

5-	1	* * * TSX Initialization * * *
5-	2	* * * Initialization taking over control from RT-11 * * *
6-	1	LODINI -- Load a segment over TSINIT
7-	1	INIOVL -- Load system overlays over TSINIT
8-	1	ENTVEC -- Set up entry point vector for overlay
9-	1	KEYSEG -- Remember memory position of system overlays
10-	2	SETUMP -- Set up Unibus mapping if needed
11-	1	DEVVEC -- Set up device vectors
12-	1	SETVEC -- Set up an interrupt vector for a device
14-	2	LININI -- Initialize time-sharing lines
15-	1	DHLPRM -- Set line parameters for a DH11 line
16-	1	VHLPRM -- Set line parameter values for DHV11 line
17-	1	DZINIT -- Initialize a DZ11 multiplexer
18-	1	MUXVEC -- Set up interrupt vectors for a multiplexer
19-	1	DHINIT -- Initialize a DH11 multiplexer
20-	1	VHINIT -- Initialize a DHV11 multiplexer
22-	1	LINTYP -- Determine the type of a line
23-	1	* * * Initialization done with RT-11 running * * *
25-	1	* * * Subroutines * * *
25-	2	ALCWRK -- Allocate a work buffer
26-	1	ALCHRB -- Allocate Region Control Blocks for handlers
27-	2	LINCHK -- Check validity of T/S line
28-	1	OPNSWP -- Open system swap file
29-	1	OPNRSF -- Open PLAS region swap file
30-	1	SPLINI -- Initialize spooling system
31-	1	SPLCLD -- Set up spooling to a CL device
32-	1	CHKCLD -- See if a device name is a CL or C1 unit
33-	1	CVTDVU -- Convert device name to dev index and unit #
34-	1	FORCEO -- Force a 2-char dev name to unit 0
35-	1	ALOCBF -- Allocate buffer space
36-	1	ALCSLO -- Allocate silo buffers for lines
37-	1	ALBFX -- Allocate buffers in extended memory region
38-	1	OPNKMN -- Open channel to TSKMON
39-	1	CLINIT -- Initialize CL handler
40-	1	LDINIT -- Determine LD translation table format
41-	1	INDINI -- Initialize IND program
42-	1	UCLINI -- Initialize TSXUCL data file
43-	1	MEMINI -- Initialize memory management
44-	1	MEMTST -- Set up information about available memory space
45-	1	CXTALC -- Set up info about job context area
46-	1	MAPALC -- Allocate memory usage table
47-	1	SETJSZ -- Set up information about maximum job sizes
48-	2	PARSET -- Setup memory parity control
49-	1	GETHNL -- Load device handlers into memory
50-	1	LDHAND -- Load a device handler
51-	1	INSCK1 -- Determine if a handler should be installed
52-	1	INSCK2 -- Additional checking for handler installation
53-	1	STDVTB -- Set up device table entries for a device
54-	1	LDHNLO -- Load device handler into low memory
55-	1	GETHNH -- Load handlers into extended memory
56-	1	LDHNHI -- Load device handler into extended memory
57-	1	STHNPV -- Initialize pointer vector in a handler
59-	1	DOHNLC -- Execute and handler load/fetch code
60-	1	LDREAD -- Perform I/O for handler load code
61-	1	HANMAP -- Set up KPAR5 to access a mapped handler
61-	42	HANUMP -- Turn off memory mapping to a handler
62-	1	FNDHRB -- Try to find a handler global region

Table of contents

63-	1	HANXMR -- Allocate XM region during handler load
64-	2	SETMIO -- Set up information about mapped devices
65-	1	OVLPOS -- Determine which overlays go over TSINIT
66-	1	OVLBLD -- Build overlay information table
67-	1	GETMAP -- Load any mapped system code regions
68-	1	ALCOVL -- Allocate space for a system overlay region
69-	1	OPTOVL -- Check for optional system overlay regions
70-	1	OVLTRY -- Find an overlay to place over TSINIT
71-	1	GETOVL -- Load system overlay into high memory
72-	1	LODOVL -- Read and relocate system overlay
73-	1	GETSRT -- Load any shared run-time systems
74-	1	CSHBUF -- Allocate space for data cache tables
75-	2	GETODT -- Load ODT
76-	1	OPNCHN -- Open a TSX-Plus channel
77-	1	SETCHN -- Copy RT-11 channel information into TSX system chan
78-	1	SETSY -- Set up information about SY device
79-	1	RTFTCH -- Fetch a RT-11 device handler
80-	1	CHKMEM -- Check for memory space overflow
81-	1	PRTOCT -- Print octal value
82-	1	PRTDEC -- Print decimal value
83-	1	PRTR50 -- Print Rad-50 value

```

1          .TITLE  TSINIT -- TSX startup initialization
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  GBL
5 000000    .CSECT  TSINIT
6 000000
7
8          ;
9          ;   There are two external assembly-time switches related to assembling
10         ;   TSINIT for execution on a PRO or a PDP-11.
11
12         ;   The following values for the PROASM flag are defined:
13         ;   0 ==> Assemble for PDP-11 (not Pro) only.
14         ;   1 ==> Assemble for Pro only.
15         ;   2 ==> Assemble for either PDP-11 or Pro execution.
16
17         ;   The following values for the PROCID flag are defined:
18         ;   0 ==> Do not lock system to ID number.
19         ;   1 ==> Lock system to ID number.
20
21         .IF      NDF,PROASM      ; If PROASM not defined
22         PROASM   =      0        ; Default value for PROASM if not defined
23         .ENDC      ; NDF,PROASM
24
25         .IF      NDF,PROCID      ; If PROCID not defined
26         .IF      EQ,<PROASM-1>    ; If assembling for PRO only
27         PROCID   =      1        ; Then check ID by default
28         .IFF      ; If not assembling for PRO only
29         PROCID   =      0        ; Then don't check ID number
30         .ENDC      ; EQ,<PROASM-1>
31         .ENDC      ; NDF,PROCID
32
33         .IF      EQ,PROASM
34         .GLOBL   TSXPRO
35         TSXPRO   =      0        ; Define dummy base for TSXPRO if not PRO
36         .ENDC
37
38         ; -----
39         ;   TSINIT is the initialization module of TSX that is executed once
40         ;   during system startup. Time-sharing character buffers and other
41         ;   run-time data areas are allocated over TSINIT.
42
43         ;   Copyright 1980, 1981, 1982, 1983, 1984, 1985.
44         ;   S&H Computer Systems, Inc.
45         ;   Nashville, TN  USA
46
47         ;   Macro calls
48
49         .MCALL   .LOOKUP,.ENTER,.READW,.SAVESTATUS,.GVAL
50         .MCALL   .TRPSET,.SETTOP,.CLOSE,.TTYOUT,.PRINT,.PURGE
51         .MCALL   .DELETE,.WRITW,.SERR,.HERR,.EXIT,.UNLOCK
52         .MCALL   .FETCH,.RELEASE,.LOCK,.GTIM,.DATE,.DSTATUS
53         .MCALL   .SCCA,.CSTAT
54
55         ;   Global definitions
56
57         .GLOBL   TSINIT,INITGO,INITOP,PPTERM,PROITP,PROASM,PISRT

```

```

58          .GLOBL  DSKBUF,PROBUF,FNDHRB,HANXMR
59
60          ; Following global only needed for the Pro distribution
61          ; creation program - MAKPRO and installation program - INSTSX
62
63          .IF      NE,PROASM
64          ;
65          ;** Assemble this code if we are generating for a Pro
66          ;
67          .GLOBL  PROSIZ,PROINI,PROLIN,PROHAN,PRONOP
68          .GLOBL  PIHAN,PIDPTR,PIDRIV
69          .IFF      ;NE,PROASM
70          ;
71          ;** Assemble this code if we are not generating for a Pro
72          ;
73          .GLOBL  TSXPRO
74          TSXPRO  =      0
75          ;
76          ;** End of conditional Pro code
77          ;
78          .ENDC      ;NE,PROASM
79
80          ;
81          ; Global references
82          ;
83          .GLOBL  HANDSK,MAXDEV,NDVRCB,HANRCB,HANRCO
84          .GLOBL  LXCL,VSYDMP,STKLVL,INTSSZ,INDFIL,NXIVMH,EXCBUF
85          .GLOBL  VNUIP,NSIP,INSTBL,INSTBN,II##SZ,DCCSZ,VNUMDC,NUMCDB
86          .GLOBL  NSCP,SCPFHD,SP##SZ,CSHDEV,CSHDVN,VMXCSH,CD##SZ
87          .GLOBL  LSTPL,LMXNUM,MXCSR,MXVEC,RSR,INVEC,VHIMEM,CXTPAG
88          .GLOBL  LSWPBK,LSTSL,SWDBLK,SWPCHN,NUMDEV,CS#NMX,SCHED
89          .GLOBL  H.DSTS,DVSTAT,HANENT,H.GEN,FORK,INTEN,PNAME
90          .GLOBL  $SXON,LSW10,LHIRBB,LHIRBE,LHIRBP,LHIRBG,LHIRBA
91          .GLOBL  LHIRBS,LHIRBC,VNCSLO,VNCXOF,VNCXON,SDDVU,VMSCHR,MAXSLO
92          .GLOBL  MPAR0,MPAR16,PARENL,MPARFL,TSEMT,VDBFLG,DX#EBA
93          .GLOBL  H.SIZ,HANSIZ,H.DVSZ,DEVSIZ,LOMAP,MMENBL,UPAR7
94          .GLOBL  PSW,HIMAP,FSTDLL,LSTDLL,LINBUF,LINSIZ,NUMCCB,TK1SEC
95          .GLOBL  FRKINI,FRKGEN,NUMFRK,FQ##SZ,H.CSR,H.INS,VSWPSL,DYDEV
96          .GLOBL  LINEND,LOTBUF,LOTSIZ,LOTEND,KMNTOP,KMNH1,NSL,NDL
97          .GLOBL  DX$MPH,DX$NHM,DX$IBH,HANPAR,HANXIT,MAPPAR,LINSPC
98          .GLOBL  KMNPGS,KMNSTK,KMNSTR,KMNCHN,SROMMR,KPAR0,PROFLG
99          .GLOBL  EMMAP,IOMAP,SR3MMR,IOPAGE,MAPSIZ,SR3FLG,NSPLDV
100         .GLOBL  UPAR0,KPDRO,UPDRO,KPAR7,BASMAP,PTWRD,PTBYT,LOKMEM
101         .GLOBL  GTBYT,MPPHY,RELOC,BRKPT,TSGEN,TSEXC,VSWPFL
102         .GLOBL  CW$GDH,CW$BTH,CW$LGS,CW$FB,CW$FGJ,MSGBAS,RPRVEC
103         .GLOBL  CW$USR,CW$XM,CONFIG,CW$50H,JMPO,DTLX,USRBAS,WINBAS
104         .GLOBL  DATIML,DATIMH,RMON,CONF2,SG$ELG,SG$IOT,CSHBAS
105         .GLOBL  SG$PAR,SG$MTS,SG$MMU,SG$MTM,LTPAR,LOKBAS,CSHVEC,LOKVEC
106         .GLOBL  SYSGEN,AUTHAN,AHEND,CLKRTI,TRP4,CW$PRO,TIOVEC
107         .GLOBL  TRP10,TRP20,TRP24,EMTENT,TRP34,INIIMP,MHNSIZ
108         .GLOBL  TK1VAL,INRECV,OTRECV,INMXV,OTMXV,DHBFSZ,MXTYPE
109         .GLOBL  ZCLR,MXRBUF,MXDTR,INTMX1,$PHONE,LCDTYP,TIOBAS
110         .GLOBL  LDHB1B,LDHB1P,LDHB2B,CLVERS,CXTSIZ,CXTWDS,CXTPDR
111         .GLOBL  CLORSZ,TSXSIT,JM##SZ,VMXMON,MONFGH,CXTRMN,CXTBAS
112         .GLOBL  ILSW2,$NOIN,LSW3,MXLPR,CW$ESP,CLTOTL,RMNPDR,MA$SYS
113         .GLOBL  SFCB,SFCBND,SFCBFH,SFCBSZ,NSPLFL,NSPLBL,INTSTK,INTSND
114         .GLOBL  NFRESB,PVSPBL,VMXWIN,DW##SZ,LDVERS,CW$QBS

```

```

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

```

```

; -----
;   Macros to enable and disable interrupts
;

```

```

161 . MACRO DISABL ; Disable interrupts
162 BIS #340, @#PSW
163 . ENDM DISABL
164
165 . MACRO ENABL
166 BIC #340, @#PSW
167 . ENDM ENABL

```

```

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

```

```

172          000040          STA      =      40          ; PROGRAM START ADDRESS
173          000042          STK      =      42          ; INITIAL STACK POINTER
174          000052          RSZ      =      52          ; ROOT SIZE
175          000056          OSZ      =      56          ; OVERLAY SIZE
176          000060          RID      =      60          ; REL FILE ID
177          000062          RBD      =      62          ; DISPLACEMENT TO 1ST REL BLOCK
178          001000          ODTBAS   =      1000         ; BASE ADDRESS ODT WAS LINKED FOR
179
180          ; Data areas
181
182 000000          AREA:      .BLKW      8.
183 000020 000000 000000 000000 NFSBLK: .WORD      0,0,0,0,0,0 ; EXTENDED TO 6 WORDS FOR .CSTAT
184          000026 000000 000000 000000
185 000034 000000          ODTFLG: .WORD      0
186 000036 000000          ODTTOP: .WORD      0
187 000040 000000          CCAFLG: .WORD      0
188 000042 000000          CLK100: .WORD      0
189 000044 000000          RTTRP4: .WORD      0
190 000046 000000          RTMNVC: .WORD      0
191 000050          SAVBLK: .BLKW      5
192 000062 075250 100020 000000 TSXSAV: .RAD50 /SY TSX SAV/
193          000070 073376
194 000072 075250 100003 051646 KMNNAM: .RAD50 /SY TSKMONSAV/
195          000100 073376
196 000102 075250 011504 000000 CCLNAM: .RAD50 /SY CCL SAV/
197          000110 073376
198 000112 000000 000000 000000 DSTBLK: .WORD      0,0,0,0
199          000120 000000
200 000122 000000          XMVBAS: .WORD      0
201 000124 000000          NMXHAN: .WORD      0
202 000126 000000          HMAP: .WORD      0
203 000130 000000          FETDEV: .WORD      0
204 000132 000000          TOPMEM: .WORD      0
205 000134 000000          FMEMHI: .WORD      0 ; 64-byte block # below high alloc memory
206 000136 000000          FMEMLO: .WORD      0 ; 64-byte block # above top of low alloc memory
207 000140 000000          OVLBAS: .WORD      0 ; Start loading overlays over TSINIT from here
208 000142 000000          FILBLK: .WORD      0
209 000144 000000          CURDEV: .WORD      0
210 000146 000000          CURNAM: .WORD      0
211 000150 000000          PROBUF: .WORD      0
212 000152 030066 WRKBUF: .WORD      INITOP
213 000154 004000          WRKSIZ: .WORD      2048
214 000156 052077          R5OMSG: .RAD50 /MSG/
215 000160 110466          R5OWIN: .RAD50 /WIN/
216 000162 046543          R5OLOK: .RAD50 /LOK/
217 000164 103112          R5OUSR: .RAD50 /USR/
218 000166 012700          R5OC SH: .RAD50 /CSH/
219 000170 077167          R5OTIO: .RAD50 /TIO/
220 000172 100040          R5OTT: .RAD50 /TT /
221 000174 075250          R5OSY: .RAD50 /SY /
222 000176 045640          R5OLD: .RAD50 /LD /
223 000200 062550          R5OPI: .RAD50 /PI /
224 000202 012240          R5OCL: .RAD50 /CL /
225 000204 012276          R5OCLO: .RAD50 /CLO/
226 000206 012305          R5OCL7: .RAD50 /CL7/
227 000210 013630          R5OC1: .RAD50 /C1/
228 000212 013666          R5OC10: .RAD50 /C10/

