                                ;  Finished
                                ;
016416  012605                          MOV     (SP)+,R5
016420  000207                          RETURN
                                ;
                                ;  Error -- Insufficient memory space available to run TSKMON.
```

```
     58                                         ;
     59 016422  004737  025332'         1$:     CALL    SIZERR          ;Generated system is too big -- abort
```

```
                        .IF     NE,<PROASM-1>   ;No parity control if PRO only
                        .SBTTL  PARSET -- Setup memory parity control
                ;-----------------------------------------------------------------------
                ;  PARSET is called to set up memory parity control.
                ;  Currently this consists of disabling memory parity.
                ;
                PARSET: TST     #MPARFL         ;Does he want to disable memory parity?
                        BNE     20$             ;Br if not
                        MOV     R2,-(SP)
                        MOV     @#4,-(SP)       ;Save contents of trap vector
                ;
                ;  Catch traps that occur when we access unimplemented parity registers
                ;
                        MOV     #2$,@#4         ;Send traps to 2$
                        NOP                     ;Clean out instruction pipeline
                        NOP
                ;
                ;  Disable parity for each block of memory
                ;
                        MOV     #MPAR0,R2       ;Point to 1st memory control register
                1$:     BIC     #PARENL,(R2)    ;Disable memory parity
                        BR      3$              ;We did not trap
                2$:     ADD     #4,SP           ;Clean trap PS and PC off of stack
                3$:     ADD     #2,R2           ;Point to next parity control register
                        CMP     R2,#MPAR16      ;Have we cleared all registers?
                        BLOS    1$              ;Loop if not
                ;
                ;  Finished
                ;
                        MOV     (SP)+,@#4       ;Restore trap vector
                        MOV     (SP)+,R2
                20$:    RETURN
                        .IFF    ;NE,<PROASM-1>
016426  000207  PARSET: RETURN
                        .ENDC   ;NE,<PROASM-1>
```

```
                                        .SBTTL  GETHNL -- Load device handlers into memory
                                ;-----------------------------------------------------------------------
                                ;  GETHNL performs two functions:
                                ;   1. Set up information in the device tables about all devices.
                                ;   2. Load those handlers that reside in low memory.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of start of free memory.
                                ;
                                ;  Outputs:
                                ;   R5 = Address of new start of free memory.
                                ;   NMXHAN = Number of handlers to load into extended memory.
                                ;
016430  010146                  GETHNL: MOV     R1,-(SP)
016432  010246                          MOV     R2,-(SP)
016434  010446                          MOV     R4,-(SP)
                                ;
                                ;  Begin loop to check all handlers specified in TSGEN with DEVDEF.
                                ;
016436  005001                          CLR     R1              ;Init device table index
016440  020127  000000C         1$:     CMP     R1,#<AHEND-AUTHAN> ;Done all devices?
016444  103015                          BHIS    2$              ;Br if yes
016446  016102  000000G                 MOV     AUTHAN(R1),R2   ;Get the name of the device
016452  001407                          BEQ     3$              ;Ignore null devices
016454  020227  000000G                 CMP     R2,#DMYDEV      ;Is this a dummy device entry?
016460  001404                          BEQ     3$              ;Skip it if yes
016462  016104  000000G                 MOV     DTYPE(R1),R4    ;Get flags specified in TSGEN
                                ;
                                ;  Load this handler
                                ;
016466  004737  016540'                 CALL    LDHAND          ;Try to load handler into memory
                                ;
                                ;  Check next device
                                ;
016472  062701  000002          3$:     ADD     #2,R1           ;Advance device index
016476  000760                          BR      1$              ;See if more devices to load
                                ;
                                ;  Now see if there are spooled devices to contend with
                                ;
016500  012704  000000G         2$:     MOV     #SPLND,R4       ;Are there any spooled devices?
016504  001411                          BEQ     9$              ;Br if not
016506  012701  000000G                 MOV     #SPLDEV,R1      ;Point to spooled device name table
016512  012102                  5$:     MOV     (R1)+,R2        ;Get the name of the next spooled device
016514  010446                          MOV     R4,-(SP)        ;Save device count
016516  005004                          CLR     R4              ;Say no TSGEN flags for device
016520  004737  016540'                 CALL    LDHAND          ;Load the handler
016524  012604                          MOV     (SP)+,R4        ;Recover the device count
016526  077407                          SOB     R4,5$           ;Loop if more handlers to load
                                ;
                                ;  Finished
                                ;
016530  012604                  9$:     MOV     (SP)+,R4
016532  012602                          MOV     (SP)+,R2
016534  012601                          MOV     (SP)+,R1
016536  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$xxx status flags for device from TSGEN.
                                ;   R5 = Address where handler is to be loaded.
                                ;
                                ;  Outputs:
                                ;   R5 = New free memory address.
                                ;   NUMDEV = Incremented by 2.
                                ;   PNAME(i) = Rad-50 name of device.
                                ;   ENTRY(i) = Handler entry point.
                                ;   DVSTAT(i) = Device status flags.
                                ;   DVFLAG(i) = TSX-Plus device status flags.
                                ;   HANPAR(i) = PAR offset if this is a mapped handler.
                                ;
016540  010446                  LDHAND: MOV     R4,-(SP)
                                ;
                                ;  Determine if we should ignore this device
                                ;
016542  004737  016710'                 CALL    INSCK1          ;Should we ignore this device?
016546  103456                         BCS     9$              ;Br if yes
                                ;
                                ;  The initial tests indicate that this handler should be loaded.
                                ;  Now open the handler file and perform some additional checks.
                                ;
016550  004737  017066'                 CALL    INSCK2          ;Perform some additional checks on handler
016554  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.
                                ;
016556  004737  017500'                 CALL    STDVTB          ;Set up info in tables for this device
                                ;
                                ;  Determine if this handler is to be loaded into low memory or
                                ;  extended memory.
                                ;
016562  016400  000000G                 MOV     DVFLAG(R4),R0   ;Get TSX-Plus status flags for device
016566  032700  000000G                 BIT     #DX$NHM,R0      ;Are we never to map this handler?
016572  001035                         BNE     1$              ;Br if handler cannot be mapped
016574  032700  000000G                 BIT     #DX$MPH,R0      ;Is mapping wanted for this handler?
016600  001432                         BEQ     1$              ;Br if not
016602  032700  000000G                 BIT     #DX$IBH,R0      ;Does this handler have an internal I/O buff?
016606  001412                         BEQ     2$              ;Br if not
016610  105737  000000G                 TSTB    UBUSMP          ;Does this machine have a mapped UNIBUS?
016614  001024                         BNE     1$              ;Br if yes -- Don't map this handler
016616  032700  000000G                 BIT     #DX$MAP,R0      ;Does this handler require I/O mapping?
016622  001404                         BEQ     2$              ;Br if not
016624  032737  000001  000306'        BIT     #EXTLSI,ICONFG  ;Is this a Q-bus system with more than 256Kb?
016632  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 016634  005237  000124'          2$:      INC     NMXHAN          ;Count # of mapped handlers
      60 016640  012764  000001  000000G          MOV     #1,HANPAR(R4)   ;Set flag saying handler should be mapped
      61                                        ;
      62                                        ;  Make sure size of mapped handler does not exceed 8KB
      63                                        ;
      64 016646  026427  000000G 020000          CMP     HANSIZ(R4),#8192. ;Is mapped handler too big?
      65 016654  101410                          BLOS    8$              ;Br if not too big
      66 016656  012700  002452'                 MOV     #HN2BIG,R0      ;Get error message address
      67 016662  000137  020566'                 JMP     HLERR           ;Abort initialization
      68                                        ;
      69                                        ;  This handler must be loaded into low memory
      70                                        ;
      71 016666  005064  000000G          1$:      CLR     HANPAR(R4)      ;Say this handler is not mapped
      72 016672  004737  017610'                 CALL    LDHNLO          ;Load handler into low memory
      73                                        ;
      74                                        ;  Close the handler file
      75                                        ;
      76 016676                           8$:      .CLOSE  #1              ;Close the handler file
      77                                        ;
      78                                        ;  Finished
      79                                        ;
      80 016704  012604                   9$:      MOV     (SP)+,R4
      81 016706  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.
                                ;
016710  010146                  INSCK1: MOV     R1,-(SP)
                                ;
                                ;  Strip off any specified unit number
                                ;
016712  010201                          MOV     R2,R1           ;Get full device name
016714  005000                          CLR     R0              ;Set for divide
016716  071027  000050                  DIV     #50,R0          ;Split off last digit
016722  070027  000050                  MUL     #50,R0          ;Now correct for divide
016726  010102                          MOV     R1,R2           ;Get device name less 3rd digit
016730  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.
                                ;
016734  020237  000176'                  CMP     R2,R50LD        ;Is device name LD?
016740  001004                          BNE     1$              ;Br if not
016742  105737  000000G                 TSTB    VLDSYS          ;Is standard system LD support included?
016746  001044                          BNE     5$              ;Br if yes -- Don't load LD
016750  000417                          BR      3$              ;Load LD
016752  012701  000224'          1$:     MOV     #SKPDEV,R1      ;Point to table of devices to skip
016756  020221                  2$:     CMP     R2,(R1)+        ;Is this a device to be skipped?
016760  001437                          BEQ     5$              ;Br if yes
016762  005711                          TST     (R1)            ;Reached end of skip list?
016764  001374                          BNE     2$              ;Loop if not
                                ;
                                ;  See if we have already loaded the handler for this device
                                ;
016766  013701  000000G                 MOV     NUMDEV,R1       ;Get index for last device
016772  001406                          BEQ     3$              ;Br if no devices installed yet
016774  020261  000000G          4$:     CMP     R2,PNAME(R1)    ;See if this device is already installed
017000  001427                          BEQ     5$              ;Br if already installed
017002  162701  000002                  SUB     #2,R1           ;More installed devices to check?
017006  003372                          BGT     4$              ;Loop if yes
                                ;
                                ;  This handler is to be loaded.
                                ;  Get default TSX-Plus control flags for this device.
                                ;
017010  012701  000342'          3$:     MOV     #DVFLBS,R1      ;Point to start of table
017014  020261  000000           6$:     CMP     R2,DV$NAM(R1)   ;Search for device in the table
017020  001003                          BNE     7$              ;Br if this is not it
017022  056104  000002                  BIS     DV$FLG(R1),R4   ;Combine default flags
017026  000405                          BR      8$
```

```
     58 017030  062701  000004          7$:     ADD     #DV$$SZ,R1      ;Point to next entry
     59 017034  020127  000522'                 CMP     R1,#DVFLND      ;Checked all entries?
     60 017040  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 017042  032704  0000000         8$:     BIT     #DX$DMA,R4      ;Is this a DMA device?
     66 017046  001402                          BEQ     10$             ;Br if not
     67 017050  052704  0000000                 BIS     #DX$EBA,R4      ;Set even-buffer-boundary flag
     68                                 ;
     69                                 ;  Load this handler
     70                                 ;
     71 017054  000241                  10$:    CLC                     ;Set flag saying to load the handler
     72 017056  000401                          BR      9$
     73                                 ;
     74                                 ;  Do not load this handler
     75                                 ;
     76 017060  000261                  5$:     SEC                     ;Set flag saying not to load the handler
     77                                 ;
     78                                 ;  Finished
     79                                 ;
     80 017062  012601                  9$:     MOV     (SP)+,R1
     81 017064  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.
                ;
017066  010146  INSCK2: MOV     R1,-(SP)
017070  010246          MOV     R2,-(SP)
017072  010346          MOV     R3,-(SP)
017074  010446          MOV     R4,-(SP)
017076  010546          MOV     R5,-(SP)
017100  013746  000004  MOV     @#4,-(SP)       ;Save the bus timeout vector
017104  013746  000010  MOV     @#10,-(SP)      ;Save illegal instruction vector
                ;
                ;  Try to lookup handler file on system disk
                ;
017110  010237  000246' MOV     R2,HANNAM+2     ;Set the device name for the lookup
017114                  .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
017134  103011          BCC     1$              ;Br if we found the handler file
                ;
                ;  Error -- Cannot find handler file
                ;
017136  105737  000000G TSTB    VINABT          ;Abort or continue on errors?
017142  001002          BNE     2$              ;Br if abort on errors
017144  000137  017446' JMP     10$             ;Say not to load this handler
017150  012700  001636' 2$:     MOV     #CFHMSG,R0      ;Can't find handler
017154  000137  020566'         JMP     HLERR           ;Abort initialization
                ;
                ;  We were able to open the handler file.
                ;  Read in block 0 of handler.
                ;
017160          1$:     .READW  #AREA,#1,WRKBUF,#256.,#0 ;Read block 0 into WRKBUF
017216  103004          BCC     3$              ;Br if read ok
017220  012700  001700' MOV     #ERHMSG,R0      ;Error during read
017224  000137  020566'         JMP     HLERR
                ;
                ;  Determine if the handler is supported under the current RT-11 version
                ;
017230  012700  000330' 3$:     MOV     #HVTBL,R0       ;Point to table with handler version info
017234  013701  000152' MOV     WRKBUF,R1       ;Point to buffer with block 0 of handler
017240  116101  000000G MOVB    H.DSTS(R1),R1   ;Get device ID code from handler
017244  120160  000000  51$:    CMPB    R1,HV$ID(R0)    ;Compare handler ID code with table entry
017250  001020          BNE     53$             ;Br if this entry not for this handler
017252  123760  000000G 000001 CMPB    SYSVER,HV$VER(R0);Compare curr RT-11 vers with min acceptable
017260  103405          BLO     52$             ;Br if version is not adequate
017262  101020          BHI     54$             ;Br if version is ok
017264  123760  000000G 000002 CMPB    SYSUPD,HV$UPD(R0);Compare update level
017272  103014          BHIS    54$             ;Br if update level is ok
017274  105737  000000G 52$:    TSTB    VINABT          ;Handler not loadable - abort or continue?
017300  001462          BEQ     10$             ;Br if continue but don't allow loading
```

```
     58 017302  012700  001744'                    MOV     #ERHNDV,R0      ;Wrong version of RT for handler
     59 017306  000137  020566'                    JMP     HLERR           ;Report error and abort
     60 017312  062700  000003          53$:      ADD     #HV$$SZ,R0      ;Point to next handler version table entry
     61 017316  020027  000341'                    CMP     R0,#HVEND       ;Are there more entries?
     62 017322  103750                             BLO     51$             ;Loop if more to check
     63                                 ;
     64                                 ;  Check handler sysgen options
     65                                 ;
     66 017324  013700  000152'         54$:      MOV     WRKBUF,R0       ;Point to buffer with handler block 0
     67 017330  032760  000000G 000000G           BIT     #SG$MMU,H.GEN(R0);Was handler genned with XM support?
     68 017336  001004                             BNE     4$              ;Br if yes
     69 017340  012700  002045'                    MOV     #HSGER,R0       ;Error if not XM version of handler
     70 017344  000137  020566'                    JMP     HLERR
     71 017350  016037  000000G 000304' 4$:       MOV     H.GEN(R0),HGENFL;Save handler sysgen flags for later
     72                                 ;
     73                                 ;  Check the CSR address specified in the handler to see if the
     74                                 ;  hardware device for this handler exists.
     75                                 ;
     76 017356  012737  015716' 000004             MOV     #TRCSET,@#4     ;Catch bus timeout traps
     77 017364  012737  015716' 000010             MOV     #TRCSET,@#10    ;Catch illegal instruction traps
     78 017372  013700  000152'                    MOV     WRKBUF,R0       ;Point to start of block 0 of handler
     79 017376  016001  000000G                    MOV     H.CSR(R0),R1    ;Get address of CSR for device
     80 017402  001402                             BEQ     5$              ;Br if no CSR specified
     81 017404  005711                             TST     (R1)            ;Is CSR accessible?
     82 017406  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 017410  062700  000000G         5$:       ADD     #H.INS,R0       ;Offset 200 in block 0
     89 017414  005710                             TST     @R0             ;Does any installation code exist?
     90 017416  001415                             BEQ     11$             ;Br if no driver installation code
     91 017420  013746  000000G                    MOV     @#RMON,-(SP)    ;Save RT-11 RMON pointer
     92 017424  012737  000000G 000000G           MOV     #MONVEC,@#RMON  ;Set TSX-Plus RMON pointer
     93 017432  013703  000000G                    MOV     RPRVEC,R3       ;Get pointer to Pro vec addr routine
     94 017436  004710                             CALL    @R0             ;Call the installation code
     95 017440  012637  000000G                    MOV     (SP)+,@#RMON    ;Restore RT-11 RMON pointer
     96 017444  000403                             BR      13$             ;C-flag now indicates handler load status
     97                                 ;
     98                                 ;  Finished with installation verification.
     99                                 ;
    100 017446  000261                 10$:       SEC                     ;Set c-bit for indicating handler
    101 017450  000401                             BR      13$             ;should not be installed
    102 017452  000241                 11$:       CLC                     ;Clear the c-bit for driver installation
    103 017454  012637  000010         13$:       MOV     (SP)+,@#10      ;Restore illegal instruction vector
    104 017460  012637  000004                    MOV     (SP)+,@#4       ;Restore the bus timeout vector
    105 017464  012605                             MOV     (SP)+,R5
    106 017466  012604                             MOV     (SP)+,R4
    107 017470  012603                             MOV     (SP)+,R3
    108 017472  012602                             MOV     (SP)+,R2
    109 017474  012601                             MOV     (SP)+,R1
    110 017476  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).
                                ;
017500                          STDVTB:
                                ;
                                ;  Increment device counter
                                ;
017500  062737  000002  000000G         ADD     #2,NUMDEV       ;Say another device added to tables
017506  013700  000000G                 MOV     NUMDEV,R0       ;Get device index number
                                ;
                                ;  Set up PNAME and DVFLAG.
                                ;
017512  010260  000000G                 MOV     R2,PNAME(R0)    ;Set permanent device name
017516  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.
                                ;
017522  012760  000001  000000G         MOV     #1,HANDSK(R0)   ;Set block # of block 1 of handler file
                                ;
                                ;  Extract parameters from handler block 0
                                ;
017530  010004                          MOV     R0,R4           ;Carry device index in R4
017532  013700  000152'                 MOV     WRKBUF,R0       ;Point to block 0 of handler
017536  016064  000000G 000000G         MOV     H.SIZ(R0),HANSIZ(R4) ;Set handler size
017544  016064  000000G 000000G         MOV     H.DVSZ(R0),DEVSIZ(R4) ;Number of blocks on device
017552  016064  000000G 000000G         MOV     H.DSTS(R0),DVSTAT(R4) ;Set device status flags
                                ;
                                ;  Disable MOUNTs and data caching for certain devices
                                ;
017560  032764  000000G 000000G         BIT     #DS$DIR,DVSTAT(R4) ;Is this a directory structured device?
017566  001404                          BEQ     1$              ;Br if not -- No mounts allowed
017570  032764  000000G 000000G         BIT     #DS$NRD,DVSTAT(R4) ;Non RT-11 directory structure (mag tape)?
017576  001403                          BEQ     9$              ;Br if not
017600  052764  000000C 000000G 1$:     BIS     #<DX$NMT!DX$NCA>,DVFLAG(R4) ;Disable mounts and data caching
                                ;
                                ;  Finished
                                ;
```

```
     58 017606  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.
                                ;
017610  010346                 LDHNLO: MOV     R3,-(SP)
                                ;
                                ;  Determine if we have enough free memory space to read the handler
                                ;
017612  005064  000000G                 CLR     HANPAR(R4)      ;Say this handler is not mapped
017616  010500                          MOV     R5,R0           ;Get current top of memory address
017620  016403  000000G                 MOV     HANSIZ(R4),R3   ;Get size of handler
017624  060300                          ADD     R3,R0           ;Get address above top of handler
017626  004737  025312'                 CALL    CHKMEM          ;See if handler will fit in memory
                                ;
                                ;  Handler will fit.  Read it into memory.
                                ;
017632  006203                          ASR     R3              ;Get number of words to read
017634                                  .READW  #AREA,#1,R5,R3,#1
017670  103004                          BCC     1$              ;Br if read ok
017672  012700  001700'                 MOV     #ERHMSG,R0      ;Error reading handler
017676  000137  020566'                 JMP     HLERR           ;Abort initialization
                                ;
                                ;  Set address of handler entry point and compute address beyond
                                ;  end of the handler.
                                ;
017702  010564  000000G         1$:     MOV     R5,HANENT(R4)   ;Set address of handler entry point
017706  062764  000006  000000G         ADD     #6,HANENT(R4)   ;(Point to fourth word of handler)
017714  006303                          ASL     R3              ;Convert handler size to bytes
017716  060305                          ADD     R3,R5           ;Point beyond end of handler
                                ;
                                ;  Set up table of addresses of support routines at end of handler.
                                ;
017720  010503                          MOV     R5,R3           ;Get address past end of handler
017722  004737  020474'                 CALL    STHNPV          ;Set up pointer vector in handler
                                ;
                                ;  If handler has any load-time exectuion code, run it now
                                ;
017726  004737  020624'                 CALL    DOHNLC          ;Run any load-time code for handler
                                ;
                                ;  Finished
                                ;
017732  012603                          MOV     (SP)+,R3
017734  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.
                                ;
017736  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.
                                ;
017740  012704  000002                  MOV     #2,R4           ;Get index for first device entry
                                ;
                                ;  See if this device has a mapped handler
                                ;
017744  005764  000000G         1$:     TST     HANPAR(R4)      ;Is this handler mapped?
017750  001402                          BEQ     2$              ;Br if not
                                ;
                                ;  We found an entry for a device with a mapped handler.
                                ;  Load the handler.
                                ;
017752  004737  017774'                 CALL    LDHNHI          ;Load a mapped handler
                                ;
                                ;  Look for more mapped handlers
                                ;
017756  062704  000002          2$:     ADD     #2,R4           ;Increment device index
017762  020437  000000G                 CMP     R4,NUMDEV       ;Checked all devices?
017766  101766                          BLOS    1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
017770  012604                          MOV     (SP)+,R4
017772  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.
                                ;
017774  010146                  LDHNHI: MOV     R1,-(SP)
017776  010246                          MOV     R2,-(SP)
020000  010346                          MOV     R3,-(SP)
020002  010446                          MOV     R4,-(SP)
                                ;
                                ;  Open channel 1 to the handler file.
                                ;
020004  016437  000000G 000246'          MOV     PNAME(R4),HANNAM+2 ;Set the device name for the lookup
020012  016437  000000G 000146'          MOV     PNAME(R4),CURNAM;Set name in case we have an error
020020                                   .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
020040  103004                          BCC     8$                 ;Br if we found the handler file
020042  012700  001636'                  MOV     #CFHMSG,R0         ;Can't find handler
020046  000137  020566'                  JMP     HLERR              ;Abort initialization
                                ;
                                ;  Read block 0 of the handler file and extract some information
                                ;
020052  013702  000152'          8$:     MOV     WRKBUF,R2          ;Get address of work buffer
020056                                   .READW  #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
020112  016237  000000G 000304'          MOV     H.GEN(R2),HGENFL;Save handler sysgen flags
                                ;
                                ;  Set virtual address of handler entry point
                                ;
020120  012764  000006G 000000G          MOV     #VPAR5+6,HANENT(R4) ;Set virtual addr of handler entry point
                                ;
                                ;  Get information about the size of the handler and determine the
                                ;  address in extended memory where the handler is to be loaded.
                                ;
020126  016402  000000G                  MOV     HANSIZ(R4),R2   ;Get size of handler (bytes)
020132  005202                          INC     R2              ;Make sure handler size is even
020134  042702  000001                  BIC     #1,R2
020140  010200                          MOV     R2,R0
020142  062700  000077                  ADD     #63.,R0         ;Round up to # 64-byte blocks
020146  072027  177772                  ASH     #-6,R0          ;Get # 64-byte blocks for handler
020152  060037  000000G                  ADD     R0,MHNSIZ       ;Accumulate total space for mapped handlers
020156  160005                          SUB     R0,R5           ;Reserve room for handler
020160  010564  000000G                  MOV     R5,HANPAR(R4)   ;Set mapping value for handler
020164  010537  000126'                  MOV     R5,HMAP         ;Set initial PAR base for handler
020170  012737  000001  000142'          MOV     #1,FILBLK       ;Set # of block to read from file
                                ;
                                ;  Begin loop to read handler into memory
                                ;
020176  010203                  1$:     MOV     R2,R3           ;Get remaining size of handler
020200  020327  001000                  CMP     R3,#512.        ;Compare with max we can read at one time
020204  101402                          BLOS    2$              ;Br if we can read remainder of handler
020206  012703  001000                  MOV     #512.,R3        ;Read one block
020212  160302                  2$:     SUB     R3,R2           ;Reduce amt of handler left to read
```

```
                                        ;
                                        ;  Read next block of handler
                                        ;
020214  006203                          ASR     R3              ;Get # words to read
020216  013701  000152'                 MOV     WRKBUF,R1       ;Get address of buffer for read
020222                                  .READW  #AREA,#1,R1,R3,FILBLK ;Read a block
020256  103004                          BCC     3$              ;Br if read ok
020260  012700  001700'                 MOV     #ERHMSG,R0      ;Get error message
020264  000137  020566'                 JMP     HLERR           ;Abort initialization
                                        ;
                                        ;  Move the code we just read into the XM area for the handler
                                        ;
020270  012700  000000G         3$:     MOV     #VPAR5,R0       ;Get virtual address of mapped region
020274                                  DISABL                  ;** Disable interrupts **
020302  013746  000000G                 MOV     @#KPAR5,-(SP)   ;;;Save current mapping of PAR 5
020306  013737  000126' 000000G         MOV     HMAP,@#KPAR5    ;;;Set up mapping to get to XM area
020314  052737  000000G 000000G         BIS     #MMENBL,@#SROMMR;;;Enable memory management
020322  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
020326  001403                          BEQ     4$              ;;;Br if not
020330  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
020336  012120                  4$:     MOV     (R1)+,(R0)+     ;;;Move from WRKBUF to XM region
020340  077302                          SOB     R3,4$           ;;;Loop till all moved
020342  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
020346  001403                          BEQ     5$              ;;;Br if not
020350  042737  000000G 000000G         BIC     #EMMAP,@#SR3MMR ;;;Disable extended memory addressing
020356  042737  000000G 000000G 5$:     BIC     #MMENBL,@#SROMMR;;;Enable memory management
020364  012637  000000G                 MOV     (SP)+,@#KPAR5   ;;;Replace PAR 5 mapping
020370                                  ENABL                   ; ** Enable interrupts **
                                        ;
                                        ;  See if there is more to read
                                        ;
020376  062737  000010  000126'         ADD     #8.,HMAP        ;Increase XM region base
020404  005237  000142'                 INC     FILBLK          ;Increment file block number
020410  005702                          TST     R2              ;Is there more to read?
020412  001271                          BNE     1$              ;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
                                        ;
020414  012703  000000G                 MOV     #VPAR5,R3       ;Get virtual address of handler base
020420  066403  000000G                 ADD     HANSIZ(R4),R3   ;Get virtual address beyond end of handler
020424  004737  021324'                 CALL    HANMAP          ;;;Map KPAR5 to the handler
020430  004737  020474'                 CALL    STHNPV          ;;;Set up handler pointer vector
020434  004737  021400'                 CALL    HANUMP          ;Restore mapping
                                        ;
                                        ;  If handler has any load-time code, run it now
                                        ;
020440  010537  000134'                 MOV     R5,FMEMHI       ;Set addr of top of free memory area
020444  004737  020624'                 CALL    DOHNLC          ;Run any load-time code for handler
020450  013705  000134'                 MOV     FMEMHI,R5       ;Get new top of free memory address
                                        ;
                                        ;  Close the handler file
                                        ;
020454                                  .CLOSE  #1              ;Close the handler file
                                        ;
                                        ;  Finished
                                        ;
```

```
    115 020462  012604                         MOV     (SP)+,R4
    116 020464  012603                         MOV     (SP)+,R3
    117 020466  012602                         MOV     (SP)+,R2
    118 020470  012601                         MOV     (SP)+,R1
    119 020472  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.
                                ;
020474  010346                 STHNPV: MOV     R3,-(SP)
                                ;
                                ;  Set up addresses in the pointer vector
                                ;
020476  012743  000000G                 MOV     #FORK,-(R3)     ;Address of fork routine
020502  012743  000000G                 MOV     #INTEN,-(R3)    ;Address of inten routine
020506  032737  000000G 000304'         BIT     #SG$IOT,HGENFL  ;Does handler want timeout support?
020514  001402                          BEQ     2$              ;Br if not
020516  012743  000000G                 MOV     #IOTIMR,-(R3)   ;Set address of timeout support routine
020522  032737  000000G 000304' 2$:     BIT     #SG$ELG,HGENFL  ;Does handler want error logging support?
020530  001402                          BEQ     3$              ;Br if not
020532  012743  000000G                 MOV     #ERRLOG,-(R3)   ;Set address of error logging routine
020536  012743  000000G         3$:     MOV     #PTWRD,-(R3)
020542  012743  000000G                 MOV     #PTBYT,-(R3)
020546  012743  000000G                 MOV     #GTBYT,-(R3)
020552  012743  000000G                 MOV     #MPPHY,-(R3)
020556  012743  000000G                 MOV     #RELOC,-(R3)
                                ;
                                ;  Finished
                                ;
020562  012603                          MOV     (SP)+,R3
020564  000207                          RETURN
```

```
                                        ;
                                        ;  Error occured while loading device handler.
                                        ;  R0 = error message address;  CURNAM = device name.
                                        ;
020566  010001                  HLERR:  MOV     R0,R1           ;SAVE ERROR MESSAGE ADDRESS
020570                                  .PRINT  #TSXHD          ;PRINT ERROR MESSAGE HEADING
020576                                  .PRINT  R1              ;PRINT ERROR MESSAGE
020602  013700  000146'                 MOV     CURNAM,R0       ;GET RAD50 DEVICE NAME
020606  004737  025502'                 CALL    PRTR50          ;PRINT DEVICE NAME
020612                                  .PRINT  #CRLF
020620  000137  004250'                 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.
                                ;
020624  010146                  DOHNLC: MOV     R1,-(SP)
020626  010246                          MOV     R2,-(SP)
020630  010346                          MOV     R3,-(SP)
020632  010446                          MOV     R4,-(SP)
020634  010546                          MOV     R5,-(SP)
                                ;
                                ;  Examine 1st word of handler to see if it could have any load-time code.
                                ;
020636  016405  000000G                 MOV     HANENT(R4),R5   ;Get address of handler entry point
020642  004737  021324'                 CALL    HANMAP          ;;;Map Kpar5 to handler if mapped handler
020646  016500  000004                  MOV     4(R5),R0        ;;;Get 1st instruction located at 4 in handler
020652  004737  021400'                 CALL    HANUMP          ;Restore normal mapping
020656  020027  000240                  CMP     R0,#240         ;Is it a NOP?
020662  103516                         BLO     7$              ;Br if can't be any load code
020664  020027  000277                  CMP     R0,#277         ;
020670  101113                         BHI     7$              ;Br if can't be any load code
020672  132700  000004                  BITB    #4,R0           ;Is there load code?
020676  001510                         BEQ     7$              ;Br if not
                                ;
                                ;  Handler may have load code.
                                ;  Read block 0 of handler and get offset to load code.
                                ;
020700  013702  000152'                 MOV     WRKBUF,R2       ;Get addr of our work buffer
020704                                  .READW  #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
020740  022227  031066                  CMP     (R2)+,#^RHAN    ;Is this a new type handler?
020744  001065                         BNE     7$              ;Br if not
020746  016203  000004                  MOV     4(R2),R3        ;Get offset to load code
020752  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.
                                ;
020754  020327  001000                  CMP     R3,#1000        ;Is load code in block 0 of handler?
020760  103424                         BLO     1$              ;Br if yes
020762  010302                         MOV     R3,R2           ;Get offset to start of load code
020764  072227  177767                  ASH     #-9.,R2         ;Convert to a block number
020770  042702  177400                  BIC     #^C377,R2       ;Clear all but block number
020774                                  .READW  #AREA,#1,WRKBUF,#512.,R2 ;Read 2 blocks from handler file
                                ;
                                ;  The load code is now in WRKBUF.  Set up and execute it.
                                ;
021032  010437  000144'         1$:     MOV     R4,CURDEV       ;Save current device index number
021036  042703  177000                  BIC     #^C777,R3       ;Get offset within block of load code entry pt
021042  010300                         MOV     R3,R0           ;Get entry point offset
021044  063700  000152'                 ADD     WRKBUF,R0       ;Add base address
```

```
     58 021050  013701  000000G                 MOV     RPRVEC,R1       ;Get pointer to GETVEC routine for Pro
     59 021054  012702  000000C                 MOV     #MAXDEV*2,R2    ;Get 2** entries in device tables
     60 021060  012703  000004                  MOV     #4,R3           ;Set code saying this is load code
     61 021064  012705  000000G                 MOV     #HANENT,R5      ;Point to handler entry address vector
     62 021070  060405                          ADD     R4,R5           ;Point to entry cell for this handler
     63 021072  004737  021324'                 CALL    HANMAP          ;;;Map Kpar5 to handler if it is a mapped
     64 021076  012704  021154'                 MOV     #LDREAD,R4      ;;;Get pointer to Read routine
     65 021102  004710                          CALL    (R0)            ;;;Execute the load code
     66 021104  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 021106  012737  001400  000000G         MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
     72 021114  004737  021400'                 CALL    HANUMP          ;Unmap the handler
     73 021120  000241                  7$:     CLC                     ;Clear the carry flag for return
     74 021122  000406                          BR      9$
     75                                 ;
     76                                 ;  Error occurred in fetch/load code
     77                                 ;
     78 021124  012737  001400  000000G 2$:      MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
     79 021132  004737  021400'                 CALL    HANUMP          ;Unmap the handler
     80 021136  000261                          SEC                     ;Set the carry flag for return
     81                                 ;
     82                                 ;  Finished
     83                                 ;
     84 021140  012605                  9$:     MOV     (SP)+,R5
     85 021142  012604                          MOV     (SP)+,R4
     86 021144  012603                          MOV     (SP)+,R3
     87 021146  012602                          MOV     (SP)+,R2
     88 021150  012601                          MOV     (SP)+,R1
     89 021152  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
                                ;
021154  010046                  LDREAD: MOV     R0,-(SP)
021156  010446                          MOV     R4,-(SP)        ;;;
021160  010004                          MOV     R0,R4           ;;;Get starting block number
                                ;
                                ;  Save current mapping information
                                ;
021162  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256?
021166  001402                          BEQ     1$              ;;;Br if not
021170  013746  000000G                 MOV     @#SR3MMR,-(SP)  ;;;Save extended memory address register
021174  013746  000000G          1$:    MOV     @#SROMMR,-(SP)  ;;;Save memory mapping
021200  013746  000000G                 MOV     @#KPAR5,-(SP)   ;;;Save current KPAR5 mapping
021204  013746  000000G                 MOV     @#KPAR6,-(SP)   ;;;Save current KPAR6 mapping
021210  012737  001400  000000G         MOV     #1400,@#KPAR6   ;;;Restore original mapping for RT-11
                                ;
                                ;  Turn off handler mapping
                                ;
021216  004737  021400'                 CALL    HANUMP          ;Turn off handler mapping
                                ;
                                ;  Read the requested data from the handler
                                ;
021222                                  .READW  #AREA,#1,R2,R1,R4 ;Read the blocks
021254  103420                          BCS     9$              ;Br if read error
                                ;
                                ;  Restore handler mapping
                                ;
021256  013704  000144'                 MOV     CURDEV,R4       ;Get current device index
021262  004737  021324'                 CALL    HANMAP          ;;;Map Kpar5 if necessary
                                ;
                                ;  Restore mapping information
                                ;
021266  012637  000000G                 MOV     (SP)+,@#KPAR6   ;;;Restore KPAR6 mapping
021272  012637  000000G                 MOV     (SP)+,@#KPAR5   ;;;Restore KPAR5 mapping
021276  012637  000000G                 MOV     (SP)+,@#SROMMR  ;;;Restore memory mapping
021302  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256?
021306  001402                          BEQ     2$              ;;;Br if not
021310  012637  000000G                 MOV     (SP)+,@#SR3MMR  ;;;Restore extended memory address register
                                ;
                                ;  Finished
                                ;
021314  000241                  2$:     CLC                     ;;;Signal success on return
021316  012604                  9$:     MOV     (SP)+,R4
021320  012600                          MOV     (SP)+,R0
021322  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
                                ;
021324                          HANMAP:
                                ;
                                ;  Disable interrupts
                                ;
021324                                  DISABL                  ;;;Disable interrupts
                                ;
                                ;  Change RMON pointer to point to TSX-Plus vector
                                ;
021332  012737  000000G 000000G         MOV     #MONVEC,@#RMON  ;;;Say TSX-Plus is the monitor
                                ;
                                ;  See if this handler is mapped
                                ;
021340  005764  000000G                 TST     HANPAR(R4)      ;;;Is this handler mapped?
021344  001403                          BEQ     9$              ;;;Br if not
                                ;
                                ;  This handler is mapped.
                                ;  Set up mapping to access it.
                                ;
021346  016437  000000G 000000G         MOV     HANPAR(R4),@#KPAR5;;;Map KPAR5 to the handler code
021354  052737  000000G 000000G 9$:     BIS     #MMENBL,@#SROMMR;;;Enable memory mapping
021362  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
021366  001403                          BEQ     10$             ;;;Br if not
021370  052737  000000G 000000G         BIS     #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
                                ;
                                ;  Finished
                                ;
021376  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.
                                ;
021400                          HANUMP:
                                ;
                                ;  Turn off memory management
                                ;
021400  105737  000000G                 TSTB    MEM256          ;;;Does machine have > 256KB?
021404  001403                          BEQ     1$              ;;;Br if not
021406  042737  000000G 000000G         BIC     #EMMAP,@#SR3MMR ;;;Turn off extended memory addressing
021414  042737  000000G 000000G 1$:     BIC     #MMENBL,@#SROMMR;;;Turn off memory mapping
021422  012737  001200  000000G         MOV     #1200,@#KPAR5   ;;;Reset KPAR5 to its normal mapping
```