```

```

224 000214 013675      R50C17: .RAD50  /C17/
225 000216 105610      R50VM:  .RAD50  /VM /
226 000220 046770      R50LS:  .RAD50  /LS /
227 000222 057164      R50ODT: .RAD50  /ODT/
228 000224 100040 015270 075250 SKPDEV: .RAD50  /TT DK SY CL C1 PI  /
      000232 012240 013630 062550
      000240 000000
229 000242      000      110      GTLIN:  .BYTE  0,110
230 000244 075250 114730 000000 HANNAM: .RAD50  /SY XXX  TSX/
      000252 100020
231 000254 075250 075273 057164 ODTBLK: .RAD50  /SY SYSODTREL/
      000262 070524
232 000264 075250 035164 000000 INDNAM: .RAD50  /SY IND  SAV/
      000272 073376
233 000274 000000      RLBF:   .WORD  0
234 000276 000000      RLBFND: .WORD  0
235 000300 000000      ODTSTA: .WORD  0
236 000302 000000      MEM LIM: .WORD  0
237 000304 000000      HGENFL: .WORD  0
238
239      ; Initialization configuration word
240
241 000306 000000      ICONFG: .WORD  0      ; Initialization configuration word
242
243      ; Flag bits in ICONFIG
244
245      000001      EXTLSI =      1      ; Q-bus system with more than 256Kb
246
247      ; Simulated shared run-time control block for PI handler
248
249 000310 075250 062550 000000 PISRT:  .RAD50  /SY PI  TSX/
      000316 100020
250 000320 000000 000000 000000      .WORD  0,0,0
251
252      ; Byte data cells
253
254 000326      000      PPTERM: .BYTE  0      ; 1 if printer port is T/S terminal
255      .EVEN

```

```

1      ; -----
2      ; The following tables are used to determine the minimum RT-11 monitor
3      ; and update versions required for particular devices.
4      ; There are three arguments for each handler definition:
5      ;   Arg 1 = Handler id code.
6      ;   Arg 2 = Minimum acceptable RT-11 version.
7      ;   Arg 3 = Minimum acceptable RT-11 update within the version.
8      ;
9      ; .MACRO HANVER DEVID,RTVERS,RTUPDT
10     ; .BYTE DEVID      ; ID code for device type
11     ; .BYTE RTVERS     ; Minimum RT-11 version
12     ; .BYTE RTUPDT     ; Minimum update level within version
13     ; .ENDM HANVER
14
15     ; Define offsets into handler version table
16
17     000000 HV$ID = 0      ; Handler identification code
18     000001 HV$VER = 1     ; Minimum RT-11 version
19     000002 HV$UPD = 2     ; Minimum update level within version
20     000003 HV$$SZ = 3     ; Size of handler version table entry
21
22     ; Define minimum versions for various handlers
23     ; *****
24     ; *** Note RT-11 version update numbering system changed ***
25     ; *** at 5.2 so that the update number for 5.2 is lower ***
26     ; *** than for 5.1C. See the CLVX entries below for ***
27     ; *** version and update correlations. ***
28     ; *****
29 000330 HVTBL:
30 000330     HANVER DI$DU,5.,0.      ; DU - 5/0 (5.0)
31 000333     HANVER DI$XL,5.,2.      ; XL - 5/6 (5.1B) ; Fixed for 5.2 SCB
32 000336     HANVER DI$MU,5.,4.      ; MU - 5/4 (5.4)
33 000341 HVEND:
34     . EVEN

```

```

1      ; -----
2      ; The following table defines default control flags for certain devices.
3      ;
4      000000 DV$NAM = 0 ; Rad50 name of device
5      000002 DV$FLG = 2 ; Flags for device
6      000004 DV$$SZ = 4 ; Size of a table entry
7      ;
8      . MACRO DEFFLG DEV, FLAGS
9      . RAD50 / 'DEV/ ; DV$NAM
10     . WORD FLAGS ; DV$FLG
11     . ENDM DEFFLG
12     ;
13     DVFLBS:
14     DEFFLG <CR>, <DX$MPH>
15     DEFFLG <CT>, <DX$MPH>
16     DEFFLG <DB>, <DX$DMA!DX$MPH>
17     DEFFLG <DD>, <DX$NHM>
18     DEFFLG <DL>, <DX$DMA!DX$MPH!DX$IBH>
19     DEFFLG <DM>, <DX$DMA!DX$NHM>
20     DEFFLG <DP>, <DX$DMA>
21     DEFFLG <DS>, <DX$DMA>
22     DEFFLG <DT>, <DX$DMA>
23     DEFFLG <DU>, <DX$DMA!DX$NHM!DX$NST>
24     DEFFLG <DW>, <DX$MPH>
25     DEFFLG <DX>, <DX$MPH>
26     DEFFLG <DY>, <DX$DMA!DX$NHM>
27     DEFFLG <DZ>, <DX$MPH>
28     DEFFLG <FW>, <DX$DMA>
29     DEFFLG <LP>, <DX$MPH>
30     DEFFLG <LS>, <DX$MPH>
31     DEFFLG <MM>, <DX$DMA!DX$MPH!DX$IBH>
32     DEFFLG <MS>, <DX$DMA!DX$MPH!DX$IBH>
33     DEFFLG <MT>, <DX$DMA!DX$MPH!DX$IBH>
34     DEFFLG <MU>, <DX$DMA!DX$NHM!DX$IBH!DX$NST>
35     DEFFLG <NL>, <DX$MPH>
36     DEFFLG <PC>, <DX$MPH>
37     DEFFLG <RF>, <DX$DMA>
38     DEFFLG <RK>, <DX$DMA!DX$MPH>
39     DEFFLG <VM>, <DX$EBA!DX$NCA!DX$NHM>
40     DEFFLG <XC>, <DX$MPH>
41     DEFFLG <XL>, <DX$MPH>
42     DVFLND:
43
44     ; -----
45     ; The following table specifies which version number is to be
46     ; returned by CL in response to the XL/CL .SPFUN used by
47     ; VTCOM to match it to the handler.
48     ; The macro has 3 arguments:
49     ; 1. Minimum RT-11 version number where this CL version should be used.
50     ; 2. Minimum RT-11 update number where this CL version should be used.
51     ; 3. CL version number that starts with specified RT-11 version.
52     ;
53     . MACRO CLVX RTV, RTU, CLV
54     . BYTE RTU, RTV
55     . WORD CLV
56     . ENDM CLVX
57     ;

```

```

58      ; Define CL versions based on RT-11 versions
59      ;
60      CLVTBL:
61          CLVX      5,1,16.      ; Version 5.1
62          CLVX      5,6,16.      ; Version 5.1B
63          CLVX      5,35,16.     ; Version 5.1
64          CLVX      5,44,16.     ; Version 5.1C
65          CLVX      5,2,17.      ; Version 5.2
66          CLVX      5,3,17.      ; Version 5.3
67          CLVX      5,4,18.      ; Version 5.4
68      CLVEND:
69      ;
70      ; -----
71      ; RT-11 v5.4 changed the structure of the LD translation tables.
72      ; The following table is used to determine which table format
73      ; to return to LD spfun 372.
74      ;
75      ; Use original format for the following versions:
76      ;
77      LD1TBL:
78          .BYTE      5,0          ; Version 5.0
79          .BYTE      5,1          ; Version 5.1
80          .BYTE      5,6          ; Version 5.1B
81          .BYTE      5,35         ; Version 5.1
82          .BYTE      5,44         ; Version 5.1C
83          .BYTE      5,2          ; Version 5.2
84          .BYTE      5,3          ; Version 5.3
85          .WORD      0
86      ;
87      ; -----
88      ; The following data structures are used to hold information about
89      ; TSX-Plus overlays as they are being initialized.
90      ;
91      ; Offsets in structure for each overlay
92      ;
93      OS$SIZ =      0          ; Total space needed for overlay
94      OS$FLG =      2          ; 0==>Load into XM space, 1==>over TSINIT
95      OS$OVL =      4          ; Pointer to overlay table entry
96      OS$$SZ =      6          ; Size of each overlay entry
97      ;
98      MAXOVL =      25.        ; Maximum number of system overlays
99      ;
100     OSTABL: .BLKB  OS$$SZ*MAXOVL ; Reserve room for table
101     OSEND:  ; -Define end of table
102     OSLAST: .WORD  OSTABL        ; Pointer past last used entry in table
103     ;
104     ; Table of system overlays that must be loaded over TSINIT
105     ;
106     LOWOVL: .RAD50 /CSH/          ; TSCASH
107             .RAD50 /TIO/          ; TSTIO
108             .RAD50 /LOK/          ; TSLOCK
109     LOWEND: ; End of table

```

```

1      ; -----
2      ;   Text messages
3      ;
4      ;   .NLIST   BEX
5 001034      077      124      123   TSXHD:   .ASCII   /?TSX-F-/<200>
6 001044      111      156      166   BADLIN: .ASCII   'Invalid status register address for T/S line: '<200>
7 001123      111      156      166   BDVMSG: .ASCII   'Invalid interrupt vector address for T/S line: '<200>
8 001203      114      151      156   BDLMSG: .ASCII   /Line # = /<200>
9 001215      000
10 001216      124      123      130   REQMIS: .ASCII   /TSX generation did not include device /<200>
11 001265      103      141      156   BADOPN: .ASCIZ   /Cannot open program swap file/
12 001323      103      141      156   RSFERR: .ASCIZ   /Cannot open PLAS region swap file/
13 001365      116      165      155   CONSPC: .ASCII   /Number of contiguous blocks needed = /<200>
14 001433      103      141      156   BDSPOP: .ASCII   /Cannot open spooled device: /<200>
15 001470      111      156      163   BOSF:   .ASCIZ   /Insufficient disk space for spool file/
16 001537      103      141      156   NDKMON: .ASCIZ   /Cannot find "SY:TSKMON.SAV" file/
17 001600      103      141      156   NDCCL:  .ASCIZ   /Cannot find "SY:CCL.SAV" file/
18 001636      103      141      156   CFHMSG: .ASCII   /Cannot find device handler file: /<200>
19 001700      105      162      162   ERHMSG: .ASCII   /Error reading device handler file: /<200>
20 001744      111      156      166   ERHNDV: .ASCII   /Invalid RT-11 version for device handler: /<200>
21 002016      124      123      130   TSXRUN: .ASCIZ   /TSX is already running/
22 002045      110      141      156   HSGER:  .ASCII   /Handler not generated with extended memory support: /<200>
23 002132      103      157      155   NDCLOK: .ASCIZ   /Computer line time clock (50 or 60 Hz) is not working/
24 002220      123      171      163   NXMMSG: .ASCIZ   /System is not equipped with memory management hardware/
25 002307      123      171      163   NEXMSG: .ASCIZ   /System is not equipped with extended memory management hardware/
26 002407      103      141      156   NODDT:  .ASCIZ   /Cannot locate "SY:SYSODT.REL" file/
27 002452      115      141      160   HN2BIG: .ASCII   /Mapped handler is larger than 8KB: /<200>
28 002516      105      162      162   ODTRDM: .ASCIZ   /Error on read of SYSODT rel file/
29 002557      110      141      156   NOSYDV: .ASCIZ   /Handler for SY device was not loaded/
30 002624      107      145      156   TOOBIG: .ASCIZ   /Generated TSX system is too large/
31 002666      122      145      144   REDUCE: .ASCII   /Reduce size of TSGEN by /<200>
32 002717      056      040      142   BYTES:  .ASCIZ   /. bytes/
33 002727      111      156      163   PHSQVF: .ASCIZ   /Insufficient total physical memory for generated system/
34 003017      103      141      156   COSRT:  .ASCII   /Cannot open shared run-time file: /<200>
35 003062      103      141      156   SVERR:  .ASCIZ   /Cannot locate "SY:TSX.SAV"/
36 003115      111      156      163   TSXSIZ: .ASCIZ   /Insufficient memory to load all mapped system regions/
37 003203      105      162      162   RDERR:  .ASCIZ   /Error reading "SY:TSX.SAV"/
38 003236      111      156      163   SRTOVF: .ASCIZ   /Insufficient memory to load all shared run-time systems/
39 003326      111      156      163   CSHOVF: .ASCIZ   /Insufficient memory space for data cache/
40 003377      103      141      156   INDOPN: .ASCIZ   /Cannot open TSXIND file/
41 003427      103      141      156   UCLOPN: .ASCIZ   /Cannot open TSXUCL data file/
42 003464      040      101      102   R5OCHR: .ASCII   / ABCDEFGHIJKLMNOPQRSTUVWXYZ#. #0123456789/
43      .EVEN
44      .LIST   BEX

```

* * * TSX Initialization * * *

```

1          .SBTTL * * * TSX Initialization * * *
2          .SBTTL * * * Initialization taking over control from RT-11 * * *
3          ;-----
4          ; The initialization code from this point onward takes over
5          ; control from RT-11.
6          ; This code is placed at the front of TSINIT so that non-initialized
7          ; data structures can be allocated over it.
8          ;
9 003534    TAKOVR:
10         ;
11         ; Read in system overlays that go over TSINIT
12         ;
13 003534    004737    004374'    CALL    INIOVL    ;Read overlays over TSINIT
14         ;
15         ; Set pointer to monitor offset vector
16         ;
17 003540    012737    000000G 000000G    MOV     #MONVEC,@#RMON ;SET POINTER TO MONVEC TABLE
18         ;
19         ; Initialize last word in interrupt stack area so we won't report a
20         ; stack overflow if an interrupt occurs.
21         ; Set STKLVL to 0 to cause INTEN not to switch to interrupt
22         ; stack during initialization.
23         ;
24 003546    012777    123456    000000G    MOV     #123456,@INTSND ;Say stack has not overflowed
25 003554    105037    000000G    CLRB     STKLVL    ;Say we are already on interrupt stack
26         ;
27         ; If we are running on a Professional, disable its interrupts
28         ;
29         .IF      NE,PROASM
30             TSTB     PROFLG    ;Are we running on a PRO?
31             BEQ      7$        ;Br if not on a PRO
32             CALL     PRNOP     ;Disable its interrupts
33             BR       5$        ;Ignore unexpected interrupts on PRO
34         .ENDC    ; NE,PROASM
35         ;
36         ; Set up vectors to catch unexpected interrupts
37         ; Note: We encode the interrupt vector address in the PS --
38         ; the low-order two bits of the address are dropped (they are
39         ; always zero) and the remainder of the address is encoded in the
40         ; PS fields priority (high-order 3 bits) and n-z-v-c (low-order 4 bits).
41         ;
42 003560    012702    000000G    7$:    MOV     #UEXINT,R2    ;SEND UNEXPECTED INTERRUPTS TO THIS ROUTINE
43 003564    012700    000044    MOV     #44,R0    ;120 ENCODED IN PS FIELDS
44 003570    105737    000000G    TSTB     VUXIFL    ;ARE WE TO IGNORE UNEXPECTED INTERRUPTS?
45 003574    001004    BNE     10$    ;BR IF NOT
46 003576    012702    000000G    5$:    MOV     #UEXRTN,R2    ;ROUTINE TO GO TO TO IGNORE INTERRUPT
47 003602    012700    000340    MOV     #340,R0    ;SET PRIO=7 IN PS
48 003606    012701    000120    10$:    MOV     #120,R1    ;INIT ALL VECTORS STARTING AT 120
49 003612    010221    1$:    MOV     R2,(R1)+    ;SET PC FOR INTERRUPT
50 003614    010021    MOV     R0,(R1)+    ;SET PS FOR INTERRUPT (ENCODED ADDRESS VALUE)
51 003616    105737    000000G    6$:    TSTB     VUXIFL    ;ARE WE TO IGNORE UNEXPECTED INTS?
52 003622    001411    BEQ     2$        ;BR IF YES
53 003624    105737    000000G    TSTB     PROFLG    ;IS THIS A PRO?
54 003630    001006    BNE     2$        ;BR IF YES
55 003632    005200    INC      R0        ;ADVANCE ENCODED ADDRESS
56 003634    032700    000020    BIT      #20,R0    ;DID WE CARRY INTO "T"-FIELD?
57 003640    001402    BEQ     2$        ;BR IF NOT