```
                                        ;
                                        ;  Restore RMON pointer to RT11
                                        ;
021430  013737  000046' 000000          MOV     RTMNVC,@#RMON   ;;;Reset RMON pointer
                                        ;
                                        ;  Enable interrupts
                                        ;
021436                                  ENABL                   ;Enable interrupts
                                        ;
                                        ;  Finished
                                        ;
021444  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.
                                ;
021446  010246                  FNDHRB: MOV     R2,-(SP)
                                ;
                                ;  Search for specified RCB and also remember if we see a free RCB
                                ;
021450  005002                          CLR     R2              ;Say no free RCB found
021452  013701  0000000                 MOV     HANRCB,R1       ;Point to start of RCB area
021456  005721                          TST     (R1)+           ;Skip over -1 word at front
021460  005711                  1$:     TST     (R1)            ;What is the status of this RCB
021462  001002                          BNE     2$              ;Br if this is not a free RCB
021464  010102                          MOV     R1,R2           ;Remember address of a free RCB
021466  000412                          BR      3$
021470  021127  177777          2$:     CMP     (R1),#-1        ;Are we at the end of the list?
021474  001412                          BEQ     4$              ;Br if yes
021476  021561  000006                  CMP     (R5),6(R1)      ;Compare the names
021502  001004                          BNE     3$              ;Br if don't match
021504  026561  000002  000010          CMP     2(R5),10(R1)    ;Compare 2nd half of name
021512  001417                          BEQ     6$              ;Br if found the RCB we were searching for
021514  062701  000012          3$:     ADD     #12,R1          ;Point to the next RCB
021520  000757                          BR      1$              ;Continue searching
                                ;
                                ;  We could not find the specified RCB.
                                ;  If there was a free one, initialize the name.
                                ;
021522  010201                  4$:     MOV     R2,R1           ;Was there a free RCB?
021524  001410                          BEQ     5$              ;Br if not
021526  011561  000006                  MOV     (R5),6(R1)      ;Set name in the RCB
021532  016561  000002  000010          MOV     2(R5),10(R1)    ;(2nd word of name)
021540  063761  000144' 000010          ADD     CURDEV,10(R1)   ;Make name unique to device
021546  000261                  5$:     SEC                     ;Signal that we did not find the RCB
021550  000401                          BR      9$
                                ;
                                ;  We found the RCB
                                ;
021552  000241                  6$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
021554  012602                  9$:     MOV     (SP)+,R2
021556  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.
                                ;
021560  010046                  HANXMR: MOV     R0,-(SP)
                                ;
                                ;  Get the total amount of free memory space available now
                                ;  and see if the requested region can be allocated.
                                ;
021562  013700  000134'               MOV     FMEMHI,R0       ;Top of free memory
021566  163700  000136'               SUB     FMEMLO,R0       ;-Base of free memory
021572  020200                       CMP     R2,R0           ;Do we have room for the requested region?
021574  101006                       BHI     8$              ;Br if not
                                ;
                                ;  There is room for the region so allocate it from the top of memory
                                ;
021576  160237  000134'               SUB     R2,FMEMHI       ;Allocate the region
021602  013701  000134'               MOV     FMEMHI,R1       ;Return the address of the base of the region
021606  000241                       CLC                     ;Signal success on return
021610  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.
                                ;
021612  010002                  8$:  MOV     R0,R2           ;Get the size of the largest possible region
021614  000261                       SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
021616  012600                  9$:  MOV     (SP)+,R0
021620  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$MAP flag is set in the DVFLAG word for devices that require mapping.
                ;
                ;  Inputs:
                ;   R5 = Pointer to low-memory free area
                ;
                ;  Outputs:
                ;   MIOFLAG = 0==>I/O mapping not required for any device;
                ;             1==>I/O mapping required for some device.
                ;   R5 = Pointer to new low-memory free area.
                ;
                SETMIO: MOV     R1,-(SP)
                ;
                ;  Determine if this machine requires mapping at all
                ;
                        TST     #MIODBG         ;Are we debugging mapped I/O system?
                        BNE     2$              ;Br if yes
                        BIT     #EXTLSI,ICONFG  ;Is this a Q-bus machine with more than 256Kb?
                        BNE     2$              ;Br if yes
                ;
                ;  This is not a Q-bus system with more than 256Kb.
                ;  Mapping is not required at all.
                ;
                        MOV     #2,R1           ;Get initial device index number
                1$:     BIC     #DX$MAP,DVFLAG(R1) ;Clear mapped-I/O flag
                        ADD     #2,R1           ;Get next device index
                        CMP     R1,NUMDEV       ;More to do?
                        BLOS    1$              ;Br if yes
                        BR      9$
                ;
                ;  This is a Q-bus system with more than 256Kb.
                ;  See if any devices have requested mapped I/O.
                ;
                2$:     CLR     R0              ;Clear composite flag word
                        MOV     #2,R1           ;Initialize device index number
                3$:     BIS     DVFLAG(R1),R0   ;Combine flags from all devices
                        ADD     #2,R1           ;Get next device index number
                        CMP     R1,NUMDEV       ;Checked all devices?
                        BLOS    3$              ;Br if not
                        BIT     #DX$MAP,R0      ;Does any device require mapping?
                        BEQ     9$              ;Br if not
                ;
                ;  I/O mapping is required
                ;
                        INCB    MIOFLG          ;Remember that mapping is required
                ;
                ;  Zero the area where we will build the control structures
                ;
                        MOVB    VMIOBF,R1       ;Get # buffers wanted
                        MUL     #MI$$SZ,R1      ;Times size for each control block
                        ADD     #MIONWB*MW$$SZ,R1;Add space for MIO wait blocks
```

```
                                        ASR     R1              ;Get # words to zero
                                        MOV     R5,R0           ;Get pointer to start of area
                                4$:     CLR     (R0)+           ;Zero the entire area
                                        SOB     R1,4$
                                ;
                                ;  Allocate and initialize the control structures
                                ;
                                        MOV     R5,MIOBHD       ;Start of area for I/O mapping control blks
                                        MOVB    VMIOBF,R0       ;Get # buffers wanted
                                5$:     MOV     R5,R1           ;Get pointer to current control block
                                        ADD     #MI$$SZ,R1      ;Get pointer to next control block
                                        DEC     R0              ;Need to link more together?
                                        BLE     6$              ;Br if not
                                        MOV     R1,MI$LNK(R5)   ;Set pointer to next control block
                                        MOV     R1,R5           ;Advance pointer to next block
                                        BR      5$              ;See if more to do
                                6$:     MOV     R1,R5           ;Set pointer past last block
                                        MOV     R5,MIOWHD       ;Start of wait blocks
                                        MOV     #MIONWB-1,R0    ;Get # wait blocks wanted - 1
                                7$:     MOV     R5,R1           ;Get pointer to current block
                                        ADD     #MW$$SZ,R1      ;Get pointer to next block
                                        MOV     R1,MW$LNK(R5)   ;Set pointer to next wait block
                                        MOV     R1,R5           ;Advance pointer to next block
                                        SOB     R0,7$           ;Loop if more to allocate
                                        ADD     #MW$$SZ,R5      ;Allocate space for last block (with 0 link)
                                ;
                                ;  Finished
                                ;
                                9$:     MOV     (SP)+,R1
                                        RETURN
                                        .IFF    ;NE,<PROASM-1>
                                ;
                                ;  Define dummy routine for Pro
                                ;
                                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$FLG(seg) = 0==>Load seg into high memory; 1==>Load over TSINIT.
                                ;   OVLBAS = Address of location within TSINIT where overlays start.
                                ;   R5 = Pointer past last overlay loaded over TSINIT.
                                ;
021624  010146          OVLPOS: MOV     R1,-(SP)
021626  010246                  MOV     R2,-(SP)
021630  010346                  MOV     R3,-(SP)
021632  010446                  MOV     R4,-(SP)
021634  010504                  MOV     R5,R4           ;Get address where we may load overlays
                                ;
                                ;  Build the table that holds information about the overlays
                                ;
021636  004737  022034'         CALL    OVLBLD          ;Build overlay information table
                                ;
                                ;  First determine how much space will be used by those overlays that are
                                ;  forced to be loaded over TSINIT.
                                ;
021642  062704  000077          ADD     #63.,R4         ;Bound address to 64-byte boundary
021646  042704  000077          BIC     #77,R4
021652  010437  000140'          MOV     R4,OVLBAS       ;Remember address where we load overlays
021656  012702  000576'          MOV     #OSTABL,R2      ;Point to start of table
021662  005762  000000  1$:     TST     OS$SIZ(R2)      ;Is this overlay to be loaded?
021666  001423                  BEQ     2$              ;Br if not
021670  016200  000004          MOV     OS$OVL(R2),R0   ;Point to linker-built entry
021674  016000  000000G         MOV     0.ADR(R0),R0    ;Get Rad50 segment ID
021700  012701  001026'          MOV     #LOWOVL,R1      ;Point to table of overlays to go over TSINIT
021704  020021          6$:     CMP     R0,(R1)+        ;Must this overlay go over TSINIT?
021706  001404                  BEQ     4$              ;Br if yes
021710  020127  001034'          CMP     R1,#LOWEND      ;End of low-overlay table?
021714  103773                  BLO     6$              ;Br if not
021716  000407                  BR      2$              ;This overlay is not forced over TSINIT
021720  005262  000002  4$:     INC     OS$FLG(R2)      ;Set flag saying load over TSINIT
021724  016200  000000          MOV     OS$SIZ(R2),R0   ;Get # 64-byte blocks needed for overlay
021730  072027  000006          ASH     #6,R0           ;Get # bytes needed for overlay
021734  060004                  ADD     R0,R4           ;Advance address within TSINIT
021736  062702  000006  2$:     ADD     #OS$$SZ,R2      ;Point to entry for next segment
021742  020237  001024'          CMP     R2,OSLAST       ;Have we finished?
021746  103745                  BLO     1$              ;Loop if not
                                ;
                                ;  Determine how much memory space is available in TSINIT for other overlays
                                ;
021750  020427  025474'          CMP     R4,#INITOP-50.  ;Any space left for other overlays?
021754  103021                  BHIS    9$              ;Br if not
021756  012705  025556'          MOV     #INITOP,R5      ;Point to top of overlay area
021762  160405                  SUB     R4,R5           ;Total space available for overlays
021764  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.
                                        ;
021770  004737  023226'        3$:     CALL    OVLTRY          ;Try to find largest overlay that will fit
021774  103411                        BCS     9$              ;Br if no more overlays will fit
021776  005262  000002                INC     OS$FLG(R2)      ;Remember to load over TSINIT
022002  016200  000000                MOV     OS$SIZ(R2),R0   ;Get # 64-byte blocks needed for overlay
022006  160005                        SUB     R0,R5           ;Reduce remaining free space in TSINIT
022010  072027  000006                ASH     #6,R0           ;Get # bytes needed for overlay
022014  060004                        ADD     R0,R4           ;Advance overlay address in TSINIT
022016  000764                        BR      3$              ;See if we can find more segments to load
                                    ;
                                    ;  Finished
                                    ;
022020  010405              9$:     MOV     R4,R5           ;Return top-of-overlay address in R5
022022  012604                        MOV     (SP)+,R4
022024  012603                        MOV     (SP)+,R3
022026  012602                        MOV     (SP)+,R2
022030  012601                        MOV     (SP)+,R1
022032  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.
                                ;
022034  010146                  OVLBLD: MOV     R1,-(SP)
022036  010246                          MOV     R2,-(SP)
022040  010346                          MOV     R3,-(SP)
                                ;
                                ;  Read 1st block of SAV file to get pointer to overlay table
                                ;
022042  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
022046                                  .READW  #AREA,#17,R2,#256.,#0 ;read first block of the save file
022102  103444                          BCS     22$             ;Br if error on read
022104  016201  000064                  MOV     64(R2),R1       ;point to the overlay table
022110  001012                          BNE     15$             ;br if overlays exist
                                ;
                                ;  Must be verion 3B overlays structure at absolute location.
                                ;
022112  012737  000137  001000          MOV     #137,@#1000     ;position jump instrucion over 3b ovly handler
022120  012737  000000G 001002          MOV     #$OVRH,@#1002   ;position overlay intercept location
022126  012701  001104                  MOV     #1104,R1        ;point to the overlay table
022132  010137  000000G                 MOV     R1,OVRADD       ;save the address of the overlay table
                                ;
                                ;  Initialize the table that holds information about the overlays
                                ;
022136  012703  000576'         15$:    MOV     #OSTABL,R3      ;Point to table for overlay info
022142  010163  000004          11$:    MOV     R1,OS$OVL(R3)   ;Save pointer to overlay control block
022146  005063  000002                  CLR     OS$FLG(R3)      ;Assume seg will be loaded in high memory
022152  004737  022354'                 CALL    ALCOVL          ;Determine if we should load this overlay
022156  010263  000000                  MOV     R2,OS$SIZ(R3)   ;Remember total size of overlay+data
022162  062703  000006                  ADD     #OS$$SZ,R3      ;Point to next overlay table entry
022166  062701  000006          12$:    ADD     #6,R1           ;find the next region
022172  021127  004537                  CMP     (R1),#4537      ;compare with a <JSR R5,$OVRH> instruction
022176  001361                          BNE     11$             ;Br if not at end
022200  010337  001024'                 MOV     R3,OSLAST       ;Save pointer past last overlay table entry
                                ;
                                ;  Finished
                                ;
022204  012603                          MOV     (SP)+,R3
022206  012602                          MOV     (SP)+,R2
022210  012601                          MOV     (SP)+,R1
022212  000207                          RETURN
                                ;
                                ;  Error -- Read error occured while reading overlay table
                                ;
022214                          22$:    .PRINT  #TSXHD
022222                                  .PRINT  #RDERR
022230  000137  004250'                 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.
                                ;
022234  010146                  GETMAP: MOV     R1,-(SP)
022236  010246                          MOV     R2,-(SP)
022240  010346                          MOV     R3,-(SP)
022242  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.
                                ;
022246  012703  000576'                 MOV     #OSTABL,R3      ;Point to 1st overlay table entry
022252  004737  022520'          1$:    CALL    OPTOVL          ;See if this segment should be loaded
022256  010263  000000                  MOV     R2,OS$SIZ(R3)   ;Save # 64-byte blocks needed for overlay
022262  062703  000006                  ADD     #OS$$SZ,R3      ;Point to next overlay table entry
022266  020337  001024'                 CMP     R3,OSLAST       ;Checked all entries in overlay table?
022272  103767                          BLO     1$              ;Br if not
                                ;
                                ;  Load those overlays that go into high memory
                                ;
022274  012702  000576'                 MOV     #OSTABL,R2      ;Point to 1st overlay entry
022300  005762  000000          3$:    TST     OS$SIZ(R2)      ;Is this overlay segment wanted?
022304  001405                          BEQ     4$              ;Br if not
022306  005762  000002                  TST     OS$FLG(R2)      ;Load over TSINIT or into high memory?
022312  001002                          BNE     4$              ;Br if load over TSINIT
022314  004737  023324'                 CALL    GETOVL          ;Load overlay into high memory
022320  062702  000006          4$:    ADD     #OS$$SZ,R2      ;Point to next overlay table entry
022324  020237  001024'                 CMP     R2,OSLAST       ;Have we done all overlays?
022330  103763                          BLO     3$              ;Loop if not
                                ;
                                ;  Finished
                                ;
022332  013700  000000G          19$:   MOV     SMRSIZ,R0       ;Get memory pointer at start of allocation
022336  160500                          SUB     R5,R0           ;Calc amt of space allocated
022340  010037  000000G                 MOV     R0,SMRSIZ       ;Save total space used for mapped regions
022344  012603                          MOV     (SP)+,R3
022346  012602                          MOV     (SP)+,R2
022350  012601                          MOV     (SP)+,R1
022352  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$xxx)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> This segment is to be loaded.
                                ;   C-flag set     ==> Do not load this overlay segment.
                                ;   R2 = # 64-Byte blocks needed for segment including data areas within it.
                                ;
022354  010146                 ALCOVL: MOV     R1,-(SP)
                                ;
                                ;  Get pointer to linker-build overlay entry for segment
                                ;
022356  016301  000004                  MOV     OS$OVL(R3),R1   ;Get pointer to linker-built entry for seg
                                ;
                                ;  Read in the first block of the overlay segment
                                ;
022362  013702  000152'                 MOV     WRKBUF,R2       ;Point to work buffer
022366                                  .READW  #AREA,#17,R2,#256.,0.BLK(R1) ;read the first block
022424  103415                          BCS     3$              ;Br if read error
                                ;
                                ;  Save the 3 character Rad50 segment ID in the 0.ADR cell of the
                                ;  linker-built overlay table entry for this segment.
                                ;
022426  016261  000002  000000G         MOV     2(R2),0.ADR(R1) ;save the rad50 overlay identifier
                                ;
                                ;  Make sure the segment is not larger than 8Kb
                                ;
022434  016102  000000G                 MOV     0.SIZ(R1),R2    ;get the word count of the code region
022440  006302                          ASL     R2              ;convert to byte count
022442  020227  020000                  CMP     R2,#20000       ;check for 8kb overflow
022446  101014                          BHI     21$             ;Br if region is too big
                                ;
                                ;  Don't load some optional segments if features were not selected
                                ;  in TSGEN.
                                ;
022450  004737  022520'                 CALL    OPTOVL          ;See if we want to load this segment
                                ;
                                ;  Finished
                                ;  The C-flag is set or reset by OPTOVL.
                                ;
022454  012601                          MOV     (SP)+,R1
022456  000207                          RETURN
                                ;
                                ;  Error -- Error on reading from SAV file
                                ;
022460                          3$:     .PRINT  #TSXHD          ;Print heading
022466                                  .PRINT  #RDERR          ;Read error
022474  000137  004250'                 JMP     INISTP          ;Abort initialization
                                ;
                                ;  Error -- Insufficient memory space to load run-time systems
                                ;
```

```
     58 022500                          21$:    .PRINT  #TSXHD          ;PRINT HEADING
     59 022506                                  .PRINT  #TSXSIZ         ;PRINT ERROR MESSAGE
     60 022514  000137  004250'                 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$xxx)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Load this overlay.
                                ;   C-flag set     ==> Do not load this overlay.
                                ;   R2 = # 64-byte blocks needed for code + data for the segment.
                                ;
022520  010346                  OPTOVL: MOV     R3,-(SP)
022522  010446                          MOV     R4,-(SP)
022524  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get the name of the overlay segment
                                ;
022526  016305  000004                  MOV     OS$OVL(R3),R5   ;Get pointer to linker-built entry
022532  016504  000000G                 MOV     O.ADR(R5),R4    ;Get name of the segment
                                ;
                                ;  Get size of code portion of overlay segment
                                ;
022536  016502  000000G                 MOV     O.SIZ(R5),R2    ;Get # words needed by code portion of seg
022542  006302                          ASL     R2              ;Convert to # bytes
                                ;
                                ;  See if this is an optional segment that we need to deal with specially
                                ;
022544  012700  022570'                 MOV     #OVLLST,R0      ;Point to overlay name list
022550  020420                  1$:     CMP     R4,(R0)+        ;Found name of overlay?
022552  001405                          BEQ     2$              ;Br if yes
022554  005720                          TST     (R0)+           ;No -- Skip over address word
022556  020027  022654'                 CMP     R0,#OVLEND      ;Checked all names in the list?
022562  103772                          BLO     1$              ;Loop if not
022564  000577                          BR      OOXYES          ;Load this overlay
                                ;
                                ;  Branch off to processing routine
                                ;
022566  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
                                ;
022570                          OVLLST:
022570                                  OVLTBL  USR             ;TSUSR  -- File management
022574                                  OVLTBL  SPL             ;TSSPOL -- Spooling system
022600                                  OVLTBL  LOK             ;TSLOCK -- Shared file record locking
022604                                  OVLTBL  MSG             ;TSMSG  -- Inter-job message communication
022610                                  OVLTBL  SWP             ;TSSWAP -- Job swapper
022614                                  OVLTBL  PLS             ;TSPLAS -- PLAS support
022620                                  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
                                ;
022654  013703  000000G       OORUSR: MOV     VNFCSH,R3       ;Get # file cache entries
022660  070327  000000G               MUL     #FC$$SZ,R3      ;Multiply by size of each entry
022664  060302                       ADD     R3,R2           ;Allocate space for directory cache
022666  000536                       BR      OOXYES          ;Load the segment
                                ;
                                ;  Spooling system
                                ;
022670  005727  000000G       OORSPL: TST     #SPLND          ;Are there any spooled devices?
022674  001530                       BEQ     OOXNO           ;Br if not
022676  062702  000000C               ADD     #<SPLNB*512.>,R2;Reserve room for spool buffers
022702  013703  000000G               MOV     NSPLBL,R3       ;Get # blocks for spool file
022706  062703  000007               ADD     #7,R3           ;Bound up to byte boundary
022712  072327  177775               ASH     #-3,R3          ;Divide by 8 to get # bytes for table
022716  005203                       INC     R3              ;Round up to word boundary
022720  042703  000001               BIC     #1,R3
022724  060302                       ADD     R3,R2           ;Add space for spool file allocation table
022726  000516                       BR      OOXYES          ;Load the segment
                                ;
                                ;  Record locking system
                                ;
022730  005737  000000G       OORLOK: TST     VMXSF           ;Any shared files?
022734  001510                       BEQ     OOXNO           ;Br if not
022736  005737  000000G               TST     VNUMDC          ;Shared file data caching wanted?
022742  001110                       BNE     OOXYES          ;Br if yes
022744  162702  000000G               SUB     #DCCSIZ,R2      ;Reduce size of segment - Leave out cache code
022750  000505                       BR      OOXYES          ;Load the segment
                                ;
                                ;  Message communication system
                                ;
022752  013703  000000G       OORMSG: MOV     VMAXMC,R3       ;Is message communication facility wanted?
022756  001477                       BEQ     OOXNO           ;Br if not
022760  070327  000000G               MUL     #MB$$SZ,R3      ;Space for message channel blocks
022764  060302                       ADD     R3,R2
022766  013703  000000G               MOV     VMXMRB,R3       ;Number of message request blocks
022772  070327  000000G               MUL     #MR$$SZ,R3      ;Times size of request block
022776  060302                       ADD     R3,R2
023000  013703  000000G               MOV     VMSCHR,R3       ;Max # chars in a message
023004  005203                       INC     R3              ;Bound up to word
023006  042703  000001               BIC     #1,R3           ;Reserve whole number of words
023012  062703  000000G               ADD     #MU$TXT,R3      ;Plus space for message header
023016  070337  000000G               MUL     VMXMSG,R3       ;Times maximum number of messages
023022  060302                       ADD     R3,R2           ;Space for message buffers
023024  000457                       BR      OOXYES
                                ;
                                ;  PLAS support
                                ;
023026  013703  000000G       OORPLS: MOV     VPLAS,R3        ;PLAS support wanted?
```