```

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

```

58 003642 062700 000020          ADD    #20,R0          ; FORCE CARRY OUT OF T-FIELD AND INTO PRI0 FIELD
59 003646 020127 000420          2$:  CMP    R1,#420      ; DONE ALL INTERRUPT VECTORS OF INTEREST?
60 003652 103757                BLO    1$              ; BR IF NOT
61 003654 010237 000060          MOV    R2,@#60         ; CATCH CONSOLE TERMINAL VECTOR TOO
62 003660 012737 000014 000062  MOV    #14,@#62      ; ENCODED 60
63 003666 010237 000064          MOV    R2,@#64
64 003672 012737 000015 000066  MOV    #15,@#66      ; ENCODED 64
65                                ;
66                                ; Direct clock interrupt to a dummy RTI instruction to avoid it causing
67                                ; trouble during the initialization process when things aren't set up
68                                ; and ready to go.
69                                ;
70 003700 012700 000340          11$:  MOV    #340,R0      ; PRIORITY 7 PS
71 003704 012737 000000G 000000G MOV    #CLKRTI,@#CLKVEC; Send clock interrupt to RTI instruct for now
72 003712 010037 000002G          MOV    R0,@#CLKVEC+2
73                                ;
74                                ; Take over traps, EMT, BPT, etc.
75                                ;
76 003716 005001                CLR    R1              ; Start at location 0
77 003720 012721 000137          MOV    #137,(R1)+      ; [JMP @#JMP0] ==> 0
78 003724 012721 000000G          MOV    #JMP0,(R1)+    ; CATCH JUMPS TO LOCATION 0
79 003730 012721 000000G          MOV    #TRP4,(R1)+    ; TRAP 4
80 003734 005021                CLR    (R1)+
81 003736 012721 000000G          MOV    #TRP10,(R1)+   ; TRAP 10
82 003742 005021                CLR    (R1)+
83 003744 012721 000000G          MOV    #TRP14,(R1)+   ; TRAP 14 (BREAKPOINTS)
84 003750 010021                MOV    R0,(R1)+
85 003752 012721 000000G          MOV    #TRP20,(R1)+   ; IOT TRAP
86 003756 005021                CLR    (R1)+
87 003760 012721 000000G          MOV    #TRP24,(R1)+   ; POWER FAIL
88 003764 010021                MOV    R0,(R1)+
89 003766 012721 000000G          MOV    #EMTENT,(R1)+  ; EMT
90 003772 005021                CLR    (R1)+
91 003774 012721 000000G          MOV    #TRP34,(R1)+   ; TRAP
92 004000 005021                CLR    (R1)+
93 004002 012737 000000G 000114  MOV    #MEMPAR,@#114 ; MEMORY PARITY TRAP
94 004010 010037 000116          MOV    R0,@#116
95 004014 012737 000000G 000244  MOV    #FPTRAP,@#244 ; TRAP 244 -- FLOATING POINT TRAP
96 004022 010037 000246          MOV    R0,@#246      ; Enter FPU trap at priority 7
97 004026 012737 000000G 000250  MOV    #TRP250,@#250 ; TRAP 250 -- MEMORY MANAGEMENT TRAP
98 004034 005037 000252          CLR    @#252
99                                ;
100                               ; Initialize the system mapped region.
101                               ;
102 004040 010546                MOV    R5,-(SP)        ; SAVE THE CURRENT CONTENTS OF R5
103 004042 012705 000006          MOV    #6,R5          ; INITIALIZE TO THE FIRST REGION
104 004046 004737 000000G          CALL   MAPSYS        ; CALL THE SYSTEM MAPPING ROUTINE
105 004052 012605                MOV    (SP)+,R5        ; RESTORE THE PREVIOUS CONTENTS OF R5
106                               ;
107                               ; Set up Unibus mapping if needed
108                               ;
109 004054 004737 004732'          CALL   SETUMP        ; SET UP UNIBUS MAPPING
110                               ;
111                               ; Initialize time-sharing lines.
112                               ;
113 004060 004737 005350'          CALL   LININI        ; INIT LINES & SET UP VECTORS
114                               ;

```

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

```

115          ; Enable memory management
116          ; (The kernel-mode mapping registers are already set up)
117          ;
118 004064 052737 000000G 000000G      BIS      #MMENBL, @SR0MMR ; Turn on memory management
119 004072 105737 000000G              TSTB      SR3FLG          ; Does machine have memory management reg 3?
120 004076 001415                      BEQ       4$              ; Br if register does not exist (no ext. mem.)
121 004100 023727 000000G 010000      CMP       PHYMEM, #4096.    ; Does machine have at least 256Kb phys memory?
122 004106 103411                      BLO       4$              ; Br if not
123 004110 052737 000000G 000000G      BIS      #EMMAP, @SR3MMR ; Set extended memory on
124 004116 105737 000000G              TSTB      MEM256          ; Will TSX-Plus use at least 256Kb?
125 004122 001403                      BEQ       4$              ; Br if not
126 004124 052737 000000G 000000G      BIS      #IOMAP, @SR3MMR ; Turn on 22-bit memory management for I/O
127          ;
128          ; Initialize the memory allocation table
129          ;
130 004132 013737 000000G 000000G 4$:    MOV      MAPPAR, @KPAR5 ; Map to memory allocation table
131 004140 013702 000000G              MOV      LOMAP, R2          ; Point to 1st user-page entry
132 004144 105022                      8$:    CLRB      (R2)+        ; Say page is free
133 004146 020237 000000G              CMP      R2, HIMAP          ; Done all user pages?
134 004152 103774                      BLO      8$              ; Loop if not
135 004154 112712 000000G              MOVB     #MA$SYS, (R2)      ; Set flag marking start of system pages
136          ;
137          ; Set up I/O device interrupt vectors.
138          ;
139 004160 004737 005022'              CALL     DEVVEC            ; SET UP DEVICE INTERRUPT VECTORS
140          ;
141          ; If we are running on a Professional, initialize the PI handler
142          ;
143          .IF      NE, PROASM
144              TSTB      PROFLG          ; Are we running on a Professional?
145              BEQ       3$              ; Br if not
146              CALL      PROHAN          ; Initialize the PI handler
147              CALL      PIDVEN          ; Make device table entry for PI
148          .ENDC      ; NE, PROASM
149          ;
150          ; Initialize interrupt stack area
151          ;
152 004164 013702 000000G      3$:    MOV      INTSND, R2          ; Point to base of stack area
153 004170 012700 123456      MOV      #123456, R0              ; Get initialization value
154 004174 010022      12$:    MOV      R0, (R2)+                ; Initialize the interrupt stack area
155 004176 020237 000000G      CMP      R2, INTSTK              ; Finished?
156 004202 103774                      BLO      12$              ; Loop if not
157 004204 112737 177777 000000G      MOVB     #-1, STKLVL        ; Say we are not running on interrupt stack
158          ;
159          ; Enter TSEXEC to complete initialization
160          ;
161 004212 000137 000000G              JMP      INIIMP            ; ENTER INITIALIZATION ROUTINE IN TSEXEC
162          ;
163          ; Abort the initialization
164          ;
165 004216 013737 000042' 000000G INISTP: MOV      CLK100, @CLKVEC ; Restore RT-11 clock vector
166 004224 013737 000044' 000004      MOV      RTTRP4, @#4        ; Restore trap 4 vector
167 004232 013737 000046' 000000G      MOV      RTMNVC, @#RMON    ; Restore RT-11 monitor pointer
168 004240      9$:    .EXIT                                ; RETURN TO RT-11

```

```

1          .SBTTL  LODINI -- Load a segment over TSINIT
2          ;-----
3          ; LODINI is called to read an overlay segment over TSINIT.
4          ;
5          ; Inputs:
6          ;   R2 = Pointer to OSTABL entry for segment to be loaded.
7          ;   R5 = 64-byte block # where segment is to be loaded.
8          ;
9 004242  010146  LODINI: MOV     R1,-(SP)
10 004244  010346      MOV     R3,-(SP)
11 004246  010446      MOV     R4,-(SP)
12          ;
13          ; Get pointer to linker-built overlay entry
14          ;
15 004250  016201  000004      MOV     OS$OVL(R2),R1  ;Get pointer to linker-built table
16          ;
17          ; Determine how much code to read from the segment
18          ;
19 004254  016204  000000      MOV     OS$SIZ(R2),R4  ;Get # 64-byte blks allocated for segment
20 004260  072427  000005      ASH     #5,R4          ;Convert to # words
21 004264  020461  000000G     CMP     R4,O.SIZ(R1)  ;Compare with original segment code size
22 004270  101402      BLOS     1$          ;Br if segment was truncated by init
23 004272  016104  000000G     MOV     O.SIZ(R1),R4  ;Get code size
24          ;
25          ; Read the segment into memory
26          ;
27 004276  010503  1$:      MOV     R5,R3          ;Get 64-byte block #
28 004300  072327  000006      ASH     #6,R3          ;Convert to byte address
29 004304      .READW  #AREA,#17,R3,R4,O.BLK(R1)
30 004340  103406      BCS     10$          ;Br if error on read
31          ;
32          ; Store the physical address of the segment into the overlay descriptor
33          ;
34 004342  010561  000000G     MOV     R5,O.PAR(R1)  ;Remember physical address of segment
35          ;
36          ; Finished
37          ;
38 004346  012604      MOV     (SP)+,R4
39 004350  012603      MOV     (SP)+,R3
40 004352  012601      MOV     (SP)+,R1
41 004354  000207      RETURN
42          ;
43          ; Error on read
44          ;
45 004356  10$:      .PRINT  #TSXHD
46 004364      .PRINT  #RDERR
47 004372      .EXIT

```

INIOVL -- Load system overlays over TSINIT

```

1          .SBTTL  INIOVL -- Load system overlays over TSINIT
2          ;-----
3          ; INIOVL is called to load into memory those system overlays that
4          ; are to be placed over the TSINIT code.
5          ;
6          ; Inputs:
7          ; Overlay segment information is in OSTABL.
8          ;
9 004374 010246 INIOVL: MOV     R2,-(SP)
10 004376 010546      MOV     R5,-(SP)
11          ;
12          ; Initialize pointer to start of memory area for overlays
13          ;
14 004400 013705 000140'      MOV     OVLBAS,R5      ; Start of area for overlays
15 004404 072527 177772      ASH     #-6,R5        ; Convert to 64-byte #
16 004410 042705 176000      BIC     #176000,R5     ; Clear possible propagated sign bits
17          ;
18          ; Begin loop to load each overlay that goes over TSINIT
19          ;
20 004414 012702 000576'      MOV     #OSTABL,R2     ; Point to 1st overlay segment entry
21 004420 005762 000002      1$:  TST     OS$FLG(R2)   ; Does this segment go over TSINIT?
22 004424 001406              BEQ     2$             ; Br if not
23 004426 004737 004604'      CALL    KEYSEG        ; Remember base of some segments
24 004432 004737 004242'      CALL    LODINI        ; Load the segment
25 004436 066205 000000      ADD     OS$SIZ(R2),R5   ; Advance memory pointer
26 004442 062702 000006      2$:  ADD     #OS$$SZ,R2   ; Point to entry for next segment
27 004446 020237 001024'      CMP     R2,OSLAST     ; Finished all segments?
28 004452 103762              BLO     1$            ; Loop if not
29          ;
30          ; Initialize entry point vector for TSTIOX segment
31          ;
32 004454 013702 000000G      MOV     TIOBAS,R2     ; Get addr of base of TSTIOX
33 004460 072227 000006      ASH     #6,R2         ; Convert to byte address
34 004464 012705 000000G      MOV     #TIOVEC,R5    ; Point to entry point vector
35 004470 004737 004546'      CALL    ENTVEC        ; Set up entry point vector
36          ;
37          ; Initialize entry point vector for TSCASH segment
38          ;
39 004474 013702 000000G      MOV     CSHBAS,R2     ; Get addr of base of TSCASH
40 004500 001406              BEQ     3$             ; Br if TSCASH not loaded
41 004502 072227 000006      ASH     #6,R2         ; Convert to byte address
42 004506 012705 000000G      MOV     #CSHVEC,R5    ; Point to entry point vector
43 004512 004737 004546'      CALL    ENTVEC        ; Set up entry point vector
44          ;
45          ; Initialize entry point vector for TSLOCK segment
46          ;
47 004516 013702 000000G      3$:  MOV     LOKBAS,R2   ; Get addr of base of TSLOCK
48 004522 001406              BEQ     9$             ; Br if TSLOCK not loaded
49 004524 072227 000006      ASH     #6,R2         ; Convert to byte address
50 004530 012705 000000G      MOV     #LOKVEC,R5    ; Point to entry point vector
51 004534 004737 004546'      CALL    ENTVEC        ; Set up entry point vector
52          ;
53          ; Finished
54          ;
55 004540 012605      9$:  MOV     (SP)+,R5
56 004542 012602      MOV     (SP)+,R2
57 004544 000207      RETURN

```

ENTVEC -- Set up entry point vector for overlay

```

1          .SBTTL  ENTVEC -- Set up entry point vector for overlay
2          ;-----
3          ; ENTVEC is called to set up addresses in an entry point vector for
4          ; overlay segments such as TSCASH that are loaded at addresses different
5          ; from where they are linked.
6          ;
7          ; Inputs:
8          ;   R2 = Address of base of segment.
9          ;   R5 = Pointer to vector that is to be initialized (word with -1 terminates)
10         ;
11 004546 010246  ENTVEC: MOV      R2,-(SP)
12 004550 010446      MOV      R4,-(SP)
13 004552 010546      MOV      R5,-(SP)
14 004554 010204      MOV      R2,R4          ; Get addr of base of segment
15 004556 062704 000004  ADD      #4,R4      ; Point to start of vector in segment
16 004562 005722      TST      (R2)+        ; Get value to use to relocate offsets
17 004564 012415 1$:  MOV      (R4)+,(R5)    ; Get offset to entry point within segment
18 004566 060225      ADD      R2,(R5)+     ; Convert to absolute address
19 004570 005715      TST      (R5)         ; Any more words to initialize?
20 004572 001774      BEQ      1$           ; Br if yes
21          ;
22          ; Finished
23          ;
24 004574 012605      MOV      (SP)+,R5
25 004576 012604      MOV      (SP)+,R4
26 004600 012602      MOV      (SP)+,R2
27 004602 000207      RETURN