```
    115 023032  001451                        BEQ     OOXNO           ;Br if not
    116 023034  062703  000021                ADD     #17.,R3         ;Bound up # blocks
    117 023040  072327  177775                ASH     #-3,R3          ;Get # bytes needed for swap file bit map
    118 023044  060302                        ADD     R3,R2           ;Reserve room for swap file bit map
    119 023046  000446                        BR      OOXYES          ;Load the segment
    120                             ;
    121                             ;  Job swapper
    122                             ;
    123 023050  105737  000000G     OORSWP: TSTB    VSWPFL          ;Is this a swapping system?
    124 023054  001440                        BEQ     OOXNO           ;Br if not
    125 023056  000442                        BR      OOXYES          ;Br if yes -- Load the segment
    126                             ;
    127                             ;  Single line editor
    128                             ;
    129 023060  105737  000000G     OORSLE: TSTB    VSLEDT          ;Is SL editor wanted?
    130 023064  001434                        BEQ     OOXNO           ;Br if not
    131 023066  000436                        BR      OOXYES          ;Load the segment
    132                             ;
    133                             ;  Display windows
    134                             ;
    135 023070  013703  000000G     OORWIN: MOV     VMXWIN,R3       ;Are any display windows wanted?
    136 023074  001430                        BEQ     OOXNO           ;Br if not
    137 023076  070327  000000G               MUL     #DW$$SZ,R3      ;Amt of space needed for window control blks
    138 023102  060302                        ADD     R3,R2           ;Add to size of overlay
    139 023104  000427                        BR      OOXYES          ;Load the segment
    140                             ;
    141                             ;  Mapped I/O
    142                             ;
    143 023106  105737  000000G     OORMIO: TSTB    MIOFLG          ;Is I/O mapping needed?
    144 023112  001421                        BEQ     OOXNO           ;Br if not
    145 023114  000423                        BR      OOXYES          ;Load the segment
    146                             ;
    147                             ;  CL handler
    148                             ;
    149 023116  005727  000000G     OORCLO: TST     #CLTOTL         ;Any I/O lines?
    150 023122  001415                        BEQ     OOXNO           ;Br if not
    151 023124  000417                        BR      OOXYES          ;Yes, load the segment
    152                             ;
    153                             ;  Program debugger
    154                             ;
    155 023126  105737  000000G     OORDBG: TSTB    VDBFLG          ;Is the program debugger wanted?
    156 023132  001411                        BEQ     OOXNO           ;Br if not
    157 023134  000413                        BR      OOXYES          ;Load this segment
    158                             ;
    159                             ;  Data caching
    160                             ;
    161 023136  005737  000000G     OORCSH: TST     CSHALC          ;Is data caching wanted?
    162 023142  001405                        BEQ     OOXNO           ;Br if not
    163 023144  000407                        BR      OOXYES          ;Load this segment
    164                             ;
    165                             ;  Crash dump generator
    166                             ;
    167 023146  105737  000000G     OORDMP: TSTB    VSYDMP          ;Is dump facility wanted?
    168 023152  001401                        BEQ     OOXNO           ;Br if not
    169 023154  000403                        BR      OOXYES          ;Br if yes
    170                             ;
    171                             ;  Don't load this segment
```

```
    172                                 ;
    173 023156  005002                  OOXNO:  CLR     R2              ;Say no space needed for overlay
    174 023160  000261                          SEC                     ;Signal don't load the segment
    175 023162  000415                          BR      OOXFIN
    176                                 ;
    177                                 ;  Load this segment
    178                                 ;
    179 023164  005202                  OOXYES: INC     R2              ;Make sure size is even
    180 023166  042702  000001                  BIC     #1,R2
    181 023172  020227  020000                  CMP     R2,#8192.       ;Don't allow code + data to exceed 8Kb
    182 023176  101402                          BLOS    1$              ;Br if ok
    183 023200  012702  020000                  MOV     #8192.,R2       ;Note, init code in segment will truncate dat
    184 023204  062702  000077          1$:     ADD     #63.,R2         ;Convert to # 64-byte blocks
    185 023210  072227  177772                  ASH     #-6,R2
    186 023214  000241                          CLC                     ;Signal to load the segment
    187                                 ;
    188                                 ;  Finished
    189                                 ;
    190 023216  012605                  OOXFIN: MOV     (SP)+,R5
    191 023220  012604                          MOV     (SP)+,R4
    192 023222  012603                          MOV     (SP)+,R3
    193 023224  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.
                                ;
023226  010346                  OVLTRY: MOV     R3,-(SP)
                                ;
                                ;  Begin loop to examine all segments
                                ;
023230  005002                          CLR     R2              ;Say we haven't found any segment yet
023232  012703  000576'                 MOV     #OSTABL,R3      ;Point to entry for 1st segment
023236  005763  000000          1$:     TST     OS$SIZ(R3)      ;Is this segment to be loaded?
023242  001415                          BEQ     2$              ;Br if not
023244  005763  000002                  TST     OS$FLG(R3)      ;Is this segment already over TSINIT?
023250  001012                          BNE     2$              ;Br if yes
023252  026305  000000                  CMP     OS$SIZ(R3),R5   ;Will this segment fit?
023256  101007                          BHI     2$              ;Br if not
023260  005702                          TST     R2              ;Have we found any other seg yet?
023262  001404                          BEQ     3$              ;Br if not
023264  026362  000000  000000          CMP     OS$SIZ(R3),OS$SIZ(R2) ;Is new seg larger than old?
023272  101401                          BLOS    2$              ;Br if not
023274  010302                  3$:     MOV     R3,R2           ;Remember largest segment
023276  062703  000006          2$:     ADD     #OS$$SZ,R3      ;Point to entry for next segment
023302  020337  001024'                 CMP     R3,OSLAST       ;Have we checked all segments?
023306  103753                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
023310  000241                          CLC                     ;Assume we found a segment
023312  005702                          TST     R2              ;Did we find a segment that will fit?
023314  001001                          BNE     9$              ;Br if yes
023316  000261                          SEC                     ;Signal failure on return
023320  012603                  9$:     MOV     (SP)+,R3
023322  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.
                                ;
023324                          GETOVL:
                                ;
                                ;  Allocate space for the overlay segment
                                ;
023324  166205  000000                  SUB     OS$SIZ(R2),R5   ;Allocate space for overlay
023330  020527  001600                  CMP     R5,#1600        ;Are we about to run over RT-11?
023334  103405                          BLO     10$             ;Br if yes -- Insufficient memory
                                ;
                                ;  Remember the base address of some key segments
                                ;
023336  004737  004636'                 CALL    KEYSEG          ;Remember address of some segments
                                ;
                                ;  Load the segment
                                ;
023342  004737  023370'                 CALL    LODOVL          ;Load the segment
                                ;
                                ;  Finished
                                ;
023346  000207                          RETURN
                                ;
                                ;  Error: Memory overflow
                                ;
023350                          10$:    .PRINT  #TSXHD
023356                                  .PRINT  #TSXSIZ
023364  000137  004250'                 JMP     INISTP
```

```
 1                                                .SBTTL  LODOVL -- Read and relocate system overlay
 2                                      ;-----------------------------------------------------------------------
 3                                      ;  LODOVL is called to load a system overlay region into memory.
 4                                      ;
 5                                      ;  Inputs:
 6                                      ;   R2 = Pointer to OSTABL entry for segment being loaded.
 7                                      ;   R5 = 64-byte physical memory block number where segment is to be loaded.
 8                                      ;
 9 023370  010146                      LODOVL: MOV     R1,-(SP)
10 023372  010246                              MOV     R2,-(SP)
11 023374  010346                              MOV     R3,-(SP)
12 023376  010446                              MOV     R4,-(SP)
13 023400  010546                              MOV     R5,-(SP)
14
15                                      ;
16                                      ;  Get info about size of the overlay and position within SAV file
17                                      ;
18 023402  016201  000004                      MOV     OS$OVL(R2),R1   ;Get pointer to linker-built segment entry
19 023406  016103  000000G                     MOV     O.SIZ(R1),R3    ;Get size of overlay segment (# words)
20 023412  016137  000000G 000142'             MOV     O.BLK(R1),FILBLK;Get block in SAV file where segment starts
21 023420  010302                              MOV     R3,R2           ;Get total number of words in segment
22 023422  062702  000377                      ADD     #255.,R2        ;round to the nearest number of blocks
23 023426  000302                              SWAB    R2              ;Divide by 256. words per segment
24 023430  042702  177400                      BIC     #177400,R2      ;kill sign extension bits
25 023434  010561  000000G                     MOV     R5,O.PAR(R1)    ;Remember where segment is being loaded
26                                      ;
27                                      ;  Read next block of overlay segment into low-memory buffer
28                                      ;
29 023440  013704  000152'             10$:    MOV     WRKBUF,R4       ;Point to work buffer
30 023444                                      .READW  #AREA,#17,R4,#256.,FILBLK       ;read a block
31 023502  103466                              BCS     22$             ;read error occured
32                                      ;
33                                      ;  Move from low buffer to high position in memory
34                                      ;
35 023504  012701  000000G                     MOV     #VPAR5,R1       ;get the virtual address of the mapped region
36 023510  012700  000400                      MOV     #256.,R0        ;obtain the number of words to move
37 023514  020300                              CMP     R3,R0           ;Do we need to move as many as 256 words?
38 023516  103001                              BHIS    2$              ;Br if yes
39 023520  010300                              MOV     R3,R0           ;Get number of words to move for last block
40 023522  160003                      2$:     SUB     R0,R3           ;Get number of words left after this move
41 023524                                      DISABL                  ;** Disable interrupts **
42 023532  013746  000000G                     MOV     @#KPAR5,-(SP)   ;save the contents of the mapping register
43 023536  010537  000000G                     MOV     R5,@#KPAR5      ;change the mapping register
44 023542  052737  000000G 000000G             BIS     #MMENBL,@#SROMMR;enable memory management
45 023550  105737  000000G                     TSTB    MEM256          ;Does machine have at least 256Kb of memory?
46 023554  001403                              BEQ     11$             ;Br if not
47 023556  052737  000000G 000000G             BIS     #EMMAP,@#SR3MMR ;enable extended memory addressing
48 023564  012421                      11$:    MOV     (R4)+,(R1)+     ;move into high memory
49 023566  077002                              SOB     R0,11$
50 023570  105737  000000G                     TSTB    MEM256          ;Does this machine have at least 256Kb?
51 023574  001403                              BEQ     12$             ;Br if not
52 023576  042737  000000G 000000G             BIC     #EMMAP,@#SR3MMR ;disable extended memory management
53 023604  042737  000000G 000000G 12$:     BIC     #MMENBL,@#SROMMR;disable memory management
54 023612  012637  000000G                     MOV     (SP)+,@#KPAR5   ;restore the mapping register
55 023616                                      ENABL                   ;** Enable interrupts **
56 023624  062705  000010                      ADD     #10,R5          ;advance 64-byte block # by 512-bytes
57 023630  005237  000142'                     INC     FILBLK          ;increment file block #
```

```
     58 023634  005302                          DEC     R2              ;More to be copied?
     59 023636  001402                          BEQ     5$              ;Br if not
     60 023640  000137  023440'                 JMP     10$             ;Read and copy rest of mapped segment
     61                                ;
     62                                ;  Finished loading the segment
     63                                ;
     64 023644  012605                 5$:     MOV     (SP)+,R5
     65 023646  012604                         MOV     (SP)+,R4
     66 023650  012603                         MOV     (SP)+,R3
     67 023652  012602                         MOV     (SP)+,R2
     68 023654  012601                         MOV     (SP)+,R1
     69 023656  000207                         RETURN
     70                                ;
     71                                ;  Error occurred on read
     72                                ;
     73 023660                         22$:    .PRINT  #TSXHD          ;Print heading
     74 023666                                 .PRINT  #RDERR          ;Read error
     75 023674  000137  004250'                 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
                                ;
023700  010146                  GETSRT: MOV     R1,-(SP)
023702  010246                          MOV     R2,-(SP)
023704  010346                          MOV     R3,-(SP)
023706  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if this is a dummy run-time entry to allow for patching
                                ;
023710  021127  000000G                 CMP     (R1),#DMYDEV    ;Dummy run-time entry?
023714  001540                          BEQ     7$              ;Br if yes
                                ;
                                ;  Try to open a channel to run-time file
                                ;
023716                                  .LOOKUP #AREA,#1,R1     ;OPEN CHANNEL TO RUN-TIME FILE
023734  103010                          BCC     8$              ;BR IF OPEN WAS SUCCESSFUL
                                ;
                                ;  Cannot open shared run-time file.
                                ;  See if he wants to abort or continue.
                                ;
023736  105737  000000G                 TSTB    VINABT          ;ABORT OR CONTINUE
023742  001132                          BNE     9$              ;BR IF ABORT WANTED
023744  005061  000000G                 CLR     RT$NAM(R1)      ;Mark run-time as not-available
023750  005061  000020G                 CLR     RT$NAM+2(R1)
023754  000520                          BR      7$              ;GO LOAD NEXT RUN-TIME SYSTEM
                                ;
                                ;  Set up information about position of run-time in physical memory
                                ;
023756  116102  000000G         8$:     MOVB    RT$SKP(R1),R2   ;GET # BLOCKS TO SKIP AT FRONT OF RUN-TIME
023762  042702  177400                  BIC     #^C377,R2       ;CLEAR SIGN EXTENSION
023766  160200                          SUB     R2,R0           ;GET # BLOCKS TO READ (LOOKUP SET R0 W  SIZE)
023770  010561  000000G                 MOV     R5,RT$TOP(R1)   ;SET 64-BYTE BLOCK # ABOVE TOP OF RUN-TIME
023774  010003                          MOV     R0,R3           ;GET # 512-BYTE BLOCKS IN RUN-TIME
023776  072027  000003                  ASH     #3,R0           ;CONVERT TO # 64-BYTE BLOCKS
024002  160005                          SUB     R0,R5           ;CALCULATE BASE 64-BYTE BLOCK # OF RUN-TIME
024004  020527  001600                  CMP     R5,#1600        ;ARE WE ABOUT TO RUN OVER RT-11?
024010  103530                          BLO     11$             ;BR IF YES
024012  010561  000000G                 MOV     R5,RT$BAS(R1)   ;SET BASE 64-BYTE BLOCK # OF RUN-TIME
                                ;
                                ;  Read run-time system into memory and position in high-memory
                                ;
024016  010546                          MOV     R5,-(SP)        ;Save address of bottom of run-time
024020  013704  000152'         4$:     MOV     WRKBUF,R4       ;Point to work buffer
024024                                  .READW  #AREA,#1,R4,#256.,R2 ;READ A BLOCK OF RUN-TIME FILE
                                ;  Use memory management to access high-memory area.
024060  012701  000000G                 MOV     #VPAR6,R1       ;GET VIRTUAL ADDRESS OF PAR6 ADDRESS REGION
024064  010537  000000G                 MOV     R5,@#UPAR6      ;SET USER-MODE PAR6 MAP OFFSET VALUE
```

```
     58 024070  012737  077406  000000G          MOV     #077406,@#UPDR6 ;SET PDR TO ALLOW FULL ACCESS TO PAGE
     59 024076  052737  000000G 000000G          BIS     #UPMODE,@#PSW   ;SET PREVIOUS-MODE = USER FOR MTPD ACCESS
     60 024104  012700  000400                    MOV     #256.,R0        ;GET # WORDS TO MOVE
     61 024110                                    DISABL                  ;** Disable interrupts **
     62 024116  052737  000000G 000000G          BIS     #MMENBL,@#SROMMR;enable memory management
     63 024124  105737  000000G                   TSTB    MEM256          ;DOES THIS MACHINE HAVE AT LEAST 256KB?
     64 024130  001403                            BEQ     3$              ;BR IF NOT
     65 024132  052737  000000G 000000G          BIS     #EMMAP,@#SR3MMR ;enable extended memory addressing
     66 024140  012446                    3$:     MOV     (R4)+,-(SP)     ;TRANSFER DATA FROM BUFFER TO HIGH MEMORY
     67 024142  106621                            MTPD    (R1)+
     68 024144  077003                            SOB     R0,3$
     69 024146  105737  000000G                   TSTB    MEM256          ;DOES THIS MACHINE HAVE AT LEAST 256KB?
     70 024152  001403                            BEQ     31$             ;BR IF NOT
     71 024154  042737  000000G 000000G          BIC     #EMMAP,@#SR3MMR ;DISABLE EXTENDED MEMORY MANAGEMENT
     72 024162  042737  000000G 000000G 31$:     BIC     #MMENBL,@#SROMMR;DISABLE MEMORY MANAGEMENT
     73 024170                                    ENABL                   ;** Enable interrupts **
     74 024176  062705  000010                    ADD     #10,R5          ;ADVANCE 64-BYTE BLOCK # BY 512-BYTES
     75 024202  005202                            INC     R2              ;INC FILE BLOCK #
     76 024204  077373                            SOB     R3,4$           ;READ AND COPY REST OF FILE
     77                                     ;
     78                                     ;  Finished loading the run-time system.
     79                                     ;
     80 024206  012605                            MOV     (SP)+,R5
     81 024210                                    .CLOSE  #1
     82                                     ;
     83                                     ;  Finished
     84                                     ;
     85 024216  012604                    7$:     MOV     (SP)+,R4
     86 024220  012603                            MOV     (SP)+,R3
     87 024222  012602                            MOV     (SP)+,R2
     88 024224  012601                            MOV     (SP)+,R1
     89 024226  000207                            RETURN
     90                                     ;
     91                                     ;  Error -- Cannot find run-time system file
     92                                     ;
     93 024230                              9$:     .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
     94 024236                                      .PRINT  #COSRT          ;PRINT ERROR MESSAGE
     95 024244  012702  000004                    MOV     #4,R2           ;PRINT 4 RAD50 VALUES
     96 024250  012100                      10$:    MOV     (R1)+,R0        ;GET PART OF NAME
     97 024252  004737  025502'                   CALL    PRTR50          ;PRINT RAD50 VALUE
     98 024256  077204                            SOB     R2,10$
     99 024260                                    .PRINT  #CRLF           ;END LINE
    100 024266  000137  004250'                   JMP     INISTP          ;ABORT INITIALIZATION
    101                                     ;
    102                                     ;  Error -- Insufficient memory space to load run-time systems
    103                                     ;
    104 024272                              11$:    .PRINT  #TSXHD          ;PRINT HEADING
    105 024300                                      .PRINT  #SRTOVF         ;PRINT ERROR MESSAGE
    106 024306  000137  004250'                   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.
                                ;
024312  010246                 CSHBUF: MOV     R2,-(SP)
024314  010346                         MOV     R3,-(SP)
                                ;
                                ;  See if data caching is wanted
                                ;
024316  013737  000000G 000000G         MOV     CSHALC,VCSHNB   ;Set # blocks in use = # blocks allocated
024324  013702  000000G                 MOV     CSHALC,R2       ;Did used request data caching?
024330  001464                         BEQ     9$              ;Br if not
                                ;
                                ;  Calculate number of 64-byte blocks needed for each cache control table
                                ;
024332  062702  000037                 ADD     #31.,R2         ;Bound up to 32 word block
024336  072227  177773                 ASH     #-5.,R2         ;Convert to # 64-byte blocks
                                ;
                                ;  Compute total space that will be used by all cache data
                                ;
024342  013703  000000G                 MOV     CSHALC,R3       ;Get # blocks in cache
024346  072327  000003                 ASH     #3,R3           ;Get # 64-byte blks used by cache data buffers
024352  012700  000010                 MOV     #8.,R0          ;Get number of cache control tables
024356  060203                 1$:     ADD     R2,R3           ;Accumulate total space needed
024360  077002                         SOB     R0,1$
024362  010337  000000G                 MOV     R3,CSHSIZ       ;Save total space used by cache data
                                ;
                                ;  See if there is enough memory space available for the specified cache
                                ;
024366  010500                         MOV     R5,R0           ;Get top of memory address
024370  160400                         SUB     R4,R0           ;Compute # free 64-byte blocks
024372  020300                         CMP     R3,R0           ;Is there enough total space?
024374  103045                         BHIS    10$             ;Br if not
                                ;
                                ;  Allocate space for cache data buffers
                                ;
024376  013700  000000G                 MOV     CSHALC,R0       ;Get # blocks in data cache
024402  072027  000003                 ASH     #3,R0           ;Get # 64-byte blocks needed for allocation
024406  160005                         SUB     R0,R5           ;Allocate space for cache data buffers
024410  010537  000000G                 MOV     R5,CSHBFP       ;Save pointer to base of buffer area
                                ;
                                ;  Allocate space for each control table
                                ;
024414  160205                         SUB     R2,R5           ;Allocate space for table
024416  010537  000000G                 MOV     R5,CA$BLK       ;Block number associated with entry
024422  160205                         SUB     R2,R5           ;Allocate space for table
024424  010537  000000G                 MOV     R5,CA$DVU       ;Device and unit number
024430  160205                         SUB     R2,R5           ;Allocate space for table
024432  010537  000000G                 MOV     R5,CA$WCT       ;Number of words
024436  160205                         SUB     R2,R5           ;Allocate space for table
```