```

KEYSEG -- Remember memory position of system overlays

```

1          .SBTTL  KEYSEG -- Remember memory position of system overlays
2          ;-----
3          ; KEYSEG is called to remember the physical memory position of some
4          ; key system overlay segments.
5          ;
6          ; Inputs:
7          ;   R2 = Pointer to segment entry in OSTABL overlay table.
8          ;   R5 = Base 64-byte block physical memory for segment.
9          ;
10         004604  010446  KEYSEG: MOV      R4, -(SP)
11         11
12         12          ; Get the name of the segment out of the linker-built segment block
13         13          ;
14         14 004606  016200  000004      MOV      OS$OVL(R2), R0      ; Point to linker-built entry
15         15 004612  016004  000000G     MOV      0. ADR(R0), R4      ; Get the name of the segment
16         16          ;
17         17          ; See if this is a segment whose address we want to remember
18         18          ;
19         19 004616  020437  000156'      CMP      R4, R5MSG          ; Is this the TSMSG segment?
20         20 004622  001003                BNE      1$                ; Br if not
21         21 004624  010537  000000G     MOV      R5, MSGBAS          ; Remember base of TSMSG segment
22         22 004630  000436                BR      8$
23         23 004632  020437  000160'      1$: CMP      R4, R5WIN          ; Is this the TSWIN segment?
24         24 004636  001003                BNE      3$                ; Br if not
25         25 004640  010537  000000G     MOV      R5, WINBAS          ; Remember base of TSWIN segment
26         26 004644  000430                BR      8$
27         27 004646  020437  000164'      3$: CMP      R4, R5USR          ; Is this the TSUSR segment?
28         28 004652  001003                BNE      4$                ; Br if not
29         29 004654  010537  000000G     MOV      R5, USRBAS          ; Remember base of TSUSR segment
30         30 004660  000422                BR      8$
31         31 004662  020437  000162'      4$: CMP      R4, R5LOK          ; Is this the TSLOCK segment?
32         32 004666  001003                BNE      5$                ; Br if not
33         33 004670  010537  000000G     MOV      R5, LOKBAS          ; Remember base of TSLOCK segment
34         34 004674  000414                BR      8$
35         35 004676  020437  000166'      5$: CMP      R4, R5CSH          ; Is this the TSCASH segment?
36         36 004702  001003                BNE      6$                ; Br if not
37         37 004704  010537  000000G     MOV      R5, CSHBAS          ; Remember base of TSCASH segment
38         38 004710  000406                BR      8$
39         39 004712  020437  000170'      6$: CMP      R4, R5TIO          ; Is this the TSTIOX segment?
40         40 004716  001003                BNE      8$                ; Br if not
41         41 004720  010537  000000G     MOV      R5, TIOBAS          ; Remember base of module
42         42 004724  000400                BR      8$
43         43          ;
44         44          ; Finished
45         45          ;
46         46 004726  012604      8$: MOV      (SP)+, R4
47         47 004730  000207                RETURN

```

```

1      . IF      NE,<PROASM-1>    ; Assemble for PDP-11
2      . SBTTL   SETUMP -- Set up Unibus mapping if needed
3      ; -----
4      ; SETUMP is called to set up the Unibus map registers for 11/44 and
5      ; 11/70 systems which more than 256Kb of memory.
6      ; If Unibus mapping is needed, the Unibus map registers # 0-4 are
7      ; initialized for a 1-to-1 mapping with the low 40Kb of memory
8      ; so that I/O to system buffers in the low memory area can be done without
9      ; having to do Unibus mapping.
10     ;
11     ; Outputs:
12     ;   UBUSMP:  1==>Do Unibus mapping;  0==>Don't do Unibus mapping.
13     ;
14 004732 010246      SETUMP: MOV      R2,-(SP)
15 004734 010346      MOV      R3,-(SP)
16 004736 013746 000004      MOV      @#4,-(SP)      ; SAVE TRAP VECTOR
17 004742 012737 005010' 000004      MOV      #9@,@#4      ; CATCH TRAPS
18     ;
19     ; See if this is a type of maching that needs unibus mapping
20     ;
21 004750 105737 000000G      TSTB     UBUSMP      ; Is UNIBUS mapping needed?
22 004754 001415      BEQ      9$      ; Br if not
23     ;
24     ; Unibus mapping is needed
25     ; Load unibus map registers # 0-4 to point to low 48Kb of memory.
26     ;
27 004756 012705 000000G 2$:      MOV      #UMRADR,R5      ; POINT TO CONTROL REGISTER FOR UNIBUS MAP 0
28 004762 005004      CLR      R4      ; START MAPPING TO BOTTOM OF MEMORY
29 004764 012700 000005      MOV      #5,R0      ; LOAD 5 MAP REGISTERS
30 004770 010425      1$:      MOV      R4,(R5)+      ; SET LOW-ORDER VALUE IN MAP REGISTER
31 004772 005025      CLR      (R5)+      ; CLEAR HIGH-ORDER VALUE IN MAP REGISTER
32 004774 062704 020000      ADD      #8192.,R4      ; ADVANCE MEMORY ADDRESS
33 005000 077005      SOB      R0,1$      ; LOOP IF MORE MAP REGISTERS TO LOAD
34     ;
35     ; Turn on Unibus mapping
36     ;
37 005002 052737 000000G 000000G      BIS      #IOMAP,@#SR3MMR ; ENABLE UNIBUS MAPPING
38     ;
39     ; Finished
40     ;
41 005010 012637 000004      9$:      MOV      (SP)+,@#4      ; RESTORE TRAP VECTOR
42 005014 012605      MOV      (SP)+,R5
43 005016 012604      MOV      (SP)+,R4
44 005020 000207      RETURN
45     . IFF      ; NE,<PROASM-1>    ; Following code for Pro-only assembly
46     ;
47     ; Define dummy SETUMP routine for Pro
48     ;
49 SETUMP: RETURN
50     . ENDC      ; NE,<PROASM-1>

```

DEVVEC -- Set up device vectors

```

1          .SBTTL  DEVVEC -- Set up device vectors
2          ;-----
3          ; DEVVEC is called to set up device interrupt vectors for handlers
4          ; that have been loaded.
5          ;
6 005022 010146      DEVVEC: MOV      R1,-(SP)
7 005024 010346      MOV      R3,-(SP)
8 005026 010546      MOV      R5,-(SP)
9 005030 013746 0000000  MOV      @#KPAR5,-(SP) ; Save PAR 5 mapping
10
11          ; Begin loop to set up vectors for each device
12          ;
13 005034 012701 0000002      MOV      #2,R1 ; Get index # of 1st device after TT
14 005040 016103 0000000 1$: MOV      HANENT(R1),R3 ; Get handler entry point address
15 005044 020327 0000006      CMP      R3,#6 ; Is this a real device?
16 005050 101436      BLOS      6$ ; Br if not
17
18          ; See if we need to map PAR 5 to this handler
19          ;
20 005052 016100 0000000      MOV      HANPAR(R1),R0 ; Get PAR 5 base for this handler
21 005056 001402      BEQ      2$ ; Br if this is not a mapped handler
22 005060 010037 0000000      MOV      R0,@#KPAR5 ; Map PAR 5 to the handler
23
24          ; Clear CGE and LGE in handler header
25          ;
26 005064 005023      2$: CLR      (R3)+ ; Clear LGE (4th word in handler)
27 005066 005013      CLR      (R3) ; Clear CGE (5th word in handler)
28 005070 162703 0000010      SUB      #10,R3 ; Point to 1st word of handler
29
30          ; Set up interrupt vectors for this handler
31          ;
32 005074 005005      CLR      R5 ; Assume vector base address is 0
33 005076 005713      TST      (R3) ; Any vectors to set up?
34 005100 001422      BEQ      6$ ; Br if no vectors to set up
35 005102 002403      BLT      5$ ; Br if multiple-vector
36 005104 004737 005174'      CALL     SETVEC ; Set up the vector
37 005110 000416      BR      6$ ; Finished
38
39          ; Multiple vectors.
40          ;
41 005112 012300      5$: MOV      (R3)+,R0 ; Get offset to list of vector info
42 005114 006300      ASL      R0 ; Get byte offset to vector list
43 005116 060003      ADD      R0,R3 ; Get absolute address of vector table
44 005120 005713      TST      (R3) ; Is this a PRO device with floating vectors?
45 005122 002005      BGE      7$ ; Br if not
46 005124 005723      TST      (R3)+ ; Point to word with device ID
47 005126 012346      MOV      (R3)+,-(SP) ; Get device ID
48 005130 004777 0000000      CALL     @RPRVEC ; Get base vector location for device
49 005134 012605      MOV      (SP)+,R5 ; This is base of vector locations
50 005136 004737 005174'      7$: CALL     SETVEC ; Set up the vector
51 005142 005713      TST      (R3) ; Any more vectors to set up?
52 005144 003374      BGT      7$ ; Br if yes
53
54          ; See if there are more devices to set up.
55          ;
56 005146 062701 0000002      6$: ADD      #2,R1 ; Advance device table index
57 005152 020137 0000000      CMP      R1,NUMDEV ; More to do?

```

```
58 005156 101730          BLOS      1$          ;Br if yes
59                          ;
60                          ; Finished
61                          ;
62 005160 012637 00000000      MOV      (SP)+, @#KPAR5
63 005164 012605      MOV      (SP)+, R5
64 005166 012603      MOV      (SP)+, R3
65 005170 012601      MOV      (SP)+, R1
66 005172 000207      RETURN
```

SETVEC -- Set up an interrupt vector for a device

```

1          .SBTTL SETVEC -- Set up an interrupt vector for a device
2          ;-----
3          ; SETVEC is called to set up one interrupt vector for a device.
4          ;
5          ; Inputs:
6          ;   R1 = Device index number.
7          ;   R3 = Pointer into device handler to 3 word cells:
8          ;       1. Address of interrupt vector.
9          ;       2. Offset to interrupt entry point in handler.
10         ;       3. PS for interrupt.
11         ;   R5 = Base address to add to vector locations.
12         ;
13         ; Outputs:
14         ;   R3 = Points beyond 3 word info block in handler.
15         ;
16         ; Size of interrupt catching routine compiled for interrupts to
17         ; mapped handlers:
18         ;
19         ; MPIVSZ = 26. ; Amt of code compiled for mapped ints
20         ;
21         ; SETVEC: MOV R4, -(SP)
22         ;
23         ; See if this is a mapped handler
24         ;
25         ; 005176 005761 00000006
26         ; 005202 001010
27         ;
28         ; TST HANPAR(R1) ; Is this a mapped handler
29         ; BNE 1$ ; Br if yes
30         ;
31         ; This is an unmapped handler.
32         ; Vector interrupts directly to the handler.
33         ;
34         ; MOV (R3)+, R0 ; Get address of interrupt vector
35         ; ADD R5, R0 ; Add base address to vector location
36         ; MOV R3, (R0) ; Store address of cell in handler
37         ; ADD (R3)+, (R0)+ ; Add offset to interrupt entry point
38         ; MOV (R3)+, (R0) ; Set PS for interrupt
39         ; BIS #340, (R0) ; Make sure priority = 7
40         ; BR 9$
41         ;
42         ; This is a mapped handler.
43         ; Vector the interrupt to a routine that performs the following functions:
44         ; 1. Save the current PAR 5 mapping.
45         ; 2. Map PAR 5 to the handler.
46         ; 3. Push a dummy PC and PS on stack that will send return from handler
47         ; to a routine that will restore the PAR 5 mapping.
48         ; 4. Jump into the handler interrupt entry point.
49         ;
50         ; 1$: MOV XMVBAS, R4 ; Point to area where we store interrupt rtn
51         ; MOV (R3)+, R0 ; Get address of interrupt vector
52         ; ADD R5, R0 ; Add base address to interrupt location
53         ; MOV R4, (R0)+ ; Direct interrupt to our routine
54         ; MOV #013746, (R4)+ ; [ MOV @#KPAR5, -(SP) ]
55         ; MOV #KPAR5, (R4)+
56         ; MOV #012737, (R4)+ ; [ MOV #par5val, @#KPAR5 ]
57         ; MOV #012737, (R4)+
58         ; MOV HANPAR(R1), (R4)+
59         ; MOV #KPAR5, (R4)+
60         ; MOV #012746, (R4)+ ; [ MOV #340, -(SP) ]
61         ; MOV #340, (R4)+

```

```

58 005272 012724 012746      MOV      #012746,(R4)+    ; [ MOV #HANXIT,-(SP) ]
59 005276 012724 000000G      MOV      #HANXIT,(R4)+
60 005302 012724 000257      MOV      #000257,(R4)+    ; [ CCC - Clear all condition codes ]
61 005306 016314 000002      MOV      2(R3),(R4)        ; [ SEx - Set condition codes specified in PS]
62 005312 042714 177760      BIC      #^C17,(R4)
63 005316 052724 000260      BIS      #260,(R4)+
64 005322 012724 000137      MOV      #000137,(R4)+    ; [ JMP @#handler_entry ]
65 005326 010314              MOV      R3,(R4)          ; Store address of int entry point
66 005330 062324              ADD      (R3)+,(R4)+
67 005332 012310              MOV      (R3)+,(R0)        ; Set PS for interrupt entry
68 005334 052710 000340      BIS      #340,(R0)          ; Make sure priority = 7
69                               ;
70                               ; Save address beyond end of compiled interrupt catcher routine
71                               ;
72 005340 010437 000122'      MOV      R4,XMVBAS        ; Save address beyond end of routine
73                               ;
74                               ; Finished
75                               ;
76 005344 012604              9$:    MOV      (SP)+,R4
77 005346 000207              RETURN

```

SETVEC -- Set up an interrupt vector for a device

```

1      .IF      NE,PROASM
2          .SBTTL  PIDVEN -- Make device table entry for PI device
3          ;-----
4          ; If we are running on a Professional computer, PIDVEN is called to
5          ; make an entry in the device tables for the PI device.
6          ;
7      PIDVEN: MOV      R1,-(SP)
8          ;
9          ; Increase number of defined devices and get device table entry index
10         ; to use for the PI device.
11         ;
12         ADD      #2,NUMDEV      ; One more device
13         MOV      NUMDEV,R1      ; Get device table index
14         ;
15         ; Set up information about the PI device
16         ;
17         MOV      R5OPI,PNAME(R1) ; Set device name
18         MOV      #<DS$SFN!DI$PI>,DVSTAT(R1) ; Set device status flags
19         CLR      DVFLAG(R1)      ; Clear other flags
20         CLR      DEVSIZ(R1)      ; Clear device size
21         MOV      #PIHAN+6,HANENT(R1); Set handler entry point (4th word)
22         MOV      #PROSIZ,HANSIZ(R1) ; Set handler size
23         ;
24         ; Finished
25         ;
26         MOV      (SP)+,R1
27         RETURN
28     .ENDC      ; NE,PROASM