```
     58 024440  010537  000000G                 MOV     R5,CA$UFL       ;LRU chain forward link
     59 024444  160205                          SUB     R2,R5           ;Allocate space for table
     60 024446  010537  000000G                 MOV     R5,CA$UBL       ;LRU chain backward link
     61 024452  160205                          SUB     R2,R5           ;Allocate space for table
     62 024454  010537  000000G                 MOV     R5,CA$HFL       ;Hash chain forward link
     63 024460  160205                          SUB     R2,R5           ;Allocate space for table
     64 024462  010537  000000G                 MOV     R5,CA$HBL       ;Hash chain backward link
     65 024466  160205                          SUB     R2,R5           ;Allocate space for table
     66 024470  010537  000000G                 MOV     R5,CA$HSH       ;Hash chain list heads
     67 024474  020527  001600                  CMP     R5,#1600        ;Did we run over RT-11?
     68 024500  101403                          BLOS    10$             ;Br if yes
     69                                 ;
     70                                 ;  Finished
     71                                 ;
     72 024502  012603                  9$:     MOV     (SP)+,R3
     73 024504  012602                          MOV     (SP)+,R2
     74 024506  000207                          RETURN
     75                                 ;
     76                                 ;  Insufficent memory space available for cache data
     77                                 ;
     78 024510                          10$:    .PRINT  #TSXHD          ;Print heading
     79 024516                                  .PRINT  #CSHOVF         ;Overlow message
     80 024524  000137  004250'                 JMP     INISTP          ;Abort the initialization
```

```
                        .IF     EQ,PROCID       ;Don't allow ODT for production PRO version
                        .SBTTL  GETODT -- Load ODT
                ;-----------------------------------------------------------------------
                ;  GETODT is called to load ODT into memory above TSX and transfer control
                ;  to it.  On return, ODT has been started.
                ;
                ;  Inputs:
                ;   R5 = Address where ODT is to be loaded.
                ;
                ;  Outputs:
                ;   R5 = Address above top of ODT.
                ;
                GETODT: MOV     R1,-(SP)
                        MOV     R2,-(SP)
                        MOV     R3,-(SP)
                        MOV     R4,-(SP)
                ;
                ;  Try to lookup ODT rel file.
                ;
                        .LOOKUP #AREA,#1,#ODTBLK ;LOOKUP ODT REL FILE
                        BCC     1$              ;BR IF FOUND IT
                        .PRINT  #TSXHD          ;CAN'T FIND ODT
                        .PRINT  #NOODT
                        JMP     INISTP          ;ABORT INITIALIZATION
                ;
                ;  Read first block of ODT file and determine size of ODT.
                ;
                1$:     ADD     #200.,R5        ;RESERVE SPACE FOR ODT STACK
                        MOV     R5,R0           ;CHECK MEMORY ADDRESS FOR OVERFLOW
                        ADD     #512.,R0
                        CALL    CHKMEM
                        .READW  #AREA,#1,R5,#256.,#0
                        BCC     2$              ;Br if no read error
                        JMP     ODTRDX          ;Read error
                2$:     MOV     RSZ(R5),R2      ;GET SIZE OF ODT
                        MOV     R2,R3
                        ADD     R5,R3           ;GET ADDRESS ABOVE TOP OF ODT
                        MOV     R3,R0           ;CHECK MEMORY ADDRESS FOR OVERFLOW
                        CALL    CHKMEM
                        CLC
                        ROR     R2              ;GET # WORDS IN ODT
                ; Get starting address of ODT
                        MOV     STA(R5),R0      ;GET OFFSET TO START ADDRESS
                        SUB     #ODTBAS,R0      ;CALCULATE ABSOLUTE STARTING ADDRESS
                        ADD     R5,R0
                        MOV     R0,ODTSTA       ;THIS IS REAL STARTING ADDRESS
                        MOV     RBD(R5),R1      ;GET # OF BLOCK WITH RELOCATION INFO
                ;
                ;  Read in ODT rel file image.
                ;
                        .READW  #AREA,#1,R5,R2,#1
                        BCS     ODTRDX          ;BR IF READ ERROR
                ;
                ;  Relocate addresses in ODT.
                ;  R5 = Address of base of ODT;  R3 = Address above top of ODT.
                ;  R1 = Block number in rel file of start of relocation info.
                ;
```

```
                        RELFIL: MOV     R3,ODTTOP       ;SAVE ADDRESS ABOVE TOP OF ODT
                                MOV     R3,RLBF
                        .IF     NE,PROCID               ;Only if PRO protection code is included
                                TSTB    PROFLG          ;Are we running on a Pro?
                                BNE     1$              ;Br if yes
                        .ENDC   ;NE,PROCID
                                MOV     WRKBUF,RLBF     ;READ RELOCATION INFO HERE
                        1$:     MOV     RLBF,RLBFND
                                ADD     #1024.,RLBFND   ;GET ADDRESS OF END OF BUFFER AREA
                                MOV     R5,R4           ;GET BASE ADDRESS OF ODT
                                SUB     #ODTBAS,R4      ;SUBTRACT LINK BASE ADDRESS
                        ;  Read in relocation address list.
                        4$:     .READW  #AREA,#1,RLBF,#512.,R1
                                BCC     7$              ;BR IF NO READ ERROR
                                TSTB    @#52            ;END OF FILE IS OK
                                BNE     ODTRDX          ;BR IF READ ERROR
                        ;  Relocate some addresses in ODT.
                        7$:     MOV     RLBF,R2         ;POINT TO RELOCATION INFO
                        3$:     MOV     (R2)+,R3        ;GET ADDRESS OF LOCATION TO RELOCATE
                                CMP     R3,#-2          ;TIME TO STOP?
                                BEQ     9$              ;BR IF FINISHED
                                MOV     (R2)+,R0        ;GET VALUE TO RELOCATE
                                ASL     R3              ;CVT TO BYTE ADDRESS
                                BCC     5$              ;BR IF ADDITIVE RELOCATION
                                SUB     R4,R0           ;RELOCATE THE ADDRESS
                                BR      6$
                        5$:     ADD     R4,R0           ;RELOCATE THE ADDRESS
                        6$:     ADD     R5,R3           ;GET LOCATION WHERE WORD GOES
                                MOV     R0,@R3          ;STORE RELOCATED ADDRESS
                                CMP     R2,RLBFND       ;TIME TO READ NEXT BUFFER FULL?
                                BLO     3$              ;BR IF NOT
                                ADD     #2,R1           ;ADVANCE BLOCK #
                                BR      4$              ;GO READ NEXT BUFFER FULL
                        ;
                        ;  Finished relocation.
                        ;  Close ODT rel file.
                        ;
                        9$:     .CLOSE  #1
                        ;
                        ;  Direct interrupts to 60 and 64 to an RTI instruction
                        ;
                        ;       MOV     #DORTI,@#60     ;Catch interrupt 60
                        ;       MOV     #DORTI,@#64     ;Catch interrupt 64
                        ;
                        ;  Load registers with the following values for initial entry to ODT:
                        ;   R0 = Base of TSINIT
                        ;   R1 = Important breakpoint (^R) in TSX
                        ;   R2 = Base of TSGEN
                        ;   R3 = Base of TSEXEC
                        ;   R4 = Base of TSEMT
                        ;   R5 = Return address to start execution
                        ;   0(SP) = Address of mapsys routine
                        ;   2(SP) = Address of  sysmap cell
                        ;
                            MOV     #TSINIT,R0
                            MOV     #BRKPT,R1
                            MOV     #TSGEN,R2
```

```
                        MOV     #TSEXEC,R3
                        MOV     #TSEMT,R4
                        MOV     #SYSMAP,-(SP)   ;PASS ADDRESS OF SYSMAP CELL TO ODT
                        MOV     #MAPSYS,-(SP)   ;PASS ADDRESS OF MAPSYS ROUTINE
                        MOV     #10$,R5         ;ADDRESS FOR ODT TO RETURN TO
                ;
                ;  Enter ODT
                ;
                        JMP     @ODTSTA         ;JUMP TO START OF ODT
                ;
                ;  Return from ODT.
                ;  Continue initialization of TSX.
                ;
10$:            MOV     @#14,ODTTRP     ;SAVE ODT BREAKPOINT ENTRY ADDRESS
                        MOV     ODTTOP,R5       ;ADDRESS ABOVE TOP OF ODT
                        MOV     (SP)+,R4
                        MOV     (SP)+,R3
                        MOV     (SP)+,R2
                        MOV     (SP)+,R1
                        RETURN
                ;
                ;  Error while reading ODT rel file.
                ;
ODTRDX:         .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
                        .PRINT  #ODTRDM
                        JMP     INISTP          ;ABORT INITIALIZATION
                ;
                ;  RTI instruction to disable interrupts
                ;
DORTI:          RTI
                        .ENDC   ;EQ,PROCID
```

```
                                        .SBTTL  OPNCHN -- Open a TSX-Plus channel
                                ;-----------------------------------------------------------------------
                                ;  OPNCHN is called to set up information in a TSX-Plus channel block
                                ;  to make it look as if the channel has been opened to a specified
                                ;  device with a .ENTER.
                                ;
                                ;  Inputs:
                                ;   R0 = Address of channel block to be opened.
                                ;   R2 = Rad50 device name.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Cannot open the device.
                                ;
024530  010146                 OPNCHN: MOV     R1,-(SP)
024532  010346                         MOV     R3,-(SP)
024534  010003                         MOV     R0,R3           ;Carry channel block address in R3
                                ;
                                ;  Initialize the channel block
                                ;
024536  010301                         MOV     R3,R1           ;Point to the channel block
024540  012700  000000C                MOV     #<CHNSIZ/2>,R0  ;Get # words to zero
024544  005021                 2$:     CLR     (R1)+           ;Zero the channel block
024546  077002                         SOB     R0,2$
024550  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 #
                                ;
024556  010200                         MOV     R2,R0           ;Get the full device name
024560  004737  011476'                CALL    CVTDVU          ;Convert to dev # and unit #
024564  103411                         BCS     9$              ;Br if we don't recognize the device name
024566  010001                         MOV     R0,R1           ;Get index # and unit #
024570  000301                         SWAB    R1              ;Get unit # to low byte
024572  110163  000000G                MOVB    R1,C.DEVQ(R3)   ;Set unit # in channel block
024576  042700  000000C                BIC     #^C<CS$NMX>,R0  ;Clear all but device index number in R0
024602  050063  000000G                BIS     R0,C.CSW(R3)    ;Store device index # into CSW
                                ;
                                ;  Success
                                ;
024606  000241                         CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
024610  012603                 9$:     MOV     (SP)+,R3
024612  012601                         MOV     (SP)+,R1
024614  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.
                                ;
024616  010146                 SETCHN: MOV     R1,-(SP)
024620  010246                         MOV     R2,-(SP)
024622  010346                         MOV     R3,-(SP)
024624  010001                         MOV     R0,R1           ;GET ADDRESS OF TSX CHANNEL BLOCK
                                ;
                                ;  Do .SAVESTATUS to store channel information into TSX channel block.
                                ;
024626                                  .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.
                                ;
024644  011103                          MOV     (R1),R3         ;GET CSW FOR CHANNEL
024646  042703  177701                  BIC     #^C76,R3        ;GET RT-11 DEVICE INDEX NUMBER
024652                                  .GVAL   #AREA,#404      ;GET RT-11 OFFSET TO PNAME TABLE
024672  060003                          ADD     R0,R3           ;GET ADDRESS OF NAME OF DEVICE IN PNAME TABLE
024674                                  .GVAL   #AREA,R3        ;GET NAME OF DEVICE FROM RT-11
024712  013703  000000G                 MOV     NUMDEV,R3       ;GET INDEX # FOR LAST TSX DEVICE
024716  020063  000000G         1$:     CMP     R0,PNAME(R3)    ;LOOK FOR DEVICE IN OUR TABLES
024722  001404                          BEQ     2$              ;BR IF FOUND
024724  162703  000002                  SUB     #2,R3           ;CHECK NEXT ENTRY
024730  002372                          BGE     1$              ;BR IF MORE TO CHECK
024732  000407                          BR      MTSXDV          ;VERY STRANGE THAT WE DIDN'T FIND IT
024734  042711  000076          2$:     BIC     #76,(R1)        ;CLEAR OUT RT-11 DEVICE #
024740  050311                          BIS     R3,(R1)         ;STORE TSX DEVICE #
                                ;
                                ;  Finished
                                ;
024742  012603                          MOV     (SP)+,R3
024744  012602                          MOV     (SP)+,R2
024746  012601                          MOV     (SP)+,R1
024750  000207                          RETURN
                                ;
                                ;  Error: Could not locate Rt-11 device number in TSX device table.
                                ;
024752                          MTSXDV: .PRINT  #TSXHD          ;PRINT ERROR MESSAGE
024760                                  .PRINT  #REQMIS         ;Missing a required device
024766  010200                          MOV     R2,R0           ;GET RAD50 DEVICE NAME
024770  004737  025502'                 CALL    PRTR50          ;DISPLAY DEVICE NAME
024774                                  .PRINT  #CRLF
```

```
     58 025002  000137  004250'                    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
                ;
025006  010146  SETSY:  MOV     R1,-(SP)
025010  010246          MOV     R2,-(SP)
                ;
                ;  Set up system device unit number
                ;
025012                  .GVAL   #AREA,#274      ;Get system unit # from RT-11 (high byte)
025032  010037  0000000 MOV     R0,SYUNIT       ;Set system unit number
                ;
                ;  Set up system device index number
                ;
025036                  .GVAL   #AREA,#364      ;Get RT-11 system device index number
025056  010002          MOV     R0,R2           ;Save device index number
025060                  .GVAL   #AREA,#404      ;Get offset within RMON of PNAME table
025100  060002          ADD     R0,R2           ;Get offset to name of SY device
025102                  .GVAL   #AREA,R2        ;Get name of RT-11 system device
025120  013701  0000000 MOV     NUMDEV,R1       ;Get index to last TSX-Plus device entry
025124  020061  0000000 1$:    CMP     R0,PNAME(R1)    ;Search for device in TSX tables
025130  001405          BEQ     2$              ;Br if found it
025132  162701  000002  SUB     #2,R1           ;Keep looking if more
025136  002372          BGE     1$
025140  010002          MOV     R0,R2           ;Save name of system device
025142  000703          BR      MTSXDV          ;Missing device error
025144  010137  0000000 2$:    MOV     R1,SYINDX       ;Store index # of TSX-Plus system device
                ;
                ;  Set up RAD50 name of SY disk
                ;
025150  113702  0000010 MOVB    SYUNIT+1,R2     ;GET SYSTEM UNIT NUMBER
025154  062702  000036  ADD     #36,R2          ;PUT IN "0" AS 3'RD CHARACTER OF NAME
025160  066102  0000000 ADD     PNAME(R1),R2    ;ADD DEVICE NAME
025164  010237  0000000 MOV     R2,SYNAME       ;THIS IS THE FULL SY DISK NAME
                ;
                ;  Finished
                ;
025170  012602          MOV     (SP)+,R2
025172  012601          MOV     (SP)+,R1
025174  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.
                                ;
025176  010046                  RTFTCH: MOV     R0,-(SP)
025200  010246                          MOV     R2,-(SP)
025202  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set the name of the device being fetched
                                ;
025204  010037  000130'                 MOV     R0,FETDEV       ;Set name of device whose handler to fetch
                                ;
                                ;  Do a .DSTAT to get information about the handler
                                ;
025210                                  .DSTAT  #DSTBLK,#FETDEV ;Get information about the device handler
025222  103425                          BCS     9$              ;Br if device not recognized
                                ;
                                ;  Determine if the handler is currently resident
                                ;
025224  005737  000116'                 TST     DSTBLK+4        ;Is the handler resident now?
025230  001021                          BNE     8$              ;Br if yes
                                ;
                                ;  The handler is not currently resident.
                                ;  See if it will fit in WRKBUF.
                                ;
025232  013702  000152'                 MOV     WRKBUF,R2       ;Set address where handler will be loaded
025236  013700  000114'                 MOV     DSTBLK+2,R0     ;Get the size of the handler
025242  020037  000154'                 CMP     R0,WRKSIZ       ;Will handler fit in WRKBUF?
025246  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.
                                ;
025250  060500                          ADD     R5,R0           ;Get address above top of area needed
025252  020037  000132'                 CMP     R0,TOPMEM       ;Is there room for handler?
025256  101013                          BHI     10$             ;Br if not
025260  010502                          MOV     R5,R2           ;Set address where handler is to be loaded
                                ;
                                ;  Fetch the handler
                                ;
025262                          1$:     .FETCH  R2,#FETDEV      ;Try to fetch the handler
025272  103401                          BCS     9$              ;Br if error on fetch
                                ;
                                ;  We successfully fetched the handler
                                ;
```

```
     58 025274  000241                  8$:     CLC                     ;Signal success on return
     59                                  ;
     60                                  ;  Finished
     61                                  ;
     62 025276  012605                  9$:     MOV     (SP)+,R5
     63 025300  012602                          MOV     (SP)+,R2
     64 025302  012600                          MOV     (SP)+,R0
     65 025304  000207                          RETURN
     66                                  ;
     67                                  ;  Insufficient memory available to load the handler
     68                                  ;
     69 025306  004737  025332'          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.
                                ;
025312  020037  000302'       CHKMEM: CMP     R0,MEMLIM       ;IS THE ADDRESS OK?
025316  103402                        BLO     1$              ;BR IF OK
025320  004737  025332'               CALL    SIZERR          ;Generated system is too big -- abort
025324  004737  025352'       1$:     CALL    CCATST          ;CHECK FOR ^C ABORT REQUEST
025330  000207                        RETURN

                                ;-----------------------------------------------------------------------
                                ;  Generated system is too big.  Abort the initialization.
                                ;
025332                          SIZERR: .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
025340                                  .PRINT  #TOOBIG         ;PRINT ERROR MESSAGE
025346  000137  004250'                JMP     INISTP          ;ABORT INITIALIZATION