```

SETVEC -- Set up an interrupt vector for a device

```

1          .IF      NE,<PROASM-1>    ;If assembling for PDP-11
2          .SBTTL   LININI -- Initialize time-sharing lines
3          ;-----
4          ; LININI is called to initialize the time-sharing lines.
5          ; This consists of setting up interrupt vectors and setting control
6          ; flags in the status registers.
7          ;
8 005350 010146  LININI: MOV      R1,-(SP)
9 005352 010246      MOV      R2,-(SP)
10 005354 010346      MOV      R3,-(SP)
11 005356 010446      MOV      R4,-(SP)
12 005360 010546      MOV      R5,-(SP)
13          ;
14          ; Set up interrupt vectors for DL11 lines
15          ;
16 005362 012701 000002      MOV      #2,R1          ;Index for 1st line
17 005366 012704 000340      MOV      #340,R4        ;Priority 7 PS
18 005372 032761 000000G 000000G 1$: BIT      $$DEAD,LSW3(R1) ;Is this line uninstalled?
19 005400 001027      BNE      B$                    ;Br if yes
20 005402 032761 000000G 000000G      BIT      $$HARD,LSW3(R1) ;Is this line connected to hardware?
21 005410 001423      BEQ      B$                    ;Br if not
22 005412 026127 000000G 000000G      CMP      LCDTYP(R1),#CDX$DL ;Is this a DL11 line?
23 005420 001017      BNE      B$                    ;Br if not
24          ;
25          ; DL-11 line
26          ;
27 005422 016105 000000G      MOV      INVEC(R1),R5    ;GET ADDRESS OF INPUT VECTOR
28 005426 012702 000000G      MOV      #INRECV,R2     ;END OF RECEIVING VECTOR
29 005432 012703 177772G      MOV      #<OTRECV-6>,R3  ;START OF INPUT INTERRUPT ENTRY POINTS
30 005436 010100      MOV      R1,R0                  ;GET LINE NUMBER
31 005440 006300      ASL      R0                      ;4 BYTES PER INPUT INTERRUPT ENTRY POINT
32 005442 160002      SUB      R0,R2                  ;GET ADDRESS OF INPUT INTERRUPT ENTRY POINT
33 005444 060100      ADD      R1,R0                  ;6 BYTES PER OUTPUT INTERRUPT ENTRY POINT
34 005446 060003      ADD      R0,R3                  ;GET ADDRESS OF OUTPUT INTERRUPT ENTRY POINT
35 005450 010225      MOV      R2,(R5)+              ;SET PC FOR INPUT INTERRUPT ENTRY POINT
36 005452 010425      MOV      R4,(R5)+              ;SET PS FOR INPUT INTERRUPT
37 005454 010325      MOV      R3,(R5)+              ;SET PC FOR OUTPUT INTERRUPT
38 005456 010425      MOV      R4,(R5)+              ;SET PS FOR OUTPUT INTERRUPT
39          ;
40          ; Try next line
41          ;
42 005460 062701 000002  B$: ADD      #2,R1          ;ADVANCE LINE INDEX NUMBER
43 005464 020127 000000G  CMP      R1,#LSTHL         ;MORE TO DO?
44 005470 101740      BLOS     1$                    ;BR IF YES
45          ;
46          ; Initialize multiplexers.
47          ;
48 005472 012701 000000G  SETMUX: MOV      #LSTMX,R1    ;Get last mux index #
49 005476 001423      BEQ      SETLIN                ;Br if there are no mux lines
50 005500 026127 000000G 000000G 3$: CMP      MXTYPE(R1),#CDX$DZ ;Is this a DZ11, DH11, or DHV11?
51 005506 001412      BEQ      1$                    ;Br if DZ11
52 005510 026127 000000G 000000G      CMP      MXTYPE(R1),#CDX$VH ;Is this a DHV11?
53 005516 001003      BNE      2$                    ;Br if not
54 005520 004737 006270'    CALL      VHINIT          ;Initialize a DHV11
55 005524 000405      BR        4$
56 005526 004737 006220'    2$: CALL      DHINIT          ;Initialize a DH11
57 005532 000402      BR        4$

```

```

58 005534 004737 006030' 1$: CALL DZINIT ; Initialize a DZ11
59 005540 162701 000002 4$: SUB #2,R1 ; More to enable?
60 005544 001355 BNE 3$ ; Br if yes
61 ;
62 ; Enable all lines
63 ;
64 005546 012701 000000G SETLIN: MOV #LSTHL,R1 ; INDEX # OF LAST REAL LINE
65 005552 032761 000000G 000000G 4$: BIT ##DEAD,LSW3(R1) ; IS THIS LINE INSTALLED?
66 005560 001057 BNE 2$ ; BR IF NOT
67 005562 032761 000000G 000000G BIT ##HARD,LSW3(R1) ; Is this line connected to hardware?
68 005570 001453 BEQ 2$ ; Br if not
69 005572 032761 000000G 000000G BIT ##PHONE,ILSW2(R1) ; IS THIS A DIAL-UP LINE?
70 005600 001403 BEQ 3$ ; BR IF NOT
71 005602 052761 000000G 000000G BIS ##NOIN,LSW3(R1) ; IGNORE INPUT TILL DIAL UP OCCURS
72 005610 016105 000000G 3$: MOV LCDTYP(R1),R5 ; Get comm device type code
73 005614 016100 000000G MOV LMXNUM(R1),R0 ; IS THIS A DL-11 OR MUX LINE?
74 005620 001423 BEQ 1$ ; BR IF DL-11
75 005622 020527 000000G CMP R5,#CDX$DZ ; Is this a DZ11 or DH11?
76 005626 001411 BEQ 6$ ; Br if DZ11
77 005630 020527 000000G CMP R5,#CDX$VH ; Is this a DH11 or DHV11?
78 005634 001403 BEQ 7$ ; Br if DHV11
79 ;
80 ; DH11 line
81 ;
82 005636 004737 005742' CALL DHLPRM ; Set line parameters for DH11 line
83 005642 000426 BR 2$
84 ;
85 ; DHV11 line
86 ;
87 005644 004737 006000' 7$: CALL VHLPRM ; Set line parameters for DHV11 line
88 005650 000423 BR 2$
89 ;
90 ; DZ-11 line
91 ;
92 005652 016102 000000G 6$: MOV LMXLN(R1),R2 ; Get line # within mux group
93 005656 052702 017030 BIS #017030,R2 ; Set line enable flags
94 005662 010270 000000G MOV R2,@MXLPR(R0) ; Enable the line
95 005666 000414 BR 2$
96 ;
97 ; DL-11 line
98 ;
99 005670 016102 000000G 1$: MOV TSR(R1),R2 ; ADDRESS OF TRANSMITTER STATUS REGISTER
100 005674 011203 MOV (R2),R3 ; CLEAR TRANSMITTER STATUS REGISTER
101 005676 005012 CLR (R2)
102 005700 016102 000000G MOV RBR(R1),R2 ; ADDRESS OF RECEIVER BUFFER REGISTER
103 005704 011203 MOV (R2),R3 ; CLEAR RECEIVER BUFFER REGISTER
104 005706 016102 000000G MOV RSR(R1),R2 ; ADDRESS OF RECEIVER STATUS REGISTER
105 005712 005012 CLR (R2)
106 005714 012712 000000G MOV #RDINT,(R2) ; ENABLE RECEIVER INTERRUPTS
107 ;
108 ; Do next line
109 ;
110 005720 162701 000002 2$: SUB #2,R1
111 005724 003312 BGT 4$
112 ;
113 ; Finished
114 ;

```

115	005726	012605	MOV	(SP)+, R5
116	005730	012604	MOV	(SP)+, R4
117	005732	012603	MOV	(SP)+, R3
118	005734	012602	MOV	(SP)+, R2
119	005736	012601	MOV	(SP)+, R1
120	005740	000207	RETURN	

DHLPRM -- Set line parameters for a DH11 line

```

1          .SBTTL  DHLPRM -- Set line parameters for a DH11 line
2          ;-----
3          ; DHLPRM is called to set up the line parameters for a DH11 line.
4          ;
5          ; Inputs:
6          ; R1 = Physical line index number.
7          ;
8 005742    DHLPRM:
9          ;
10         ; Enable DM11 for this line
11         ;
12 005742    016100  0000000  MOV     LMXNUM(R1),R0    ;Get mux index number
13 005746    005760  0000000  TST     DM$CSR(R0)       ;Does this DH11 have DM11 modem control?
14 005752    001411                BEQ     2$          ;Br if not
15 005754    142770  0000000 0000000  BICB    #MF$LIN,@DM$CSR(R0) ;Clear line # field in DM11 CSR
16 005762    156170  0000000 0000000  BISB    LMXLN(R1),@DM$CSR(R0);Select line of interest
17 005770    012770  0000000 0000000  MOV     #MF$LE,@DM$LSR(R0);Enable the line
18         ;
19         ; Finished
20         ;
21 005776    000207      2$:  RETURN

```

VHLPRM -- Set line parameter values for DHV11 line

```

1          .SBTTL  VHLPRM -- Set line parameter values for DHV11 line
2          ;-----
3          ; Set the line parameter values for a DHV11 line.
4          ;
5          ; Inputs:
6          ;   R1 = Physical line index number.
7          ;
8 006000    VHLPRM:
9          ;
10         ; Enable the line
11         ;
12 006000    016100  0000000    MOV     LMXNUM(R1),R0    ;Get mux index number
13 006004    042770  0000000 0000000    BIC     #VF$LIN,@VH$CSR(R0) ;Clear line # field in mux CSR
14 006012    156170  0000000 0000000    BISB    LMXLN(R1),@VH$CSR(R0) ;Set our line #
15 006020    012770  0000000 0000000    MOV     #<VF$RE>,@VH$LCR(R0) ;Enable the line
16         ;
17         ; Finished
18         ;
19 006026    000207          RETURN

```

DZINIT -- Initialize a DZ11 multiplexer

```

1          .SBTTL  DZINIT -- Initialize a DZ11 multiplexer
2          ;-----
3          ; DZINIT is called to initialize a DZ11 multiplexer.
4          ;
5          ; Inputs:
6          ; R1 = Mux index number.
7          ;
8 006030    DZINIT:
9          ;
10         ; See if this DZ11 is installed
11         ;
12 006030    005761 000000G          TST      MXCSR(R1)      ; Is this DZ-11 installed?
13 006034    001416          BEQ      4$                  ; Br if not
14         ;
15         ; Set up interrupt vector connections for this MUX
16         ;
17 006036    004737 006074'          CALL     MUXVEC          ; Set up interrupt vectors for this DZ11
18         ;
19         ; Start up the mux operation
20         ;
21 006042    052771 000000G 000000G          BIS      #ZCLR,@MXCSR(R1); Do master clear on DZ-11
22 006050    032771 000000G 000000G 1$:      BIT      #ZCLR,@MXCSR(R1); Wait for clear to finish
23 006056    001374          BNE      1$
24 006060    017100 000000G          2$:      MOV      @MXRBUF(R1),R0 ; Clear silo
25 006064    100775          BMI      2$
26 006066    105071 000000G          CLRB     @MXDTR(R1)      ; Disable all data sets
27         ;
28         ; Finished
29         ;
30 006072    000207          4$:      RETURN

```

```

1          .SBTTL  MUXVEC -- Set up interrupt vectors for a multiplexer
2          ;-----
3          ; MUXVEC is called to set up the interrupt vector connections for
4          ; a DZ11, DH11, or DHV11 multiplexer.
5          ;
6          ; Inputs:
7          ; R1 = Mux index number.
8          ;
9 006074 010246 MUXVEC: MOV      R2,-(SP)
10 006076 010346      MOV      R3,-(SP)
11 006100 010546      MOV      R5,-(SP)
12          ;
13          ; Set interrupt vector for mux
14          ;
15 006102 016105 000000G      MOV      MXVEC(R1),R5      ;Get address of input interrupt vector
16 006106 012702 000000G      MOV      #INMXV,R2        ;End of receiving vector
17 006112 012703 177772G      MOV      #<OTMXV-6>,R3      ;Output interrupt table
18 006116 010100      MOV      R1,R0                    ;Get mux index number
19 006120 006300      ASL      R0                        ;4 bytes per line in input int table
20 006122 160002      SUB      R0,R2                    ;Get address of input entry point
21 006124 060100      ADD      R1,R0                    ;6 bytes per mux in output entry point table
22 006126 060003      ADD      R0,R3                    ;Get address of output int entry point
23 006130 010225      MOV      R2,(R5)+                ;Set PC for input interrupt
24 006132 012725 000340      MOV      #340,(R5)+        ;Set PS for output interrupt
25 006136 010325      MOV      R3,(R5)+                ;Set PC for output interrupt
26 006140 012715 000340      MOV      #340,(R5)         ;Set PS for output interrupt
27          ;
28          ; Now store an instruction sequence of the form:
29          ;
30          ; JSR      R5,@#interrupt_routine
31          ; .WORD   mux_index
32          ;
33          ; to catch mux output interrupts and vector them to the interrupt routine.
34          ;
35 006144 012723 004537      MOV      #004537,(R3)+      ;JSR R5,@#x
36 006150 012700 000000G      MOV      #DZ0INT,R0        ;Assume this is a DZ11
37 006154 026127 000000G 000000G      CMP      MXTYPE(R1),#CDX$DZ ;Is this a DZ11?
38 006162 001410      BEQ      1$                        ;Br if yes
39 006164 012700 000000G      MOV      #VH0INT,R0        ;Assume this is a DHV11
40 006170 026127 000000G 000000G      CMP      MXTYPE(R1),#CDX$VH ;Is this a DHV11?
41 006176 001402      BEQ      1$                        ;Br if yes
42 006200 012700 000000G      MOV      #DH0INT,R0        ;Get interrupt routine for DH11's
43 006204 010023      1$:    MOV      R0,(R3)+          ;Store address of interrupt routine
44 006206 010113      MOV      R1,(R3)                  ;Store mux index number
45          ;
46          ; Finished
47          ;
48 006210 012605      9$:    MOV      (SP)+,R5
49 006212 012603      MOV      (SP)+,R3
50 006214 012602      MOV      (SP)+,R2
51 006216 000207      RETURN

```

```

1          .SBTTL  DHINIT -- Initialize a DH11 multiplexer
2          ;-----
3          ; DHINIT is called to initialize a DH11 multiplexer
4          ;
5          ; Inputs:
6          ;   R1 = Mux index number
7          ;
8 006220    DHINIT:
9          ;
10         ; See if this DH11 is installed
11         ;
12 006220    005761    000000G          TST     MH$SCR(R1)      ; Is this DH11 installed?
13 006224    001420          BEQ     9$              ; Br if not
14         ;
15         ; Connect interrupt vector to DH11
16         ;
17 006226    004737    006074'          CALL    MUXVEC          ; Set up interrupt vectors for DH11
18         ;
19         ; Clear the multiplexer
20         ;
21 006232    012771    000000G 000000G          MOV     #HF$MC,@MH$SCR(R1) ; Set the master-clear flag
22 006240    032771    000000G 000000G 1$:      BIT     #HF$MC,@MH$SCR(R1) ; Wait for the master clear to be completed
23 006246    001374          BNE     1$
24         ;
25         ; Clear the DM11 scanner
26         ;
27 006250    016100    000000G          MOV     DM$CSR(R1),R0    ; Is there an associated DM11?
28 006254    001404          BEQ     3$              ; Br if not
29 006256    012710    000000G          MOV     #MF$CS,(R0)      ; Clear the scanner
30 006262    052710    000000G          BIS     #MF$CM,(R0)      ; Clear the multiplexer
31 006266          3$:
32         ;
33         ; Finished
34         ;
35 006266          9$:
36 006266    000207          RETURN

```

```

1          .SBTTL  VHINIT -- Initialize a DHV11 multiplexer
2          ;-----
3          ; Perform initialization for a DHV11 mux.
4          ;
5          ; Inputs:
6          ;   R1 = Mux index number.
7          ;
8 006270    VHINIT:
9          ;
10         ; See if this DHV11 is installed
11         ;
12 006270    005761  0000000  TST      VH$CSR(R1)      ; Is this DHV11 installed?
13 006274    001414          BEQ      9$              ; Br if not
14         ;
15         ; Connect interrupt vector to DHV11
16         ;
17 006276    004737  006074'   CALL    MUXVEC          ; Set up interrupt vectors
18         ;
19         ; Clear the multiplexer
20         ;
21 006302    012771  0000000  0000000  MOV      #VF$MR,@VH$CSR(R1) ; Reset the multiplexer
22 006310    032771  0000000  0000000  1$:  BIT      #VF$MR,@VH$CSR(R1) ; Wait for reset to finish
23 006316    001374          BNE      1$
24         ;
25         ; Clean out the FIFO buffer in the mux
26         ;
27 006320    017100  0000000          2$:  MOV      @MXRBUF(R1),R0 ; Get contents of receiver buffer register
28 006324    002775          BLT      2$              ; Loop until RBUF empty
29         ;
30         ; Finished
31         ;
32 006326    000207          9$:  RETURN
  
```

```

1      ; -----
2      ; End of code that can be omitted for Pro-only systems
3      ;
4      . IFF      ; NE, <PROASM-1> ; Begin code for Pro only
5      ;
6      ; This code is assembled only for Pro systems.
7      ; T/S line init routines for Pro only.
8      ;
9      LINCHK:
10     DHLPRM:
11     VHLPRM:
12     DZINIT:
13     MUXVEC:
14     DHINIT:
15     VHINIT:
16         RETURN
17     ;
18     ; LININI routine for Pro systems
19     ;
20     LININI: MOV      R1, -(SP)
21             MOV      #LSTLIN, R1      ; Get # of last line
22     ;
23     ; Determine if this line is connected to hardware
24     ;
25     1$: CALL      LINTYP      ; Determine the type of this line
26         BIT      #$HARD, LSW3(R1) ; Is this line connected to hardware?
27         BEQ      2$          ; Br if not
28     ;
29     ; Call Pro line initialization routine
30     ;
31         CALL      PROLIN      ; Initialize Pro line
32     ;
33     ; Do some special init for phone lines
34     ;
35         BIT      #$PHONE, ILSW2(R1) ; Is this a dialup line?
36         BEQ      2$          ; Br if not
37         BIS      #$NOIN, LSW3(R1) ; Ignore input till dial up occurs
38     ;
39     ; Check next line
40     ;
41     2$: SUB      #2, R1      ; Get index # of next line
42         BGT      1$          ; Loop if more lines to do
43     ;
44     ; Finished
45     ;
46         MOV      (SP)+, R1
47         RETURN
48     ;
49     ; End of Pro-only code
50     ;
51     . ENDC      ; NE, <PROASM-1>

```

LINTYP -- Determine the type of a line

```

1          .SBTTL  LINTYP -- Determine the type of a line
2          ;-----
3          ; LINTYP is called to determine if the current line is a time-sharing line
4          ; a CL line, or a non-hardware connected line.
5          ;
6          ; Inputs:
7          ; R1 = Line index number
8          ;
9 006330 020127 000000G LINTYP: CMP R1,#LSTPL ;Is this a time-sharing line?
10 006334 101422 BLOS 1$ ;Br if yes
11 006336 020127 000000G CMP R1,#FSTIOL ;Is this a CL line?
12 006342 103004 BHIS 2$ ;Br if yes
13 006344 012761 177777 000000G MOV #-1,LCLUNT(R1) ;Say line not in use as a CL line
14 006352 000432 BR 9$
15          ;
16          ; This is a CL line
17          ;
18 006354 016100 000000G 2$: MOV LCLUNT(R1),R0 ;Get the CL unit index number
19 006360 010160 000000G MOV R1,CL$LIX(R0) ;Say which line is assoc with this CL unit
20 006364 012761 000000G 000000G MOV #CLOTIR,LOUTIR(R1) ;Set terminal output interrupt routine
21 006372 012761 000000G 000000G MOV #CLINCP,LINIR(R1) ;Set terminal input interrupt routine
22 006400 000414 BR 8$
23          ;
24          ; This is a time-sharing line
25          ;
26 006402 012761 177777 000000G 1$: MOV #-1,LCLUNT(R1) ;Say line is not in use as a CL unit
27 006410 012761 177777 000000G MOV #-1,LXCL(R1) ;Line is not cross-connected to CL unit
28 006416 012761 000000G 000000G MOV #NEDCHR,LOUTIR(R1) ;Set terminal output interrupt routine
29 006424 012761 000000G 000000G MOV #TTINCP,LINIR(R1) ;Set terminal input interrupt routine
30 006432 052761 000000G 000000G 8$: BIS #$HARD,LSW3(R1) ;This line is connected to hardware
31          ;
32          ; Finished
33          ;
34 006440 000207 9$: RETURN

```

* * * Initialization done with RT-11 running * * *

```

1          .SBTTL  * * * Initialization done with RT-11 running * * *
2          ;-----
3          ; Initialization at start of execution of TSX.
4          ;
5          ; The initialization done in this section uses the running RT-11 system
6          ; to perform functions for it.
7          ;
8 006442    INITGD:
9          ;
10         ; Save some RT-11 pointers in case we abort the initialization
11         ;
12 006442    013737    000004    000044'      MOV     @#4,RTTRP4      ; Save trap 4 vector
13 006450    013737    000000G 000046'      MOV     @#RMON,RTMNVG    ; RT-11 monitor pointer
14 006456    013737    000000G 000042'      MOV     @#CLKVEC,CLK100 ; Clock vector (defined in TSGEN at 100)
15         ;
16         ; Get the current time of day which we will use later to make sure
17         ; the line time clock is working.
18         ;
19 006464          .GTIM    #AREA,#SYTIMH    ; Get the current time of day
20         ;
21         ; Trap ^C for later test so we can restore clock vector
22         ;
23 006504          .SCCA    #AREA,#CCAFLG    ; Catch control-C
24         ;
25         ; Check for TSGEN size overflow
26         ;
27 006524    012700    000000G      MOV     #GENTOP,R0      ; Get top of TSGEN
28 006530    162700    037776      SUB     #<40000-2>,R0    ; Will TSKMON have problems?
29 006534    003422          BLE     15$      ; Continue if not
30 006536    010046      MOV     R0,-(SP)      ; Save overflow size
31 006540          .PRINT    #TSXHD          ; Print error message
32 006546          .PRINT    #TOOBIG
33 006554          .PRINT    #REDUCE
34 006562    012600      MOV     (SP)+,R0      ; Recover amount of overflow
35 006564    004737    027746'      CALL    PRTDEC
36 006570          .PRINT    #BYTES
37 006576    000137    004216'      JMP     INISTP
38         ;
39         ; Initialize the system stack (below 1000)
40         ;
41 006602    012701    000000G 15$: MOV     #SSEND,R1      ; Point to bottom of stack
42 006606    012700    123456      MOV     #123456,R0      ; Get initialization value
43 006612    010021 13$: MOV     R0,(R1)+      ; Initialize the stack
44 006614    020127    000000G      CMP     R1,#SS        ; Reached top of the stack area?
45 006620    103774          BLO     13$        ; Loop if not
46 006622    010106      MOV     R1,SP        ; Run on system stack
47         ;
48         ; Make sure we are not already running under TSX.
49         ;
50 006624          .SERR          ; DON'T DIE ON ERRORS
51 006632    012700    000242'      MOV     #GTLIN,R0      ; TSX EMT TO GET LINE NUMBER
52 006636    104375          EMT     375          ; TRY A TSX EMT
53 006640    103410          BCS     1$        ; BR IF NOT UNDER TSX
54 006642          .PRINT    #TSXHD          ; ALREADY UNDER TSX
55 006650          .PRINT    #TSXRUN
56 006656    000137    004216'      JMP     INISTP
57 006662 1$: .HERR          ; RENABLE FATAL ERRORS

```

* * * Initialization done with RT-11 running * * *

```

58      ;
59      ; Make sure this machine has memory management facilities.
60      ;
61 006670      .TRPSET #AREA,#NOXM      ; CATCH TRAPS
62 006710      005737 000000G      TST      @#SROMMR      ; TRY TO ACCESS MEMORY MANAGEMENT REGISTER
63 006714      .TRPSET #AREA,#0      ; Release trap control
64      ;
65      ; Request all available memory from RT-11.
66      ;
67 006732      .SETTOP #-2      ; REQUEST ALL AVAILABLE MEMORY
68 006740      010037 000132'      MOV      R0,TOPMEM      ; REMEMBER WHERE TOP OF MEMORY IS
69 006744      020027 000000G      CMP      R0,#VPAR5      ; TSX CANNOT EXTEND ABOVE PAR5 BASE ADDRESS
70 006750      101402      BLOS      3$      ; BR IF RT-11 IS BELOW THAT
71 006752      012700 000000G      MOV      #VPAR5,R0      ; SET PAR5 BASE AS UPPER LIMIT ON TSX SIZE
72 006756      010037 000302'      3$: MOV      R0,MEMLIM      ; TSX MAY NOT EXCEED THIS UPPER LIMIT
73      ;
74      ; Lock USR in memory for speed
75      ; (Set USR to swap over TSEMT to get out of the way)
76      ;
77 006762      012705 177776'      MOV      #TSINIT-2,R5      ; GET THE BASE OF TSINIT
78 006766      .GVAL      #AREA,#374      ; GET SIZE OF RT-11 USR MODULE
79 007006      160005      SUB      R0,R5      ; ALLOCATE SPACE BELOW TSINIT FOR USR
80 007010      010537 000046      MOV      R5,@#46      ; SET USR TO SWAP OVER TSEMT
81 007014      5$: .LOCK      ; LOCK USR IN MEMORY
82      ;
83      ; Determine if we are to run system with the system debugger
84      ;
85 007016      032737 000000G 000000G      BIT      #CHAIN,@#JSWLOC ; WERE WE CHAINED TO?
86 007024      001406      BEQ      10$      ; BR IF NOT
87 007026      023737 000510 000222'      CMP      @#510,R500DT      ; SHOULD WE RUN UNDER ODT?
88 007034      001002      BNE      10$      ; BR IF NOT
89 007036      005237 000034'      INC      ODTFLG      ; SET FLAG SAYING DEBUGGER WANTED
90      ;
91      ; Call Pro TSX initialization only if assembling for the Pro
92      ; Jump to INISTP if checking fails.
93      ;
94 007042      10$:
95      .IF      NE,PROCID      ; ** Do if assembling for pro only **
96      CALL      INSCHK      ; PERFORM VERIFICATION AND DECRYPTION FOR PRO
97      .ENDC      ; NE,PROCID
98      ;
99      ; Allocate non-initialized buffer space over TSINIT.
100      ;
101 007042      012705 000000'      MOV      #TSINIT,R5      ; Allocate buffer space over TSINIT
102 007046      004737 013006'      CALL      ALOCBF      ; Do allocation
103 007052      004737 013452'      CALL      ALCSLO      ; Allocate silo buffers for lines
104 007056      020527 006442'      CMP      R5,#INITG0      ; Are we beyond code that takes over control?
105 007062      103002      BHS      12$      ; Br if yes
106 007064      012705 006442'      MOV      #INITG0,R5      ; Advance up to initial code
107      ;
108      ; Allocate the interrupt stack over TSINIT
109      ; If we are running on a Pro, allocate buffer for the PI handler
110      ; initialization code over the interrupt stack area.
111      ;
112      PIINSZ      =      700.      ; Space needed for PI init code
113 007070      010537 000000G      12$: MOV      R5,INTSND      ; Ptr to base of interrupt stack
114 007074      062705 000002      ADD      #2,R5      ; Always leave last word of stack for flag val

```

* * * Initialization done with RT-11 running * * *

```

115 007100 013701 000000G      MOV      @#RMON,R1      ;Get pointer to RT-11 RMON base
116 007104 032761 000000G 000370  BIT      #CW$PRO,370(R1) ;Are we running on a PRO?
117 007112 001407          BEQ      11$              ;Br if not
118 007114 105237 000000G      INCB     PROFLG        ;Set flag saying this is a PRO-350
119 007120 010537 000150'      MOV      R5,PROBUF      ;Save pointer to buffer area
120 007124 062705 001274      ADD      #PIINSZ,R5      ;Allocate space for buffer
121 007130 000402          BR      14$
122 007132 062705 000000G      11$:  ADD      #INTSSZ,R5      ;Allocate space for interrupt stack
123 007136 010537 000000G      14$:  MOV      R5,INTSTK      ;Address of top of interrupt stack
124                                ;
125                                ; Allocate space for those overlays that go over TSINIT
126                                ;
127 007142 004737 023402'      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 007146 020527 030066'      CMP      R5,#INITOP      ;Have we allocated up to top of TSINIT?
133 007152 103002          BHS      4$              ;Br if yes
134 007154 012705 030066'      MOV      #INITOP,R5      ;Advance to top of TSINIT
135                                ;
136                                ; Allocate a 2048. byte work buffer
137                                ;
138 007160 004737 010266'      4$:  CALL     ALCWRK          ;Allocate work buffer
139                                ;
140                                ; Allocate empty Region Control Blocks for use by handlers
141                                ;
142 007164 004737 010322'      CALL     ALCHRB
143                                ;
144                                ; If we were started in debug mode, load ODT.
145                                ;
146                                ; . IF      EQ,PROCID      ;Don't allow ODT for production PRO version
147 007170 005737 000034'      TST      ODTFLG          ;Are we to load system debugger?
148 007174 001402          BEQ      2$              ;Br if not
149 007176 004737 026306'      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 007202 004737 016512'      2$:  CALL     MEMINI          ;Initialize memory management
156                                ;
157                                ; Extract information from RT-11 configuration and sysgen words.
158                                ;
159 007206 013701 000000G      MOV      @#RMON,R1      ;GET POINTER TO BASE OF RMON
160 007212 016102 000300      MOV      300(R1),R2      ;GET RT-11 CONFIGURATION WORD
161 007216 042702 000000C      BIC      #CW$GDH+CW$BTH+CW$LGS,R2 ;RESET A FEW FLAGS
162 007222 052702 000000C      BIS      #CW$FB+CW$FGJ+CW$USR+CW$XM,R2 ;SET A FEW FLAGS
163 007226 010237 000000G      MOV      R2,CONFIG      ;INITIALIZE OUR CONFIGURATION WORD
164                                ; Now get extended configuration word.
165 007232 016137 000370 000000G  MOV      370(R1),CONFIG2 ;EXTENDED CONFIGURATION WORD
166 007240 052737 000000G 000000G  BIS      #CW$ESP,CONFIG2 ;SET EXIT NO SWAP FLAG
167 007246 123727 000000G 000000G  CMPB     VBUSTP,#QBUS      ;Is this a Q-bus machine?
168 007254 001003          BNE      25$              ;Br if not
169 007256 052737 000000G 000000G  BIS      #CW$QBS,CONFIG2 ;Set QBUS flag
170                                ; And sysgen option word.
171 007264 016102 000372      25$:  MOV      372(R1),R2