                                ;-----------------------------------------------------------------------
                                ;  Check for control-C and abort initialization if requested.
                                ;
025352  005737  000040'       CCATST: TST     CCAFLG          ;DID USER REQUEST ^C ABORT?
025356  001402                        BEQ     1$              ;BRANCH IF NOT
025360  000137  004250'               JMP     INISTP          ;ELSE ABORT INITIALIZATION
025364  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.
                                ;
025366  010146                 PRTOCT: MOV     R1,-(SP)
025370  010246                         MOV     R2,-(SP)
025372  010001                         MOV     R0,R1           ;GET VALUE TO PRINT
025374  012702  000006                 MOV     #6,R2           ;PRINT 6 DIGITS
025400  005000                         CLR     R0
025402  073027  000001                 ASHC    #1,R0           ;GET 1ST OCTAL DIGIT (1 BIT)
025406  000403                         BR      2$
025410  005000                 1$:     CLR     R0              ;INITIALIZE FOR SHIFT
025412  073027  000003                 ASHC    #3,R0           ;SHIFT AN OCTAL DIGIT INTO R0
025416  062700  000060         2$:     ADD     #'0,R0          ;CONVERT TO ASCII CHARACTER
025422                                 .TTYOUT                 ;PRINT THE CHARACTER
025426  077210                         SOB     R2,1$           ;LOOP AND PRINT MORE DIGITS
025430  012602                         MOV     (SP)+,R2
025432  012601                         MOV     (SP)+,R1
025434  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
                                ;
025436  010146                  PRTDEC: MOV     R1,-(SP)
025440  005046                          CLR     -(SP)           ;NULL ON STACK TO STOP US
                                ;
                                ;  Convert value to ascii digit string and stack the digits.
                                ;
025442  010001                          MOV     R0,R1           ;GET VALUE TO BE CONVERTED
025444  005000                  1$:     CLR     R0              ;SET HIGH-ORDER PART OF VALUE TO 0
025446  071027  000012                  DIV     #10.,R0         ;DIVIDE R0-R1 BY 10.
025452  062701  000060                  ADD     #'0,R1          ;CONVERT REMAINDER TO ASCII DIGIT
025456  010146                          MOV     R1,-(SP)        ;AND STACK THE DIGIT
025460  010001                          MOV     R0,R1           ;GET QUOTIENT
025462  001370                          BNE     1$              ;BR IF MORE DIGITS TO CONVERT
                                ;
                                ;  Finished conversion.  Print result.
                                ;
025464  012600                  2$:     MOV     (SP)+,R0        ;GET A DIGIT FROM THE STACK
025466  001403                          BEQ     3$              ;BR IF REACHED END
025470                                  .TTYOUT                 ;PRINT THE DIGIT
025474  000773                          BR      2$              ;PRINT MORE
                                ;
                                ;  Finished
                                ;
025476  012601                  3$:     MOV     (SP)+,R1
025500  000207                          RETURN
```

```
                                        .SBTTL  PRTR50 -- Print Rad-50 value
                                ;-------------------------------------------------------------------------
                                ;  PRTR50 is called to print a Rad-50 value.
                                ;
                                ;  Inputs:
                                ;   R0 = value to be printed.
                                ;
025502  010146                  PRTR50: MOV     R1,-(SP)
025504  010246                          MOV     R2,-(SP)
                                ;
                                ;  Convert value to ascii string and stack the characters.
                                ;
025506  012702  000003                  MOV     #3,R2           ;GET # CHARS TO CVT
025512  010001                          MOV     R0,R1           ;GET VALUE TO BE CONVERTED
025514  005000                  1$:     CLR     R0              ;CLEAR HIGH-ORDER VALUE
025516  071027  000050                  DIV     #50,R0          ;DIVIDE R0-R1 BY 50
025522  116101  003464'                 MOVB    R50CHR(R1),R1   ;CONVERT REMAINDER TO ASCII CHARACTER
025526  010146                          MOV     R1,-(SP)        ;STACK THE CHARACTER
025530  010001                          MOV     R0,R1           ;GET QUOTIENT
025532  077210                          SOB     R2,1$           ;BR IF MORE CHARS TO CONVERT
                                ;
                                ;  Finished conversion.  Print the result.
                                ;
025534  012702  000003                  MOV     #3,R2           ;GET # CHARS TO PRINT
025540  012600                  2$:     MOV     (SP)+,R0        ;GET NEXT CHARACTER
025542                                  .TTYOUT                 ;PRINT THE CHARACTER
025546  077204                          SOB     R2,2$           ;LOOP IF MORE CHARS TO PRINT
                                ;
                                ;  Finished
                                ;
025550  012602                          MOV     (SP)+,R2
025552  012601                          MOV     (SP)+,R1
025554  000207                          RETURN
                                ;-------------------------------------------------------------------------
                                ;  Define top of TSINIT
                                ;
025556                          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
                                ;
025556  177740                 EDARGB: .WORD   -32.                     ;# OF BYTES TO BE DECRYPTED (EDMTH3)
025560  026066'                 EDADDR: .WORD   DSKBUF                  ;POINTER TO BUFFER TO BE DECRYPTED
                                ;
                                ;  Recover license number and decrypt disk image of Pro ID to intermed. state
                                ;
025562  010146                 INSCHK: MOV     R1,-(SP)                 ;SAVE REGISTERS
025564  010246                         MOV     R2,-(SP)
025566  010346                         MOV     R3,-(SP)
025570  010446                         MOV     R4,-(SP)
025572  010546                         MOV     R5,-(SP)
025574  012700                         MOV     (PC)+,R0                 ;DECRYPT LICENSE NUMBER
025576  073472                         .RAD50  /SCB/                    ;WITH THIS CODE
025600  074037  026126'                XOR     R0,LICNUM                ;BY XORING IT
025604  013737  026126' 0000000        MOV     LICNUM,TSXSIT            ;MOVE LICENCE NUMBER TO TSGEN CELL
025612  012700  025556'                MOV     #EDARGB,R0               ;POINT TO ENC/DEC ARG BLOCK (PRESET)
025616  004737  026254'                CALL    EDMTH3                   ;DECRYPT TO INTERMED STATE
                                ;
                                ;  Copy Pro ID ROM low bytes into memory, and encrypt 1 step
                                ;
        173600                  IDADDR  = 173600        ;ADDRESS OF START OF PRO 350 ID ROM
025622  012701  173600                 MOV     #IDADDR,R1               ;GET POINTER TO PRO ID ROM
025626  012702  026130'                MOV     #ROMBUF,R2               ;POINTER TO COPY OF HARDWARE ID
025632  010237  025560'                MOV     R2,EDADDR                ;SAVE ADDRESS FOR ENCRYPTION
025636  005437  025556'                NEG     EDARGB                   ;MAKE +32. FOR ENCRYPTION
025642  013700  025556'                MOV     EDARGB,R0                ;ALSO USE AS LOOP COUNTER
025646  112122                 3$:     MOVB    (R1)+,(R2)+              ;GET NEXT LOW BYTE
025650  005201                         INC     R1                       ;SKIP ID ROM HIGH BYTES
025652  077003                         SOB     R0,3$                    ;REPEAT THROUGH 32 BYTE ROM
025654  012700  025556'                MOV     #EDARGB,R0               ;POINT TO ENCRYPTION ARG BLOCK
025660  004737  026170'                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.
                                ;
025664  012701  026130'                MOV     #ROMBUF,R1               ;POINT TO HARDWARE COPY OF ID
025670  012702  026066'                MOV     #DSKBUF,R2               ;POINT TO DISK COPY OF ID
025674  012704  0000000               MOV     #SCHED,R4                ;POINT TO CODE TO BE CORRECTED
025700  013703  025556'                MOV     EDARGB,R3                ;INIT LOOP COUNTER
025704  004737  026060'                CALL    GETLIC                   ;USE LIC # AS SEED FOR EDPRNW, IN R0
025710  122122                 4$:     CMPB    (R1)+,(R2)+              ;VERIFY ID'S ARE THE SAME
025712  001014                         BNE     5$                       ;ABORT IF NO MATCH ON ANY BYTE
025714  004737  026456'                CALL    EDPRNW                   ;RANDOMIZE R0 FOR XOR (LIC# INIT SEED)
025720  011405                         MOV     @R4,R5                   ;GET ENCRYPTED CODE
025722  074005                         XOR     R0,R5                    ;RESTORE FUNCTIONAL CODE
025724  010524                         MOV     R5,(R4)+                 ;PUT DECRYPTED CODE BACK IN MEMORY
```

```
     58 025726  077310                          SOB     R3,4$                   ;REPEAT THROUGH ID TESTS
     59 025730  012605                          MOV     (SP)+,R5                ;RESTORE REGISTERS
     60 025732  012604                          MOV     (SP)+,R4
     61 025734  012603                          MOV     (SP)+,R3
     62 025736  012602                          MOV     (SP)+,R2
     63 025740  012601                          MOV     (SP)+,R1
     64 025742  000207                          RETURN                          ;ID CHECKS AND CODE DECRYPTED
     65                                 ;
     66 025744                          5$:     .PRINT  #TSXHD                  ;?TSX-F
     67 025752                                  .PRINT  #NOTLIC                 ;NOT LICENSED FOR THIS MACHINE
     68 025760  000137  004250'                 JMP     INISTP                  ;ID'S DON'T MATCH, ABORT INIT.
     69                                 ;
     70                                         .NLIST  BEX
     71 025764     124     150     151 NOTLIC: .ASCIZ  /This copy of TSX-Plus not licensed for use on this machine./
     72                                         .LIST   BEX
     73                                         .EVEN
     74                                 ;
     75                                 ;  Subroutine to recover incremental license number.  Assume it has been
     76                                 ;  decrypted already by XORing with .RAD50 /SCB/.
     77                                 ;
     78 026060  013700  026126'         GETLIC: MOV     LICNUM,R0               ;RETRIEVE DECRYPTED LIC # INTO R0
     79 026064  000207                          RETURN
     80                                 ;
     81                                 ;  Reserve room for both disk and hardware copies of the Pro ID number
     82                                 ;  and for the incremental license number
     83                                 ;
     84 026066                          DSKBUF: .BLKB   32.                     ;DISK IMAGE OF PRO ID
     85 026126  000000                  LICNUM: .WORD   0                       ;INCREMENTAL LICENSE NUMBER
     86 026130                          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
                                ;
026170                          EDMTH2:
026170  010146                         MOV     R1,-(SP)        ;Save registers
026172  010246                         MOV     R2,-(SP)
026174  010346                         MOV     R3,-(SP)
                                ;
                                ;  Fetch byte count, buffer pointer and initialize PRN seed
                                ;
026176  012003                         MOV     (R0)+,R3        ;Fetch byte count of input buffer
026200  011001                         MOV     (R0),R1         ;Fetch pointer to input buffer
026202  004737  026060'                CALL    GETLIC          ;Use license number as initial PRN seed
026206  010002                         MOV     R0,R2           ;Copy license number to form repeat count
026210  042702  177770                 BIC     #^C7,R2         ;No more than 8 repeats
026214  005202                         INC     R2              ;Make sure there is at least one
026216  004737  026456'         2$:     CALL    EDPRNW          ;Get a new PRN
026222  077203                         SOB     R2,2$           ;Advance the seed between 1 and 8 times
                                ;
                                ;  Now sweep the buffer, XORing each byte with the high PRN byte
                                ;
026224  004737  026456'         1$:     CALL    EDPRNW          ;With seed in R0, get next random number
026230  111102                         MOVB    (R1),R2         ;Get next input byte
```

```
      58 026232  000300                          SWAB    R0              ;Reverse PRN high and low bytes
      59 026234  074002                          XOR     R0,R2           ;Encrypt the byte
      60 026236  000300                          SWAB    R0              ;Restore PRN high and low bytes for next seed
      61 026240  110221                          MOVB    R2,(R1)+        ;Save encrypted bytes back into input buffer
      62 026242  077310                          SOB     R3,1$           ;Repeat for entire input buffer
      63                                 ;
      64 026244  012603                          MOV     (SP)+,R3        ;Restore registers
      65 026246  012602                          MOV     (SP)+,R2
      66 026250  012601                          MOV     (SP)+,R1
      67 026252  000207                          RETURN
```

```
                                        .SBTTL  EDMTH3 -- Encryption/decryption meth 3 (swap bytes&shift bits)
                                ;
                                ;  Using a prn of repeat length same as input string length, select prn
                                ;  numbered bytes from the input string, combine them into a word,
                                ;  shift the combined bytes a random number of bits, recombine the shifted
                                ;  bits and set the confused bytes back into the prn selected string bytes.
                                ;  Sign of the byte count indicates: + = encryption; - = decryption.
                                ;  If the byte count is 0 or 1, no encryption occurs. If the byte count is
                                ;  odd, then one random selected byte will not be encrypted.
                                ;
                                ;  Inputs:
                                ;       R0      Points to an arg block of the form:
                                ;               R0 --->  buff_siz       ;Word holding byte length of buffer.
                                ;                                       ;Note that buffer must be 512 or less
                                ;                                       ;in length. Flag encryption by using
                                ;                                       ;positive byte count ( 2 to 512.).
                                ;                                       ;Flag decryption by using negative
                                ;                                       ;byte count (-2 to -512.).
                                ;                                       ;
                                ;                       buff_addr       ;Address of buffer to be encrypted
                                ;  Outputs:
                                ;       R0      Randomized
                                ;       Input buffer encrypted
                                ;
026254  010146                  EDMTH3: MOV     R1,-(SP)                ;Save registers
026256  010246                          MOV     R2,-(SP)
026260  010346                          MOV     R3,-(SP)
026262  010446                          MOV     R4,-(SP)
026264  010546                          MOV     R5,-(SP)
026266  012003                          MOV     (R0)+,R3                ;Save the string length
026270  011046                          MOV     @R0,-(SP)               ;And save the input buffer pointer
                                ;  Initialize prn generator of desired length
026272  010300                          MOV     R3,R0                   ;Recover string length
026274  002002                          BGE     1$                      ;Branch if encryption
026276  005400                          NEG     R0                      ;If decryption, get real repeat
026300  005203                          INC     R3                      ;If neg, correct for ASR round down
026302  004737  026506'          1$:     CALL    INPRNM                  ;Set up for desired repeat length
                                ;  Start encryption loop through string
026306  006203                          ASR     R3                      ;Repeat for 1/2 the string length
026310  004737  026060'                  CALL    GETLIC                  ;Get lic. num. for initial seed in R0
026314  004737  026532'                  CALL    EDPRNM                  ;Seed PRN generator (-adjacent pairs)
026320  005703                  2$:     TST     R3                      ;Less than 2 bytes left?
026322  001446                          BEQ     9$                      ;Quit if so (odd len -> 1 byte unch.)
026324  005004                          CLR     R4                      ;Clean out shifting registers
026326  005005                          CLR     R5
026330  011601                          MOV     @SP,R1                  ;Retrieve buffer pointer
026332  010102                          MOV     R1,R2                   ;And second copy
                                ;  Select first random byte
026334  004737  026532'                  CALL    EDPRNM                  ;Randomize in range 0 - <strlen-1>
026340  060001                          ADD     R0,R1                   ;Point to first random byte of pair
                                ;  Select second random byte
026342  004737  026532'                  CALL    EDPRNM                  ;Randomize again
026346  060002                          ADD     R0,R2                   ;Point to next random byte
                                ;  Use part of PRNM as semi-random shift amount
026350  010046                          MOV     R0,-(SP)                ;Save EDPRNM seed for later
026352  042700  177771                  BIC     #^C6,R0                 ;Get a semi-random shift amount
                                ;  Select encryption or decryption
```

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

```
                                        .SBTTL  EDPRNW -- Pseudo random number generator with MOD 2^16
                                ;
                                ;  Linear congruential pseudo-random number generator with maximum repeat
                                ;  length of 65536 (2^16).  cf.  Hull and Dobell and Knuth, vol 2.
                                ;
                                ;  Inputs:
                                ;       R0      Seed value
                                ;
                                ;  Outputs:
                                ;       R0      New PRN, should be used for next seed
                                ;
026456                          EDPRNW:
026456  010446                          MOV     R4,-(SP)        ;Save registers
026460  010546                          MOV     R5,-(SP)
026462  010004                          MOV     R0,R4           ;Get seed to be multiplied
026464  012700                          MOV     (PC)+,R0        ;Fetch multiplier
026466  104375                  EDPRNA: .WORD   104375          ;Multiplier, can be replaced
026470  070400                          MUL     R0,R4           ;Multiply by A
026472  062705                          ADD     (PC)+,R5        ;Add C
026474  012705                  EDPRNC: .WORD   012705          ;Additive, can be replaced
026476  010500                          MOV     R5,R0           ;Return result mod 65536. as PRN
026500  012605                          MOV     (SP)+,R5        ;Restore registers
026502  012604                          MOV     (SP)+,R4
026504  000207                          RETURN
```

```
                                        .SBTTL  INPRNM -- Initialize PRN generator with repeat range M
                                ;
                                ;  Using the Hull and Dobell rules, determine acceptable values for
                                ;  A and C to get a repeat range of M.
                                ;
                                ;  Outputs:
                                ;       EDMULA  Set with first acceptable multiplier
                                ;       EDADDC  Set with first acceptable additive factor
                                ;       EDMODM  Set with desired repeat length
                                ;
026506                          INPRNM:
026506  012737  000040  026574' MOV     #32.,EDMODM     ;Get repeat length to cover Pro ID
026514  012737  000005  026562' MOV     #5,EDMULA       ;Use first valid A
026522  012737  000003  026566' MOV     #3,EDADDC       ;And first valid C
026530  000207                  RETURN
```

```
                                        .SBTTL  EDPRNM -- Generate pseudo-random number in specified range M
                                ;
                                ;  ************************************************************************
                                ;  * INPRNM MUST BE CALLED BEFORE FIRST SEED IS PASSED TO THIS ROUTINE!!!! *
                                ;  ************************************************************************
                                ;
                                ;  Using linear congruential method (cf. Hull and Dobell), generate
                                ;  pseudo-random number using seed passed in R0.  Return new PRN in R0.
                                ;
                                ;  Inputs:
                                ;       R0      Seed value, must be in range 0 to M (EDMODM)
                                ;
                                ;  Outputs:
                                ;       R0      New pseudo-random number, should be used for next seed
                                ;
026532                          EDPRNM:
026532  010446                          MOV     R4,-(SP)        ;Save R4 and R5
026534  010546                          MOV     R5,-(SP)
026536  020037  026574'                 CMP     R0,EDMODM       ;Is seed in range 0 to EDMODM?
026542  103405                          BLO     1$              ;Branch and proceed if so
026544  010005                          MOV     R0,R5           ;Set up to divide it by EDMODM
026546  005004                          CLR     R4              ;Set up for divide
026550  071437  026574'                 DIV     EDMODM,R4       ;Divide it
026554  010500                          MOV     R5,R0           ;And use remainder as seed
026556  010004                  1$:     MOV     R0,R4           ;Get current seed ready to be multiplied
026560  070427                          MUL     (PC)+,R4        ;Multiply by chosen A
026562  061125                  EDMULA: .WORD   25173.          ;Replace at run-time with 5
026564  062705                          ADD     (PC)+,R5        ;Add in C
026566  033031                  EDADDC: .WORD   13849.          ;Replace at run-time with 3
026570  005004                          CLR     R4              ;Clear high word for division
026572  071427                          DIV     (PC)+,R4        ;Perform mod M
026574  000400                  EDMODM: .WORD   256.            ;Replace at run-time with 32.
026576  010500                          MOV     R5,R0           ;Return remainder
026600  012605                          MOV     (SP)+,R5        ;Restore R4 and R5
026602  012604                          MOV     (SP)+,R4
026604  000207                          RETURN
                                ;
                                .IFF    ;NE,PROCID              ;Assemble if protection code not included
                                DSKBUF:                         ;Define dummy DSKBUF global symbol
                                .ENDC   ;NE,PROCID
                                ;
                                ;  Address of real top of TSINIT, including PRO init code
                                ;
                                PROITP:
                                        .CSECT  TSXEND
                                        .END
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:  00:02:08.36
```

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

Cross reference table (CREF V05.04)

```
$8BIT     2-129     40-72
$DEAD     2-122     29-27     38-20
$FORM     2-125     40-67
$HARD     2-154     22-26     23-30
$MEMSZ    2-141     45-102*
$NOIN     2-112     22-37
$OVRH     2-136     67-25
$PHONE    2-109     22-35
$SXON     2-90      40-25
$TAB      2-125     40-66
...V1      7-29      7-29      7-45      7-46     24-19     24-23     24-31     24-32     24-33     24-36     24-54     24-55
          24-61     24-63     24-67     24-78     24-176    25-221    25-226    25-227    25-243    25-243    29-50     29-50
          29-50     29-57     29-57     29-58     29-58     29-58     29-62     29-62     29-62     29-70     29-70     29-72
          29-72     29-84     29-95     29-96     29-102    29-103    29-108    29-111    29-116    29-119    30-30     30-30
          30-30     30-42     30-42     30-43     30-43     30-43     30-47     30-47     30-47     30-55     30-55     30-56
          30-56     30-68     30-77     30-78     30-84     30-85     31-60     31-61     31-85     31-85     31-85     31-89
          31-89     31-90     31-90     31-90     31-94     31-94     31-94     31-99     31-99     31-101    31-101    31-110
          31-131    31-132    31-138    31-139    39-21     39-21     39-21     39-27     39-27     39-62     39-62     39-62
          39-75     39-76     39-81     39-82     42-22     42-22     42-22     42-29     42-29     42-31     42-31     42-68
          42-68     42-68     42-72     42-73     42-73     42-73     42-78     42-78     42-78     42-82     42-82     42-83
          42-83     42-84     42-84     42-84     42-91     42-101    42-102    42-108    42-109    43-43     43-43     43-43
          43-47     43-48     43-48     43-48     43-52     43-52     43-52     43-55     43-55     43-60     43-65     43-65
          43-66     43-76     43-77     43-83     43-84     45-151    45-152    47-76     47-77     51-76     51-76     53-25
          53-25     53-25     53-39     53-39     55-25     55-25     57-21     57-21     57-21     57-29     57-29     57-63
          57-63     57-111    57-111    59-6      59-7      59-10     60-36     60-36     60-50     60-50     61-35     61-35
          67-17     67-17     67-51     67-52     69-24     69-24     69-52     69-53     69-58     69-59     72-34     72-35
          73-30     73-30     73-73     73-74     74-25     74-25     74-25     74-54     74-54     74-81     74-81     74-93
          74-94     74-99     74-104    74-105    75-78     75-79     78-25     78-25     78-32     78-34     78-53     78-54
          78-57     79-19     79-24     79-26     79-28     80-27     80-53     81-20     81-21     82-18     83-26     84-26
          85-66     85-67
...V2      7-29      7-29      7-29#     7-29#     25-243    25-243    25-243#   25-243#   29-50     29-50     29-50#    29-50#
          29-57     29-57#    29-58     29-58     29-58#    29-58#    29-62     29-62     29-62#    29-62#    29-70     29-70
          29-70#    29-70#    29-72     29-72#    30-30     30-30     30-30#    30-30#    30-42     30-42#    30-43     30-43
          30-43#    30-43#    30-47     30-47     30-47#    30-47#    30-55     30-55     30-55#    30-55#    30-56     30-56#
          31-85     31-85     31-85#    31-85#    31-89     31-89#    31-90     31-90     31-90#    31-90#    31-94     31-94
          31-94#    31-94#    31-99     31-99     31-99#    31-99#    31-101    31-101#   39-21     39-21     39-21#    39-21#
          39-27     39-27     39-27#    39-27#    39-62     39-62     39-62#    39-62#    42-22     42-22     42-22#    42-22#
          42-29     42-29     42-29#    42-29#    42-31     42-31     42-31#    42-31#    42-68     42-68     42-68#    42-68#
          42-72     42-72#    42-73     42-73     42-73#    42-73#    42-78     42-78     42-78#    42-78#    42-82     42-82
          42-82#    42-82#    42-83     42-83#    42-84     42-84     42-84#    42-84#    43-43     43-43     43-43#    43-43#
          43-47     43-47#    43-48     43-48     43-48#    43-48#    43-52     43-52     43-52#    43-52#    43-55     43-55
          43-55#    43-55#    43-60     43-60     43-60#    43-60#    43-65     43-65#    51-76     51-76#    53-25     53-25
          53-25#    53-25#    53-39     53-39     53-39#    53-39#    55-25     55-25     55-25#    55-25#    57-21     57-21
          57-21#    57-21#    57-29     57-29     57-29#    57-29#    57-63     57-63     57-63#    57-63#    57-111    57-111#
          60-36     60-36     60-36#    60-36#    60-50     60-50     60-50#    60-50#    61-35     61-35     61-35#    61-35#
          67-17     67-17     67-17#    67-17#    69-24     69-24     69-24#    69-24#    73-30     73-30     73-30#    73-30#
          74-25     74-25     74-25#    74-25#    74-54     74-54     74-54#    74-54#    74-81     74-81#    78-25     78-25
          78-25#    78-25#
AHEND      2-106     50-21
ALBFX     25-161     38-12#
ALCHRB    24-142     27-12#
ALCOVL    67-34      69-15#
ALCSLO    24-103     37-12#
ALCWRK    24-138     26-12#
ALOCBF    24-102     36-14#
AREA       2-182#    7-29      24-19     24-23     24-61     24-63     24-78     24-176    25-221    25-243    29-50     29-58
```

Cross reference table (CREF V05.04)

```
        29-62   29-70   30-30   30-43   30-47   30-55   31-85   31-90   31-94   31-99   39-21   39-27
        39-62   42-22   42-29   42-31   42-68   42-73   42-78   42-82   42-84   43-43   43-48   43-52
        43-55   43-60   53-25   53-39   55-25   57-21   57-29   57-63   60-36   60-50   61-35   67-17
        69-24   73-30   74-25   74-54   78-25   78-32   78-34   79-19   79-24   79-26   79-28
AUTHAN   2-106  50-21   50-23
BADDEV  29-104  29-116# 30-86   31-140  42-110  43-85
BADLIN   5-6#
BADOPN   5-11#  29-96   29-103
BASMAP   2-100  47-52*
BDLMSG   5-8#
BDSPOP   5-14#  31-61
BDVMSG   5-7#
BOSF     5-15#  31-132  31-139
BRKPT    2-101
BYTES    5-32#  24-36
C.CSW    2-142  32-30*  77-24*  77-35*
C.DEVQ   2-142  32-31*  77-33*
C.SBLK   2-142  32-23*
C1DEVX   2-142  32-27   40-112*
CA$BLK   2-147  75-52*
CA$DVU   2-147  75-54*
CA$HBL   2-148  75-64*
CA$HFL   2-148  75-62*
CA$HSH   2-148  75-66*
CA$UBL   2-148  75-60*
CA$UFL   2-148  75-58*
CA$WCT   2-147  75-56*
CC$$SZ   2-149  36-74
CCAFLG   2-186# 24-23   81-27
CCATST  25-247  81-14   81-27#
CCBHD    2-149  36-73*
CCLNAM   2-193# 39-62   39-65
CCLSAV   2-127  39-64
CD$$SZ   2-86   36-64
CDX$DH   2-141  36-104
CDX$DL   2-150
CDX$DZ   2-149
CDX$VH   2-152  36-106
CFHMSG   5-18#  29-116  53-33   57-23
CHAIN    2-116  24-85
CHKCLD  31-40   33-13#
CHKMEM  25-149  36-140  55-20   81-11#
CHNSIZ   2-121  77-21
CKLIN   25-34#
CL$EPN   2-126  40-58*
CL$EPP   2-126  40-52*
CL$EPS   2-125  40-31*  40-51*
CL$LEN   2-156  40-79*
CL$LIX   2-155  23-19*  40-23
CL$OPT   2-126  40-75*
CL$ORA   2-124  40-43*
CL$ORB   2-124  40-36*
CL$ORE   2-124  40-46*
CL$ORG   2-124  40-38*
CL$ORP   2-124  40-37*
CL$ORS   2-126  40-44*
```

Cross reference table (CREF V05.04)

```
CL$STA    2-126     40-83*
CLDEVX    2-142     32-29     40-94*
CLEOFS    2-125     40-53     40-54
CLHEAD    2-142     40-99     40-117
CLINCP    2-154     23-21
CLINIT   25-73      40-11#
CLK100    2-187#     6-165    24-14*
CLKRTI    2-106      6-71
CLKVEC    2-157      6-71*     6-72*     6-165*    24-14     25-78*
CLORSZ    2-111     40-39
CLOTIR    2-154     23-20
CLSIZE    2-142     40-100
CLSTS     2-117     40-96     40-114
CLTOTL    2-112     25-71     32-18     40-18     40-101    40-108    70-149
CLVEND    4-68#     40-132
CLVERS    2-110     40-123    40-137*
CLVTBL    4-60#     40-128
CO$8BT    2-129     40-74
CO$DEF    2-125     40-62
CO$FF     2-125     40-71
CO$TAB    2-125     40-68
CONFG2    2-104     24-165*   24-166*   24-169*   25-39     25-79
CONFIG    2-103     24-163*   25-5
CONSPC    5-13#     29-108
COSRT     5-34#     74-94
CRLF      5-9#      29-111    29-119    59-10     74-99     78-57
CS$ENT    2-144     77-24
CS$NMX    2-88      77-34
CS$OPN    2-144     77-24
CSHALC    2-121     36-71     70-161    75-17     75-18     75-28     75-44
CSHBAS    2-104      8-39     10-37*
CSHBFP    2-147     75-47*
CSHBUF   25-206     75-12#
CSHDEV    2-86      36-63*
CSHDVN    2-86      36-67*
CSHOVF    5-39#     75-79
CSHSIZ    2-147     75-33*
CSHVEC    2-105      8-42
CURDEV    2-204#    60-54*    61-40     63-45
CURNAM    2-205#    52-25*    57-20*    59-8
CVTDVU   31-37      34-14#    77-29
CW$50H    2-103     25-5
CW$BTH    2-102     24-161
CW$ESP    2-112     24-166
CW$FB     2-102     24-162
CW$FGJ    2-102     24-162
CW$GDH    2-102     24-161
CW$LGS    2-102     24-161
CW$PRO    2-106     24-116    25-39     25-79
CW$QBS    2-114     24-169
CW$USR    2-103     24-162
CW$XM     2-103     24-162
CXTALC   25-93      46-12#
CXTBAS    2-111     46-24
CXTBUF    2-118     36-134*
CXTPAG    2-87      29-40     39-41     46-45*    48-31
```

Cross reference table (CREF V05.04)

```
CXTPDR    2-110     46-38*
CXTRMN    2-111     46-23*    46-24*
CXTSIZ    2-110     46-16
CXTWDS    2-110     46-30*    46-42
DATIMH    2-104     25-29*
DATIML    2-104     25-30*
DC$$SZ    2-132     38-64
DCCSIZ    2-85      70-92
DEVSIZ    2-93      14-20*    25-50*    25-65*    40-98*    40-116*   54-45*
DEVVEC    6-139     12-6#
DFJMEM    2-133     48-50*
DHBFSZ    2-108     36-110    36-112
DHINIT   22-14#
DHLPRM   22-10#
DHQINT    2-135
DI$CL     2-117
DI$DU     2-155      3-30
DI$LD     2-117     25-63
DI$MU     2-155      3-32
DI$PI     2-156     14-18
DI$TT     2-118     25-48
DI$XL     2-155      3-31
DM$CSR    2-151
DM$LSR    2-151
DMYDEV    2-95      31-34     31-57     50-25     74-20
DOHNLC   55-45      57-106    60-13#
DS$ABT    2-124
DS$DIR    2-138     25-63     54-50
DS$NRD    2-129     54-52
DS$SFN    2-139     14-18     25-63
DS$VSZ    2-138     25-63
DSKBUF    2-58      85-13     85-48     85-84#
DSTBLK    2-194#    80-27     80-32     80-39
DTLX      2-103     39-52*
DTYPE     2-128     50-27
DV$$SZ    4-6#      52-58
DV$FLG    4-5#      52-56
DV$NAM    4-4#      52-54
DVFLAG    2-138     14-19*    25-49*    25-64*    40-97*    40-115*   51-43     54-30*    54-54*
DVFLBS    4-13#     52-53
DVFLND    4-42#     52-59
DVSTAT    2-89      14-18*    25-48*    25-63*    40-96*    40-114*   54-46*    54-50     54-52
DW$$SZ    2-114     70-137
DX$DMA    2-138      4-16      4-18      4-19      4-20      4-21      4-22      4-23      4-26      4-28      4-31      4-32
          4-33       4-34      4-37      4-38      52-65
DX$EBA    2-92       4-39      25-64     52-67
DX$IBH    2-97       4-18      4-31      4-32      4-33      4-34      51-48
DX$MAP    2-145     51-52
DX$MPH    2-97       4-14      4-15      4-16      4-18      4-24      4-25      4-27      4-29      4-30      4-31      4-32
          4-33       4-35      4-36      4-38      4-40      4-41      51-46
DX$NCA    2-157      4-39      40-97     40-115    54-54
DX$NHM    2-97       4-17      4-19      4-23      4-26      4-34      4-39      51-44
DX$NMT    2-129     40-97     40-115    54-54
DX$NRD    2-130     40-97     40-115
DX$NST    2-137      4-23      4-34
DZINIT   22-12#
```

Cross reference table (CREF V05.04)

```
DZOINT    2-135
EDADDC   89-14*    90-29#
EDADDR   85-13#    85-34*
EDARGB   85-12#    85-26     85-35*    85-36     85-40     85-50
EDMODM   89-12*    90-19     90-23     90-32#
EDMTH2   85-41     86-38#
EDMTH3   85-27     87-25#
EDMULA   89-13*    90-27#
EDPRNA   88-17#
EDPRNC   88-20#
EDPRNM   87-41     87-49     87-52     90-16#
EDPRNW   85-54     86-51     86-56     88-12#
EMMAP     2-99      6-123    45-109    57-77     57-82     62-36     62-55     73-47     73-52     74-65     74-71
EMTENT    2-107     6-89
ENTVEC    8-35      8-43      8-51      9-11#
ERHMSG    5-19#    53-41     55-27     57-65
ERHNDV    5-20#    53-58
ERRLOG    2-134    58-22
EXCBUF    2-84     26-17
EXTLSI    2-245#   45-119    51-54
FC$$SZ    2-134    70-69
FC$LBN    2-115    38-61
FETDEV    2-198#   80-23*    80-27     80-53
FF$$SZ    2-115    38-56
FILBLK    2-203#   57-49*    57-63     57-90*    73-20*    73-30     73-57*
FMEMHI    2-200#   25-168    45-103*   45-114    47-65*    57-105*   57-107    64-27     64-34*    64-35
FMEMLO    2-201#   25-167*   47-37     47-62     64-28
FNDHRB     2-58    63-18#
FORCEO    31-26    35-8#
FORK       2-89    58-15
FPTRAP    2-117     6-95
FQ$$SZ     2-95    36-37
FREIOQ    2-117    36-18*
FREPGS    2-118    48-22*
FRKGEN     2-95    36-34     36-37
FRKINI     2-95    36-36*
FSTDL      2-94    38-22
FSTIOL    2-154    23-11     36-102    37-24
FW$$SZ    2-115    38-58
GENTOP    2-156    24-27
GETHNH   25-202    56-13#
GETHNL   25-97     50-14#
GETLIC   85-51     85-78#    86-47     87-40
GETMAP   25-172    68-12#
GETOVL   68-34     72-12#
GETSRT   25-180    25-195    74-13#
GTBYT     2-101    58-25
GTLIN     2-229#   24-51
H.CSR      2-95    53-79
H.DSTS     2-89    53-48     54-46
H.DVSZ     2-93    54-45
H.GEN      2-89    53-67     53-71     57-30
H.INS      2-95    53-88
H.SIZ      2-93    54-44
HANDSK     2-83    54-38*
HANENT     2-89    12-14     14-21*    25-51*    25-66*    40-99*    40-117*   55-33*    55-34*    57-34*    60-21     60-61
```

Cross reference table (CREF V05.04)

```
HANMAP  57-99    60-22    60-63    61-41    62-14#
HANNAM   2-230#  53-24*   53-25    57-19*   57-21
HANPAR   2-97    12-20    13-25    13-54    51-60*   51-71*   55-16*   56-23    57-47*   62-26    62-32
HANRCB   2-83    27-20*   63-23
HANRCO   2-83    27-21*   27-22*
HANSIZ   2-93    14-22*   40-102*  40-118*  51-64    54-44*   55-18    57-39    57-98
HANUMP  57-101   60-24    60-72    60-79    61-31    62-49#
HANXIT   2-97    13-59
HANXMR   2-58    64-22#
HF$LIN   2-150
HF$MC    2-151
HF$RIE   2-150
HF$TIE   2-150
HF$TSB   2-150
HGENFL   2-237#  53-71*   57-30*   58-17    58-20
HIMAP    2-94     6-133   47-56*   48-20
HLERR   51-67    53-34    53-42    53-59    53-70    55-28    57-24    57-66    59-5#
HMAP     2-197#  57-48*   57-73    57-89*
HN2BIG   5-27#   51-66
HSGER    5-22#   53-69
HV$$SZ   3-20#   53-60
HV$ID    3-17#   53-49
HV$UPD   3-19#   53-54
HV$VER   3-18#   53-51
HVEND    3-33#   53-61
HVTBL    3-29#   53-46
ICONFG   2-241#  45-119*  51-54
IDADDR  85-31#   85-32
II$$SZ   2-85    36-57
ILSW2    2-112   22-35    40-65
INDDBL   2-139   42-40*
INDDBS   2-139   42-45*   42-56
INDFIL   2-84    42-62    42-68    42-73    42-78    42-84    42-89    42-91
INDINI  25-118   42-12#
INDNAM   2-232#  42-21*   42-22    42-51
INDOPN   5-40#   42-102   42-109
INDSAV   2-139   42-17*   42-50
INDTSV   2-139   42-88
INICLK  25-5#
INIDEV  25-40    25-47#
INIJMP   2-107    6-161
INIOVL   6-13     8-9#
INISTP   6-165#  24-37    24-56    25-228   29-112   29-120   31-64    31-133   39-77    39-83    45-153   47-78
        59-11    67-53    69-54    69-60    72-36    73-75    74-100   74-106   75-80    78-58    81-22    81-29
        85-68
INITGO   2-57    24-8#    24-104   24-106
INITOP   2-57     2-207   24-132   24-134   66-52    66-54    84-37#
INMXV    2-108
INPRNM  87-37    89-11#
INRECV   2-108
INSCHK  24-96    85-17#
INSCK1  51-25    52-16#
INSCK2  51-31    53-14#
INSTBL   2-85    36-54*
INSTBN   2-85    36-59*
INTEN    2-89    58-16
```

Cross reference table (CREF V05.04)

```
INTMX1    2-109
INTSND    2-113     6-24*     6-152     24-113*
INTSSZ    2-84      24-122
INTSTK    2-113     6-155     24-123*
INVEC     2-87
IOMAP     2-99      2-137     6-126
IOPAGE    2-99      44-29
IOQSIZ    2-118     36-19
IOTIMR    2-134     58-19
JCXPGS    2-133     48-32*
JM$$SZ    2-111     36-44
JMPO      2-103     6-78
JSWLOC    2-116     24-85
KEYSEG    8-23      10-10#    72-22
KMNBAS    2-119     39-34
KMNCHN    2-98      39-56
KMNHI     2-96      39-35*
KMNNAM    2-192#    39-21     39-57
KMNPGS    2-98      39-42*    48-26
KMNSTK    2-98      39-44*
KMNSTR    2-98      39-46*
KMNTOP    2-96      39-32*
KPAR0     2-98      44-14
KPAR5     2-135     6-130*    12-9      12-22*    12-62*    13-52     13-55     57-72     57-73*    57-84*    61-25     61-46*
          62-32*    62-57*    73-42     73-43*    73-54*
KPAR6     2-157     60-71*    60-78*    61-26     61-27*    61-45*
KPAR7     2-100     44-29*
KPDR0     2-100     44-16
LCDTYP    2-109     36-104    36-106
LCLUNT    2-136     23-13*    23-18     23-26*    40-26*
LD1TBL    4-77#     41-15
LDDEVX    2-138     25-56*    25-61*
LDHAND    50-31     50-46     51-21#
LDHB1B    2-110     36-108*
LDHB1P    2-110     36-109*
LDHB2B    2-110     36-111*
LDHNHI    56-29     57-12#
LDHNLO    51-72     55-12#
LDINIT    25-67     41-8#
LDREAD    60-64     61-15#
LDVERS    2-114     41-9      41-21*
LHIRBA    2-90      37-30     37-39*
LHIRBB    2-90      37-43*
LHIRBC    2-91      37-54     37-60*    37-61     37-67*
LHIRBE    2-90      37-48*
LHIRBG    2-90      37-45*
LHIRBP    2-90      37-44*
LHIRBS    2-91      37-46*
LICNUM    85-24*    85-25     85-78     85-85#
LINBUF    2-94      38-28*
LINCHK    22-9#     25-34
LINEND    2-96      38-32*
LININI    6-113     22-20#
LINIR     2-154     23-21*    23-29*
LINSIZ    2-94      38-29
LINSPC    2-97      38-30*
```

Cross reference table (CREF V05.04)