```

* * * Initialization done with RT-11 running * * *

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

* * * Initialization done with RT-11 running * * *

```

1          ;
2          ; Set up a few clock constants based on clock frequency.
3          ; See if we have a 50 or 60 Hz clock
4          ;
5 007330 032737 000000G 000000G INICLK: BIT      #CW$50H,CONFIG ; 50 or 60 Hz clock?
6 007336 001017          BNE      2$          ; Br if 50 Hz
7          ;
8          ; 60 Hz clock
9          ;
10 007340 012737 000074 000000G      MOV      #60.,TK1SEC      ; Clock ticks per 1 second
11 007346 012737 000036 000000G      MOV      #30.,TK5VAL      ; Clock ticks per 0.5 seconds
12 007354 012737 000264 000000G      MOV      #180.,TK3SVL      ; Clock ticks per 3 seconds
13 007362 012737 000006 000000G      MOV      #6.,TK1VAL       ; Clock ticks per 0.1 seconds
14 007370 012700 001130          MOV      #600.,R0             ; Get # clock ticks per 10 seconds
15 007374 000416          BR       8$
16          ;
17          ; 50 Hz clock
18          ;
19 007376 012737 000062 000000G 2$:      MOV      #50.,TK1SEC      ; Clock ticks per 1 second
20 007404 012737 000031 000000G      MOV      #25.,TK5VAL      ; Clock ticks per 0.5 seconds
21 007412 012737 000226 000000G      MOV      #150.,TK3SVL      ; Clock ticks per 3 seconds
22 007420 012737 000005 000000G      MOV      #5.,TK1VAL       ; Clock ticks per 0.1 seconds
23 007426 012700 000764          MOV      #500.,R0             ; Get # clock ticks per 10 seconds
24          ;
25          ; Set number of clock ticks per day
26          ;
27 007432 012702 020700          8$:      MOV      #8640.,R2      ; (# seconds per day) / 10.
28 007436 070200          MUL      R0,R2          ; Get # clock ticks per day
29 007440 010237 000000G      MOV      R2,DATIMH      ; High-order value
30 007444 010337 000000G      MOV      R3,DATIML      ; Low-order value
31          ;
32          ; Do a fast check to make sure specified T/S line addresses are ok.
33          ;
34 007450 004737 010376'      CKLIN: CALL      LINCHK          ; CHECK T/S LINE ADDRESSES
35          ;
36          ; Do PRO-350 system initialization
37          ;
38          .IF      NE,PROASM
39          BIT      #CW$PRO,CONF02 ; Are we running on a PRO-350?
40          BEQ      INIDEV          ; Br if not
41          CALL      PROINI          ; Do PRO-350 initialization
42          MOV      #PIDRIV,PIDPTR ; Set up pointer to clock driven PI routine
43          .ENDC ; NE,PROASM
44          ;
45          ; Make entry in device handler table for TT device.
46          ;
47 007454 013737 000172' 000000G INIDEV: MOV      R50TT,PNAME      ; PERMANENT NAME "TT"
48 007462 012737 000000G 000000G      MOV      #DI$TT,DVSTAT      ; SET DEVICE STATUS FLAGS FOR TT
49 007470 005037 000000G          CLR      DVFLAG
50 007474 005037 000000G          CLR      DEVSIZ
51 007500 012737 000002 000000G      MOV      #2,HANENT          ; SET UP HANENT SO HANDLER LOOKS RESIDENT
52 007506 005037 000000G          CLR      NUMDEV          ; IT IS DEVICE # 0
53          ;
54          ; Make device table entry for LD (logical disk) device
55          ;
56 007512 012737 177777 000000G      MOV      #-1,LDDEVX          ; ASSUME LD SUPPORT NOT WANTED
57 007520 105737 000000G          TSTB      VLDSYS          ; IS LD SUPPORT GENNED IN?

```

* * * Initialization done with RT-11 running * * *

```

58 007524 001427          BEQ      6$          ; BR IF NOT
59 007526 062737 000002 000000G      ADD      #2, NUMDEV      ; ONE MORE DEVICE
60 007534 013701 000000G      MOV      NUMDEV, R1      ; GET DEVICE TABLE INDEX
61 007540 010137 000000G      MOV      R1, LDDEVX      ; REMEMBER INDEX NUMBER FOR LD DEVICE
62 007544 013761 000176' 000000G      MOV      R5OLD, PNAME(R1) ; SET DEVICE NAME ("LD")
63 007552 012761 000000C 000000G      MOV      #<DS$DIR+DS$SFN+DS$VSZ+DI$LD>, DVSTAT(R1); SET DEV STATUS FLAGS
64 007560 012761 000000G 000000G      MOV      #DX$EBA, DVFLAG(R1); Say buffers must be on even byte boundaries
65 007566 005061 000000G      CLR      DEVSIZ(R1)
66 007572 012761 000002 000000G      MOV      #2, HANENT(R1) ; SAY HANDLER IS RESIDENT
67 007600 004737 015312'      CALL      LDINIT      ; DETERMINE LD TRANSLATION TABLE FORMAT
68                                ;
69                                ; Make device table entry for CL (communications line) device
70                                ;
71 007604 005727 000000G      6$:      TST      #CLTOTL      ; Are there any communications lines?
72 007610 001402          BEQ      8$          ; Br if not
73 007612 004737 014572'      CALL      CLINIT      ; Initialize CL handler
74                                ;
75                                ; Disable clock interrupts.
76                                ;
77 007616 012737 000002 000000 8$:      MOV      #2, @#0      ; LOAD RTI IN LOCATION 0
78 007624 005037 000000G      CLR      @#CLKVEC      ; ATTACH CLOCK INTERRUPT TO 0
79 007630 032737 000000G 000000G      BIT      #CW$PRO, CFG2      ; ARE WE RUNNING ON A PRO?
80 007636 001402          BEQ      1$          ; BR IF NOT
81 007640 005037 000230          CLR      @#230      ; 380 CLOCK INTERRUPT VECTOR
82                                ;
83                                ; Set up memory parity control
84                                ;
85 007644 004737 017700'      1$:      CALL      PARSET      ; SET UP MEMORY PARITY CONTROL
86                                ;
87                                ; Determine how much memory is installed on machine
88                                ;
89 007650 004737 016616'      CALL      MEMTST      ; FIND OUT HOW MUCH PHYSICAL MEMORY THERE IS
90                                ;
91                                ; Set up information about the size of the job context area
92                                ;
93 007654 004737 017230'      CALL      CXTALC      ; Determine size of job context area
94                                ;
95                                ; Load TSX-Plus device handlers that go in low memory
96                                ;
97 007660 004737 017766'      CALL      GETHNL      ; Load low memory handlers
98                                ;
99                                ; Reserve space for interrupt vector intercept routines for mapped handlers
100                               ;
101 007664 010537 000122'      MOV      R5, XMVBAS      ; Save address of base of area for XM vectors
102 007670 013701 000124'      MOV      NMXHAN, R1      ; Get # mapped handlers
103 007674 006301          ASL      R1      ; Reserve room for 2 interrupts per handler
104 007676 062701 000000G      ADD      #NXIVMH, R1      ; Add # requested extra interrupt vectors
105 007702 070127 000032      MUL      #MPIVSZ, R1      ; Calc space needed for interrupt entry code
106 007706 060105          ADD      R1, R5      ; Advance the address of free memory
107                               ;
108                               ; Set up device index number and unit number for "SY:" device.
109                               ;
110 007710 004737 027316'      CALL      SETSY      ; SET UP INFO ABOUT SY DEVICE
111                               ;
112                               ; Open channel to TSKMON and set up information about it.
113                               ;
114 007714 004737 014270'      CALL      OPNKMN      ; OPEN CHANNEL TO TSKMON

```

* * * Initialization done with RT-11 running * * *

```

115      ;
116      ; Set up information about the IND program
117      ;
118 007720 004737 015372'      CALL      INDINI      ; INITIALIZE FOR IND PROGRAM
119      ;
120      ; Initialize the TSXUCL data file
121      ;
122 007724 004737 016134'      CALL      UCLINI
123      ;
124      ; Set name of device that UCL program is to be run from
125      ;
126 007730 013737 0000000 0000000      MOV      SYNAME,UCLNAM      ; SET DEVICE NAME FOR UCL PROGRAM
127      ;
128      ; Initialize spooling system
129      ;
130 007736 004737 011656'      CALL      SPLINI      ; INITIALIZE SPOOLING SYSTEM
131      ;
132      ; Open system swap file
133      ;
134 007742 105737 0000000      TSTB      VSWPFL      ; IS JOB SWAPPING ALLOWED?
135 007746 001402              BEQ      3$           ; BR IF NOT
136 007750 004737 010720'      CALL      OPNSWP      ; OPEN THE SYSTEM SWAP FILE
137      ;
138      ; Open swap file used for PLAS regions
139      ;
140 007754 004737 011352'      3$: CALL      OPNRSF      ; Open PLAS region swap file
141      ;
142      ; Set up information about which devices need to have their I/O mapped
143      ;
144 007760 004737 023160'      CALL      SETMIO      ; Set up information about mapped I/O
145      ;
146      ; We are finished allocating low-memory buffer space.
147      ;
148 007764 010500              MOV      R5,R0      ; ENSURE WE DON'T OVERFLOW 40KB
149 007766 004737 027622'      CALL      CHKMEN      ; ABORT IF > 40KB OR INTO RT-11
150      ;
151      ; From this point on carry the free memory address in R5
152      ; as a 64-byte block # in physical memory.
153      ;
154 007772 010537 0000000      MOV      R5,UMSYTP      ; SAVE ADDRESS OF NON-EXTENDED SYSTEM TOP
155 007776 062705 000077      ADD      #77,R5      ; BOUND UP TO 64-BYTE BOUNDARY
156 010002 072527 177772      ASH      #-6,R5      ; CONVERT TO 64-BYTE BLOCK #
157 010006 042705 176000      BIC      #176000,R5      ; KILL SIGN EXTENSION
158      ;
159      ; Allocate buffer space that is not constrained to 40Kb TSX-Plus region.
160      ;
161 010012 004737 013672'      CALL      ALBFX      ; ALLOCATE EXTENDED BUFFERS
162      ;
163      ; We will now do some allocation from the top of physical memory downward.
164      ; Save the base of free memory in R4 and get the top of free memory to R5.
165      ;
166 010016 010504              MOV      R5,R4      ; Save the base of free memory in R4
167 010020 010437 000136'      MOV      R4,FMEMLO      ; Save pointer above top of alloc low memory
168 010024 013705 000134'      MOV      FMEMHI,R5      ; Get 64-byte block # of free high memory
169      ;
170      ; Load any mapped system code
171      ;

```

* * * Initialization done with RT-11 running * * *

```

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

```

* * * Initialization done with RT-11 running * * *

```
229 ;
230 ; Unlock the USR so that TSEMT will be swapped back in.
231 ;
232 010200 11$: . UNLOCK ; RELEASE USR
233 ;
234 ; Read back into memory that part of the resident portion of TSX
235 ; that we overlayed with our work buffer.
236 ;
237 010202 013702 000154' MOV WRKSIZ,R2 ; Get size of work buffer
238 010206 006202 ASR R2 ; Convert to # words
239 010210 013703 000152' MOV WRKBUF,R3 ; Get address of work buffer area
240 010214 000241 CLC ; Convert to block # in TSX.SAV file
241 010216 006003 ROR R3
242 010220 000303 SWAB R3
243 010222 . READW #AREA,#17,WRKBUF,R2,R3 ; Read back TSX over work buffer
244 ;
245 ; See if user requested control-C abort
246 ;
247 010256 004737 027662' CALL CCATST ; JUMP TO INISTP IF ^C^C BEFORE THIS POINT
248 ;
249 ; Jump to code at end of TSINIT which takes over control from RT-11
250 ;
251 010262 000137 003534' JMP TAKOVR
```

* * * Subroutines * * *

```

1          .SBTTL * * * Subroutines * * *
2          .SBTTL ALCWRK -- Allocate a work buffer
3          ;-----
4          ; Allocate a 2048. byte work buffer over a resident portion of TSX.
5          ; This area will be restored from the TSX.SAV disk file after we
6          ; are finished using the work area.
7          ;
8          ; Outputs:
9          ;   WRKBUF = Address of base of work buffer.
10         ;   WRKSIZ = Size of work buffer (2048).
11         ;
12 010266 010246 ALCWRK: MOV      R2,-(SP)
13         ;
14         ; Get address of start of area where buffer can go and then bound
15         ; up to a block boundary.
16         ;
17 010270 012702 000000G      MOV      #EXCBUF,R2      ;Get address of base of buffer area
18 010274 062702 000777      ADD      #777,R2        ;Bound up to block boundary
19 010300 042702 000777      BIC      #777,R2        ;Set to block boundary
20 010304 010237 000152'      MOV      R2,WRKBUF
21 010310 012737 004000 000154'  MOV      #2048.,WRKSIZ
22         ;
23         ; Finished
24         ;
25 010316 012602      MOV      (SP)+,R2
26 010320 000207      RETURN

```

```

1          .SBTTL  ALCHRB -- Allocate Region Control Blocks for handlers
2          ;-----
3          ; This routine allocates and initializes empty Region Control Blocks for
4          ; use by device handlers.
5          ;
6          ; Inputs:
7          ;   R5 = Pointer to start of memory area where RCB's are to be built.
8          ;
9          ; Outputs:
10         ;   R5 = Pointer past end of RCB area.
11         ;
12 010322  010246  ALCHRB: MOV      R2, -(SP)
13         ;
14         ; Get count of # RCB's to build
15         ;
16 010324  013700  00000006      MOV      NDVRCB, R0      ; Get # RCB's to build for handlers
17         ;
18         ; Store pointer to start of RCB area and store -1 at beginning of area
19         ;
20 010330  010537  00000006      MOV      R5, HANRCB      ; Start of RCB area
21 010334  010537  00000006      MOV      R5, HANRCO      ; Store offset relative to MONVEC
22 010340  162737  00000006 00000006      SUB      #MONVEC, HANRCO ; Convert address to offset
23 010346  012725  177777      MOV      #-1, (R5)+      ; Store -1 at start of area
24         ;
25         ; Allocate and initialize to zero the RCB's
26         ;
27 010352  012702  0000005      1$:      MOV      #5, R2      ; Each RCB has 5 words
28 010356  005025      2$:      CLR      (R5)+      ; Zero the RCB
29 010360  077202      SOB      R2, 2$
30         ;
31         ; See if there are more RCB's to build
32         ;
33 010362  005300      DEC      R0      ; More RCB's to initialize?
34 010364  003372      BGT      1$      ; Loop if yes
35         ;
36         ; Store -1 at end of RCB area
37         ;
38 010366  012725  177777      MOV      #-1, (R5)+      ; Mark end of RCB list
39         ;
40         ; Finished
41         ;
42 010372  012602      MOV      (SP)+, R2
43 010374  000207      RETURN