```
LINTYP  22-25     23-9#
LMXLN    2-150
LMXNUM   2-87     40-29
LODINI   7-9#      8-24
LODOVL  72-26     73-9#
LOKBAS   2-105     8-47     10-33*
LOKCSH   2-132    38-71*
LOKMEM   2-100    38-55*
LOKVEC   2-105     8-50
LOMAP    2-93      6-131    47-50*   48-21
LOTBUF   2-96     38-38*
LOTEND   2-96     38-41*
LOTSIZ   2-96     38-39    38-40    40-42
LOUTIR   2-154    23-20*   23-28*
LOWEND   4-109#   66-39
LOWOVL   4-106#   66-36
LSTDL    2-94     38-24
LSTHL    2-154    36-114   37-17
LSTLIN   2-126    22-21
LSTMX    2-116    36-90
LSTPL    2-87     23-9     29-26    36-100   37-22
LSTSL    2-88     38-46    42-57    43-32
LSW10    2-90     40-25*
LSW3     2-112    22-26    22-37*   23-30*   29-27    38-20
LSW5     2-157
LSW6     2-122
LSWPBK   2-88
LTTPAR   2-105    38-26*
LXCL     2-84     23-27*
MA$SYS   2-112     6-135
MAPALC  25-212    47-27#
MAPPAR   2-97      6-130    47-64*
MAPSIZ   2-99     45-99    45-101
MAPSYS   2-136     6-104
MAXDEV   2-83     60-59
MAXOVL   4-98#     4-100
MAXSLO   2-91     37-33    37-35
MB$$SZ   2-144    70-99
MEM256   2-132     6-124    45-116*  45-125   57-75    57-80    61-21    61-48    62-34    62-53    73-45    73-50
        74-63     74-69
MEMINI  24-155    44-6#
MEMLIM   2-236#   24-72*   81-11
MEMPAR   2-130     6-93
MEMTST  25-89     45-24#
MF$CM    2-151
MF$CS    2-151
MF$LE    2-151
MF$LIN   2-149
MH$LPR   2-151
MH$PBR   2-152
MH$SCR   2-150
MHNSIZ   2-107    25-197*  57-45*
MI$$SZ   2-146
MI$LNK   2-145    38-86
MI$SBP   2-145    38-82*
MINCTR   2-119
```

Cross reference table (CREF V05.04)

```
MIOBHD    2-145     38-80
MIODBG    2-127
MIOFLG    2-145     38-78      70-143
MIONWB    2-146
MIOWHD    2-146
MMENBL    2-93       6-118     45-110     57-74      57-83      62-33      62-56      73-44      73-53      74-62      74-72
MONFQH    2-111     36-43*
MONVEC    2-119      6-17      27-22      53-92      62-22
MPARO     2-92
MPAR16    2-92
MPARFL    2-92
MPIVSZ   13-19#     25-105
MPPHY     2-101     58-26
MR$$SZ    2-144     70-102
MSGBAS    2-102     10-21*
MTSXDV   78-40      78-53#     79-35
MU$TXT    2-116     70-107
MUXVEC   22-13#
MVSIZ     2-130     46-25      46-49
MW$$SZ    2-146
MW$LNK    2-146
MXCSR     2-87
MXDTR     2-109
MXJADR    2-141     48-43*
MXJMEM    2-133     48-38*
MXLNT     2-117     36-92*
MXLPR     2-112
MXRBUF    2-109
MXTYPE    2-108
MXVEC     2-87
NDL       2-96      29-25
NDVRCB    2-83      27-16
NEDCHR    2-154     23-28
NEXMSG    5-25#
NFRESB    2-114     31-118*
NFSBLK    2-183#    43-60      43-61      43-62
NIOL      2-121     40-101
NLINES    2-120     38-58
NMSNMB    2-131     36-50
NMXHAN    2-196#    25-102     51-59*
NOCCL     5-17#     39-76
NOCLOK    5-23#     25-227
NOKMON    5-16#     39-82
NOODT     5-26#
NOSYDV    5-29#
NOTLIC   85-67      85-71#
NOXM     24-61      45-151#
NSCP      2-86      36-130
NSIP      2-85      36-56
NSL       2-96      29-25
NSPLBL    2-113     31-80      31-117     70-78
NSPLDV    2-99      31-22*     31-44*     31-50*     31-75      31-114
NSPLFL    2-113     36-78
NUMCCB    2-94      36-74
NUMCDB    2-85      38-54*
NUMDCD    2-132     38-53*
```

Cross reference table (CREF V05.04)

```
NUMDEV    2-88     12-57    14-12*   14-13    25-52*   25-59*   25-60    34-31    40-92*   40-93    40-110*  40-111
         52-43     54-24*   54-25    56-34    78-35    79-29
NUMFRK    2-95     36-34    36-37
NUMIOQ    2-117    36-19
NUMRDB    2-120     2-148   25-176
NXIVMH    2-84     25-104
NXMMSG    5-24#    45-152
O.ADR     2-135    10-15    66-35    69-30*   70-22
O.BLK     2-135     7-29    69-24    73-20
O.PAR     2-135     7-34*   73-25*
O.SIZ     2-135     7-21     7-23    69-34    70-26    73-19
ODTBAS    2-178#
ODTBLK    2-231#
ODTFLG    2-184#   24-89*
ODTRDM    5-28#
ODTSTA    2-235#
ODTTOP    2-185#
ODTTRP    2-123
OORCLO   70-60     70-149#
OORCSH   70-62     70-161#
OORDBG   70-61     70-155#
OORDMP   70-63     70-167#
OORLOK   70-53     70-88#
OORMIO   70-59     70-143#
OORMSG   70-54     70-97#
OORPLS   70-56     70-114#
OORSLE   70-57     70-129#
OORSPL   70-52     70-75#
OORSWP   70-55     70-123#
OORUSR   70-51     70-68#
OORWIN   70-58     70-135#
OOXFIN   70-175    70-190#
OOXNO    70-76     70-89    70-98    70-115   70-124   70-130   70-136   70-144   70-150   70-156   70-162   70-168
         70-173#
OOXYES   70-37     70-71    70-84    70-91    70-93    70-110   70-119   70-125   70-131   70-139   70-145   70-151
         70-157    70-163   70-169   70-179#
OPNCHN   31-48     77-14#
OPNKMN   25-114    39-17#
OPNRSF   25-140    30-11#
OPNSWP   25-136    29-13#
OPTOVL   68-21     69-42    70-15#
OS$$SZ    4-96#     4-100    8-26    66-46    67-36    68-23    68-35    71-31
OS$FLG    4-94#     8-21    66-42*   66-64*   67-33*   68-32    71-22
OS$OVL    4-95#     7-15    10-14    66-34    67-32*   69-19    70-21    73-18
OS$SIZ    4-93#     7-19     8-25    66-32    66-43    66-65    67-35*   68-22*   68-30    71-20    71-24    71-28
         71-28     72-16
OSEND     4-101#
OSLAST    4-102#    8-27    66-47    67-40*   68-24    68-36    71-32
OSTABL    4-100#    4-102    8-20    66-31    67-31    68-20    68-29    71-19
OSZ       2-175#
OTMXV     2-108
OTRECV    2-108
OVLBAS    2-202#    8-14    66-30*
OVLBLD   66-23     67-10#
OVLEND   70-35     70-64#
OVLLST   70-31     70-50#
```

Cross reference table (CREF V05.04)

```
OVLPOS  24-127    66-15#
OVLTRY  66-62     71-14#
OVRADD   2-136    67-27*
PARENL   2-92
PARSET  25-85     49-34#
PHSOVF   5-33#    47-77
PHYMEM   2-141     6-121    45-98*
PIDPTR   2-68     25-42*
PIDRIV   2-68     25-42
PIDVEN   6-147    14-7#
PIHAN    2-68     14-21
PIINSZ  24-112#   24-120
PISRT    2-57      2-249#   25-193
PMPAR    2-131    38-93*
PMSIZE   2-131    38-91
PNAME    2-89     14-17*   25-47*   25-62*   34-32    40-95*   40-113*  52-45    54-29*   57-19    57-20    78-36
        79-30     79-42
PPTERM   2-57      2-254#
PROASM   1-1#      2-20     2-32     2-57     2-63     6-29     6-143    11-1     14-1     15-1     25-38    25-187
        28-1      45-45    45-55    49-1     65-1
PROBUF   2-58      2-206#   24-119*
PROCID   2-24     24-95    24-146   76-1     85-1
PROFLG   2-98      6-30     6-53     6-144    24-118*  25-191   45-46
PROHAN   2-67      6-146
PROINI   2-67     25-41
PROITP   2-57     90-44#
PROLIN   2-67     22-31
PRONOP   2-67      6-32
PROSIZ   2-67     14-22
PRTDEC  24-35     29-110   83-9#
PRTOCT  82-8#
PRTR50  29-118    31-63    59-9     74-97    78-56    84-8#
PSW      2-94     57-71*   57-85*   62-18*   62-65*   73-41*   73-55*   74-59*   74-61*   74-73*
PTBYT    2-100    58-24
PTWRD    2-100    58-23
PVSPBL   2-114    31-115
QBUS     2-137    24-167   45-117
R50C1    2-222#   33-30    40-113
R50C10   2-223#   33-32    33-36
R50C17   2-224#   33-34
R50CHR   5-42#    84-17
R50CL    2-219#   33-17    40-95
R50CL0   2-220#   33-19    33-23
R50CL7   2-221#   33-21
R50CSH   2-213#   10-35
R50LD    2-217#   25-62    52-30
R50LOK   2-211#   10-31
R50LS    2-226#
R50MSG   2-209#   10-19
R50ODT   2-227#   24-87
R50PI    2-218#   14-17
R50SY    2-216#
R50TIO   2-214#   10-39
R50TT    2-215#   25-47
R50USR   2-212#   10-27
R50VM    2-225#
```

```
R50WIN    2-210#    10-23
RBD       2-177#
RBR       2-116
RC$$SZ    2-121     36-28
RDB       2-120     25-178
RDBEND    2-120     25-182
RDERR     5-37#      7-46     67-52     69-53     73-74
RDINT     2-116
REDUCE    5-31#     24-33
RELOC     2-101     58-27
REQMIS    5-10#     78-54
RID       2-176#
RLBF      2-233#
RLBFND    2-234#
RMNPDR    2-112     46-56*
RMON      2-104      6-17*     6-167*    24-13     24-115    24-159    53-91     53-92*    53-95*    62-22*    62-61*
ROMBUF   85-33      85-47     85-86#
RPRVEC    2-102     12-48     53-93     60-58
RSFBLK    2-140     30-23     30-30     30-43     30-47     30-63     30-68
RSFERR    5-12#     30-78     30-85
RSR       2-87      40-27
RSZ       2-174#
RT$$SZ    2-122     25-181
RT$BAS    2-121     74-48*
RT$NAM    2-138     74-33*    74-34*
RT$SKP    2-120     74-39
RT$TOP    2-120     74-42*
RTFTCH   29-20      30-24     31-78     42-63     43-37     80-17#
RTMNVC    2-189#     6-167    24-13*    62-61
RTNKM    45-146#
RTTRP4    2-188#     6-166    24-12*
SAVBLK    2-190#
SB$$SZ    2-131     36-50
SCHED     2-88      85-49
SCPFHD    2-86      36-129*
SDANAM    2-119     31-30
SDCB      2-127     31-23
SDCBSZ    2-128     31-68
SDCHAN    2-128     31-47     32-23*    32-30*    32-31*
SDDVU     2-91      31-38*
SDNAME    2-128     31-28*    31-57*
SEGCHN    2-140     30-62
SETCHN   29-80      30-64     31-109    39-58     39-66     42-52     42-90     78-18#
SETJSZ   25-216     48-16#
SETMIO   25-144     65-92#
SETSY    25-110     79-14#
SETUMP    6-109     11-49#
SETVEC   12-36      12-50     13-21#
SFCB      2-113     36-81*
SFCBFH    2-113
SFCBND    2-113     36-83*
SFCBSZ    2-113     36-80
SG$ELG    2-104     24-172    58-20
SG$IOT    2-104     24-173    58-17
SG$MMU    2-105     24-173    53-67
SG$MTM    2-105     24-173
```

```
SG$MTS    2-105     24-172
SG$PAR    2-105     24-172
SG$TSX    2-141     24-173
SHRRCB    2-120     36-23*
SHRRCN    2-120     36-30*
SIZERR   48-59      80-69     81-13     81-20#
SKPDEV    2-228#    52-35
SLTSIZ    2-116     29-41*
SMRSIZ    2-149     68-15*    68-41     68-43*
SNMSHD    2-131     36-49*
SP$$SZ    2-86      36-130
SPLANM    2-127     31-25
SPLBLK    2-130     31-77     31-85     31-90     31-94     31-108    31-110
SPLCHN    2-130     31-107
SPLCLD   31-42      32-13#
SPLDEV    2-127     31-24     50-42
SPLDVN    2-128     31-70
SPLINI   25-130     31-7#
SPLNB     2-121     70-77
SPLND     2-127     31-7      50-40     70-75
SPNEED   29-97      29-108#   30-79     42-103    43-78
SROMMR    2-98       6-118*   24-62     45-110*   57-74*    57-83*    61-24     61-47*    62-33*    62-56*    73-44*    73-53*
         74-62*    74-72*
SR3FLG    2-99       6-119    45-39*    45-107
SR3MMR    2-99       6-123*    6-126*   45-37     45-109*   57-77*    57-82*    61-23     61-50*    62-36*    62-55*    73-47*
         73-52*    74-65*    74-71*
SRTOVF    5-38#     74-105
SRTSIZ    2-149     25-185*
SS        2-116     24-44
SSEND     2-118     24-41
STA       2-172#
STDVTB   51-38      54-20#
STHNPV   55-41      57-100    58-11#
STK       2-173#
STKLVL    2-84       6-25*     6-157*
SVERR     5-35#
SWDBLK    2-88      29-19     29-50     29-58     29-62     29-79     29-84
SWPCHN    2-88      29-78
SWPJOB    2-115     36-127*
SWPPOS    2-115     36-125*
SYINDX    2-119     79-36*
SYNAME    2-140     25-126    42-21     79-43*
SYSDAT    2-123     25-223*
SYSGEN    2-106     24-174*
SYSMAP    2-136
SYSUPD    2-155     40-127    53-54
SYSVER    2-155     24-177*   40-125    41-13     53-51
SYTIMH    2-123     24-19     25-221
SYTIML    2-123     25-220    25-224
SYUNIT    2-118     79-20*    79-40
TAKOVR    6-9#      25-251
TIOBAS    2-109      8-32     10-41*
TIOVEC    2-106      8-34
TK1SEC    2-94      25-10*    25-19*
TK1VAL    2-108     25-13*    25-22*
TK3SVL    2-133     25-12*    25-21*
```

Cross reference table (CREF V05.04)

```
TK5VAL    2-133     25-11*    25-20*
TOOBIG    5-30#     24-32     81-21
TOPMEM    2-199#    24-68*    80-47
TRCSET   45-34      45-141#   53-76     53-77
TRP10     2-107      6-81
TRP14     2-123      6-83
TRP20     2-107      6-85
TRP24     2-107      6-87
TRP250    2-123      6-97
TRP34     2-107      6-91
TRP4      2-106      6-79
TSEMT     2-92
TSEXEC    2-101
TSGEN     2-101
TSINIT    2-6#       2-57      24-77     24-101    29-70     30-55     31-99
TSR       2-116
TSTWRD   45-22#
TSXHD     5-5#       7-45      24-31     24-54     25-226    29-95     29-102    30-77     30-84     31-60     31-131    31-138
         39-75      39-81     42-101    42-108    43-76     43-83     45-151    47-76     59-6      67-51     69-52     69-58
         72-34      73-73     74-93     74-104    75-78     78-53     81-20     85-66
TSXRUN    5-21#     24-55
TSXSAV    2-191#
TSXSIT    2-111     85-25*
TSXSIZ    5-36#     69-59     72-35
TTINCP    2-153     23-29
UBUSMP    2-137     45-127*   51-50
UC$$SZ    2-143     43-25
UCLBLK    2-143     43-28*
UCLDAT    2-143     43-36     43-43     43-48     43-52     43-64*    43-66
UCLINI   25-122     43-10#
UCLNAM    2-140     25-126*
UCLOPN    5-41#     43-77     43-84
UEXINT    2-130      6-42
UEXRTN    2-129      6-46
UK$$SZ    2-143     43-22
UMODE     2-117     45-146
UMRADR    2-137
UMSYTP    2-118     25-154*
UNIBUS    2-137     45-123
UPAR0     2-100     44-15
UPAR6     2-122     74-57*
UPAR7     2-93
UPDR0     2-100     44-17
UPDR6     2-122     74-58*
UPMODE    2-121     74-59
US$$SZ    2-143     43-23
USRBAS    2-103     10-29*
VBUSTP    2-119     24-167    45-117    45-123
VCSHNB    2-126     75-17*
VDBFLG    2-92      70-155
VDFMEM    2-133     48-47     48-49
VF$LIN    2-153
VF$MR     2-152
VF$RE     2-153
VF$RIE    2-152
VF$SC     2-153
```

Cross reference table (CREF V05.04)

```
VF$TIE    2-152
VH$CSR    2-152
VH$LCR    2-153
VH$LPR    2-152
VHIMEM    2-87     29-38     48-35     48-37
VHINIT   22-15#
VHLPRM   22-11#
VHOINT    2-153
VINABT    2-122    31-58     53-30     53-56     74-31
VLDSYS    2-144    25-57     52-32
VMAXMC    2-144    70-97
VMIOBF    2-146
VMIOSZ    2-145    38-83
VMLBLK    2-115    38-59
VMSCHR    2-91     70-104
VMXCSH    2-86     36-65
VMXMON    2-111    36-41
VMXMRB    2-147    70-101
VMXMSG    2-144    70-108
VMXSF     2-115    38-51     38-57     70-88
VMXSFC    2-115    38-54     38-62
VMXWIN    2-114    36-25     36-27     70-135
VNCSLO    2-91     37-32
VNCXOF    2-91     37-56
VNCXON    2-91     37-63
VNFCSH    2-134    70-68
VNGR      2-121    36-24
VNUIP     2-85     36-55
VNUMDC    2-85     38-53     38-65     38-72     70-90
VPAR5     2-135    24-69     24-71     47-49     47-53     57-34     57-70     57-97     73-35
VPAR6     2-134    38-27     74-56
VPLAS     2-140    30-18     30-26     30-36     30-47     70-114
VSLEDT    2-136    70-129
VSWPFL    2-101    25-134    30-16     36-121    70-123
VSWPSL    2-95     29-32     29-34*    29-45     36-123
VSYDMP    2-84     70-167
VU$CL     2-143    43-15
VUCLMC    2-143    43-17
VUXIFL    2-129     6-44      6-51
WINBAS    2-103    10-25*
WRKBUF    2-207#   25-239    25-243    26-20*    39-26     42-28     42-82     43-55     53-39     53-47     53-66     53-78
         54-43     57-28     57-62     60-35     60-50     60-57     67-16     69-23     73-29     74-53     80-38
WRKSIZ    2-208#   25-237    26-21*    80-40
XMVBAS    2-195#   13-47     13-72*    25-101*
ZCLR      2-109
```

Cross reference table (CREF V05.04)

Cross reference table (CREF V05.04)

```
        85-67
. PURGE  2-50#   42-72   43-47
. READW  2-49#    7-29   25-243  39-27   42-29   42-31   53-39   55-25   57-29   57-63   60-36   60-50
        61-35    67-17   69-24   73-30   74-54
. RELEA  2-52#   29-84   30-68   31-110  42-91   43-66
. SAVES  2-49#   78-25
. SCCA   2-53#   24-23
. SERR   2-51#   24-50
. SETTO  2-50#   24-67
. TRPSE  2-50#   24-61   24-63
. TTYOU  2-50#   82-18   83-26   84-26
. UNLOC  2-51#   25-232
. WRITW  2-51#   29-70   30-55   31-99   42-82   43-55
CLVX     4-53#    4-61    4-62    4-63    4-64    4-65    4-66    4-67
DEFFLG   4-8#     4-14    4-15    4-16    4-17    4-18    4-19    4-20    4-21    4-22    4-23    4-24
         4-25    4-26    4-27    4-28    4-29    4-30    4-31    4-32    4-33    4-34    4-35    4-36
         4-37    4-38    4-39    4-40    4-41
DISABL   2-161#  57-71   62-18   73-41   74-61
ENABL    2-165#  57-85   62-65   73-55   74-73
HANVER   3-9#     3-30    3-31    3-32
OVLTBL  70-45#   70-51   70-52   70-53   70-54   70-55   70-56   70-57   70-58   70-59   70-60   70-61
        70-62    70-63
```