```

```

1          .IF      NE,<PROASM-1>      ; If not assembling for Pro only
2          .SBTTL   LINCHK -- Check validity of T/S line
3          ;-----
4          ; LINCHK is called to check the validity of specified T/S line
5          ; vector and status register addresses.
6          ; If an uninstalled line is detected this routine aborts if
7          ; INIABT=1 or sets the $DEAD flag for the line if INIABT=0.
8          ;
9 010376 010146 LINCHK: MOV      R1,-(SP)
10 010400 010246      MOV      R2,-(SP)
11 010402 010346      MOV      R3,-(SP)
12 010404 010446      MOV      R4,-(SP)
13 010406 013746 000004      MOV      @#4,-(SP)      ; SAVE ORIGINAL TRAP VECTOR
14          ;
15          ; Take over trap control
16          ;
17 010412 012737 010612' 000004      MOV      #6$,@#4      ; CATCH TRAPS
18          ;
19          ; Loop through the test for each line.
20          ;
21 010420 012701 000000G      MOV      #LSTLIN,R1      ; NUMBER OF LAST LINE
22          ;
23          ; Determine if this is a primary line or an I/O line and set the
24          ; addresses of the interrupt service routines.
25          ;
26 010424 004737 006330' 1$: CALL      LINTYP      ; Determine the type of this line
27 010430 032761 000000G 000000G      BIT      ##HARD,LSW3(R1) ; Is this line connected to hardware?
28 010436 001440      BEQ      31$      ; Br if not
29 010440 016103 000000G      MOV      LMXNUM(R1),R3      ; IS THIS A DL-11 OR MULTIPLEXER LINE?
30 010444 001411      BEQ      2$      ; BR IF DL-11
31 010446 016302 000000G      MOV      MXCSR(R3),R2      ; GET DZ11 OR DH11 STATUS REGISTER ADDRESS
32 010452 001403      BEQ      11$      ; BR IF ALREADY MARKED AS DEAD
33 010454 016304 000000G      MOV      MXVEC(R3),R4      ; GET MUX INTERRUPT VECTOR ADDRESS
34 010460 000407      BR      3$
35 010462 004737 010564' 11$: CALL      4$      ; MARK LINE AS DEAD
36 010466 000424      BR      31$      ; CONTINUE CHECKING TERMINALS
37 010470 016102 000000G 2$: MOV      RSR(R1),R2      ; GET DL-11 STATUS REGISTER ADDRESS
38 010474 016104 000000G      MOV      INVEC(R1),R4      ; GET DL-11 INTERRUPT VECTOR ADDRESS
39          ; Check validity of status register address.
40 010500 020227 160000 3$: CMP      R2,#160000      ; IS IT IN I/O PAGE?
41 010504 103445      BLO      LINTRP      ; ERROR IF NOT
42 010506 032702 000007      BIT      #7,R2      ; IS IT ON 8-BYTE BOUNDARY?
43 010512 001042      BNE      LINTRP      ; ERROR IF NOT
44 010514 005712      TST      @R2      ; TRY TO ACCESS IT AND SEE IF WE TRAP
45          ; Check validity of interrupt vector address.
46 010516 020427 000060      CMP      R4,#60      ; CAN'T BE BELOW 60
47 010522 103445      BLO      BADVEC
48 010524 020427 000500      CMP      R4,#500      ; OR ABOVE 500
49 010530 103042      BHIS      BADVEC
50 010532 032704 000007      BIT      #7,R4      ; MUST BE ON 8-BYTE BOUNDARY
51 010536 001037      BNE      BADVEC
52          ; This line looks good. Check next.
53 010540 162701 000002 31$: SUB      #2,R1      ; MORE TO CHECK?
54 010544 003327      BGT      1$      ; BR IF YES
55          ;
56          ; Finished -- all lines look ok.
57          ;

```

```

58 010546 012637 000004      MOV      (SP)+, @#4      ; RESTORE TRAP VECTOR
59 010552 012604      MOV      (SP)+, R4
60 010554 012603      MOV      (SP)+, R3
61 010556 012602      MOV      (SP)+, R2
62 010560 012601      MOV      (SP)+, R1
63 010562 000207      RETURN
64
65      ; See if we should abort or just mark the line as dead.
66
67 010564 105737 000000G      4$:      TSTB      VINABT      ; DOES HE WANT TO ABORT?
68 010570 001013      BNE      LINTRP      ; YES
69 010572 052761 000000G 000000G      BIS      ##DEAD, LSW3(R1) ; MARK LINE AS DEAD
70 010600 005703      TST      R3          ; IS THIS A DL11 OR A MUX LINE?
71 010602 001402      BEQ      5$          ; BR IF DL11
72 010604 005063 000000G      CLR      MXCSR(R3)      ; MARK DZ OR DH AS DEAD
73 010610 000207      5$:      RETURN
74
75      ;
76      ; Trap occured while trying to access status register.
77
78 010612 004737 010564'      6$:      CALL      4$          ; REPORT ERROR OR MARK AS DEAD LINE
79 010616 000002      RTI          ; RETURN TO LINE CHECKING
80
81      ; Error: Invalid status register address.
82      ; R1 = Line number, R2 = status register address
83
84 010620      LINTRP: .PRINT      #TSXHD      ; PRINT ERROR MESSAGE
85 010626      .PRINT      #BADLIN
86 010634 000407      BR          ERP
87
88      ; Error: Invalid interrupt vector address.
89      ; R1 = Line number, R4 = interrupt vector address
90
91 010636      BADVEC: .PRINT      #TSXHD      ; PRINT ERROR MESSAGE
92 010644      .PRINT      #BDVMSG
93 010652 010402      MOV      R4, R2      ; GET VECTOR ADDRESS TO R2
94 010654 010200      ERP:      MOV      R2, R0      ; GET ADDRESS TO R0
95 010656 004737 027676'      CALL      PRTDCT      ; PRINT OCTAL VALUE
96 010662      .PRINT      #CRLF
97 010670      .PRINT      #BDLMSG      ; LINE # =
98 010676 010100      MOV      R1, R0      ; GET LINE NUMBER
99 010700 006200      ASR      R0          ; # 1
100 010702 004737 027746'      CALL      PRTDEC      ; PRINT LINE NUMBER
101 010706      .PRINT      #CRLF
102 010714 000137 004216'      JMP      INISTP      ; ABORT INITIALIZATION
103      .ENDC      ; NE, <PROASM-1>

```

```

1          .SBTTL  OPNSWP -- Open system swap file
2          ;-----
3          ; OPNSWP is called to open the TSX job swap file.
4          ; It also assigns swap file slots for each line.
5          ;
6          ; Inputs:
7          ;   R5 = Address of base of free memory region
8          ;
9          ; Outputs:
10         ;   SWPCHN = Set up for access to swap file.
11         ;   LSWPBK(i) = Starting block number in swap file for swap area for line.
12         ;
13 010720 010146 OPNSWP: MOV     R1, -(SP)
14 010722 010246      MOV     R2, -(SP)
15 010724 010346      MOV     R3, -(SP)
16         ;
17         ; Load RT-11 handler for swap device.
18         ;
19 010726 013700 0000000 MOV     SWDBLK, R0      ; Get name of device
20 010732 004737 027506' CALL    RTFTCH      ; Fetch the RT-11 handler
21 010736 103546      BCS     11$      ; Br if invalid device
22         ;
23         ; Compute the maximum number of slots in swap file that we could need
24         ;
25 010740 012703 0000000 MOV     #NSL+NDL, R3      ; Get # virtual lines and detached jobs
26 010744 012701 0000000 MOV     #LSTPL, R1      ; Get index to last primary line
27 010750 032761 0000000 0000000 1$: BIT     #$DEAD, LSW3(R1) ; Is this line installed?
28 010756 001001      BNE     5$      ; Br if not
29 010760 005203      INC     R3      ; Count another primary line
30 010762 162701 0000002 5$: SUB     #2, R1      ; Get next line index
31 010766 003370      BGT     1$      ; Loop if more lines to check
32 010770 020337 0000000 CMP     R3, VSWPSL      ; Compare with # slots specified
33 010774 002002      BGE     6$      ; Br if VSWPSL value is ok
34 010776 010337 0000000 MOV     R3, VSWPSL      ; Reduce number of slots in swap file
35         ;
36         ; Determine how many blocks are needed for each slot in swap file.
37         ;
38 011002 013703 0000000 6$: MOV     VHIMEM, R3      ; GET # BLOCKS NEEDED FOR LARGEST JOB SIZE
39 011006 006303      ASL     R3
40 011010 063703 0000000 ADD     CXTAG, R3      ; ADD # BLOCKS NEEDED FOR JOB CONTEXT AREA
41 011014 010337 0000000 MOV     R3, SLTSIZ      ; Save size of swap file slot
42         ;
43         ; Compute the total number of blocks needed for the swap file.
44         ;
45 011020 070337 0000000 MUL     VSWPSL, R3      ; Multiply by # slots in swap file
46         ;
47         ; R3 now contains total number of blocks needed in swap file.
48         ; See if swap file already exists on disk.
49         ;
50 011024 4$: .LOOKUP #AREA, #1, #SWDBLK; DOES SWAP FILE EXIST NOW?
51 011044 103415      BCS     2$      ; BR IF NOT
52         ; Swap file exists. See if it is the right size.
53 011046 020003      CMP     R0, R3      ; IS SWAP FILE THE RIGHT SIZE?
54 011050 001453      BEQ     3$      ; BR IF YES
55         ; Old swap file is not of correct size.
56         ; Delete it and open a new swap file.
57 011052      .CLOSE #1

```

```

58 011060          .DELETE #AREA,#1,#SWDBLK;DELETE THE OLD SWAP FILE
59
60          ; Create a new swap file.
61
62 011100          2$:      .ENTER  #AREA,#1,#SWDBLK,R3 ;CREATE A NEW SWAP FILE
63 011124 103443      BCS      9$          ;BR IF SOME ERROR ON OPEN
64
65          ; Swap file has been created.
66          ; Write to last block to reserve full space in file then close
67          ; and reopen the channel using a .lookup.
68
69 011126 005303      DEC      R3          ;GET # OF LAST BLOCK IN FILE
70 011130          .WRITW  #AREA,#1,#TSINIT,#256.,R3 ;WRITE TO LAST BLOCK IN FILE
71 011166 005203      INC      R3          ;GET BACK # BLOCKS IN FILE
72 011170          .CLOSE  #1          ;CLOSE FILE WE CREATED
73 011176 000712      BR      4$          ;NOW GO REOPEN USING A .LOOKUP
74
75          ; Swap file has been successfully opened using a .lookup.
76          ; Now copy channel status to TSX swap channel.
77
78 011200 012700 000000G 3$:      MOV      #SWPCHN,R0      ;POINT TO SWAP CHANNEL BLOCK
79 011204 013702 000000G      MOV      SWDBLK,R2      ;GET DEVICE NAME
80 011210 004737 027126'      CALL     SETCHN      ;SET UP SWAP CHANNEL INFO
81
82          ; Release the RT-11 device handler
83
84 011214          .RELEASE #SWDBLK      ;Release RT-11 device handler
85
86          ; Finished
87
88 011224 012603      MOV      (SP)+,R3
89 011226 012602      MOV      (SP)+,R2
90 011230 012601      MOV      (SP)+,R1
91 011232 000207      RETURN
92
93          ; Error: Cannot open swap file
94
95 011234          9$:      .PRINT  #TSXHD      ;PRINT ERROR MESSAGE
96 011242          .PRINT  #BADOPN
97 011250 004737 011276'      CALL     SPNEED      ;Print info about number of blocks needed
98
99          ; Error: Invalid device specification.
100
101 011254 010001      11$:     MOV      R0,R1      ;Save device name
102 011256          .PRINT  #TSXHD      ;Print error message
103 011264          .PRINT  #BADOPN
104 011272 004737 011324'      CALL     BADDEV      ;Print invalid device specification
105
106          ; Error: Number of contiguous blocks required.
107
108 011276          SPNEED: .PRINT  #CONSPC      ;Print contiguous blocks needed
109 011304 010300      MOV      R3,R0      ;GET # BLOCKS NEEDED FOR FILE
110 011306 004737 027746'      CALL     PRTDEC      ;DISPLAY # BLOCKS NEEDED
111 011312          .PRINT  #CRLF
112 011320 000137 004216'      JMP      INISTP      ;ABORT INITIALIZATION
113
114          ; Bad file specification.

```

115				
116	011324		BADDEV: . PRINT	#CFHMSG ; Print invalid device specification
117	011332	010100	MOV	R1, R0 ; Get the rad50 device name
118	011334	004737 030012'	CALL	PRTR50 ; Print rad50 device name
119	011340		. PRINT	#CRLF ; Print carriage return/line feed
120	011346	000137 004216'	JMP	INISTP ; Abort initialization

OPNRSF -- Open PLAS region swap file

```

1          .SBTTL  OPNRSF -- Open PLAS region swap file
2          ;-----
3          ; OPNRSF is called to open the swap file used for PLAS regions.
4          ;
5          ; Inputs:
6          ;   R5 = Address of base of free memory area.
7          ;
8          ; Outputs:
9          ;   SEGCHN = Set up to access swap file.
10         ;
11 011352 010346 OPNRSF: MOV      R3, -(SP)
12         ;
13         ; Return if this is a non-swapping system or if region swap file is
14         ; not wanted.
15         ;
16 011354 105737 00000000      TSTB    VSWPFL      ; Is this a non-swapping system?
17 011360 001513              BEQ      9$          ; Br if yes
18 011362 005737 00000000      TST     VPLAS       ; Is a PLAS swap file wanted?
19 011366 001510              BEQ      9$          ; Br if not
20         ;
21         ; Load RT-11 device handler for swap device
22         ;
23 011370 013700 00000000      MOV      RSFBLK, R0    ; Get name of device
24 011374 004737 027506'      CALL     RTFTCH       ; Try to fetch the RT-11 device handler
25 011400 103515              BCS      11$          ; Br if error on handler fetch
26 011402 013703 00000000      MOV      VPLAS, R3    ; Get # blocks in PLAS swap file
27         ;
28         ; See if PLAS swap file already exists on disk
29         ;
30 011406          4$: .LOOKUP #AREA, #1, #RSFBLK ; Try to find existing PLAS swap file
31 011426 103416          BCS      2$          ; Br if file does not now exist
32         ;
33         ; PLAS swap file exists.
34         ; See if it is the correct size.
35         ;
36 011430 020037 00000000      CMP      R0, VPLAS    ; Is swap file of the correct size?
37 011434 001453              BEQ      3$          ; Br if yes
38         ;
39         ; Old PLAS swap file is not of correct size.
40         ; Delete it and open a new swap file.
41         ;
42 011436          .CLOSE #1          ; Close and delete the old file
43 011444          .DELETE #AREA, #1, #RSFBLK; Delete the old file
44         ;
45         ; Create new swap file
46         ;
47 011464          2$: .ENTER #AREA, #1, #RSFBLK, VPLAS ; Create a new PLAS swap file
48 011512 103440          BCS      10$          ; Br if cannot create new file
49         ;
50         ; New swap file has been created.
51         ; Write to last block to reserve full file size
52         ; and then close and reopen with a lookup.
53         ;
54 011514 005303          DEC      R3              ; Get # of last block in file
55 011516          .WRITW #AREA, #1, #TSINIT, #256., R3
56 011554          .CLOSE #1
57 011562 000711          BR       4$          ; Go back and lookup file

```

```

58      ;
59      ; Swap file has been successfully opened using lookup.
60      ; Copy channel status to TSX channel block.
61      ;
62 011564 012700 000000G      3$:      MOV      #SEGCHN,R0      ;Point to TSX PLAS swap channel
63 011570 013703 000000G      ;      MOV      RSFBLK,R3      ;Get device name
64 011574 004737 027126'      ;      CALL     SETCHN      ;Set up TSX channel block
65      ;
66      ; Release the RT-11 device handler
67      ;
68 011600      ;      .RELEASE #RSFBLK      ;Release RT-11 device handler
69      ;
70      ; Finished
71      ;
72 011610 012603      9$:      MOV      (SP)+,R3
73 011612 000207      ;      RETURN
74      ;
75      ; Error -- Cannot open PLAS swap file
76      ;
77 011614      10$:     .PRINT  #TSXHD      ;Print error prefix
78 011622      ;      .PRINT  #RSFERR     ;Print error message
79 011630 004737 011276'      ;      CALL     SPNEED      ;Print information about amt of space needed
80      ;
81      ; Error: Invalid device specification.
82      ;
83 011634 010001      11$:     MOV      R0,R1      ;Save device name
84 011636      ;      .PRINT  #TSXHD     ;Print error message
85 011644      ;      .PRINT  #RSFERR     ;Print error message
86 011652 004737 011324'      ;      CALL     BADDEV      ;Print invalid device specification

```

```

1          .SBTTL  SPLINI -- Initialize spooling system
2          ;-----
3          ; SPLINI performs the initialization of the spooling system.
4          ; Inputs:
5          ;   R5 = Current base of free memory area.
6          ;
7 011656 005727 000000G SPLINI: TST      #SPLND      ; Are there any spooled devices?
8 011662 001401          BEQ      13$      ; Br if not
9 011664 000401          BR       10$      ; Initialize the spooled devices
10 011666 000207          13$:  RETURN
11          ;
12          ;   There are some spooled devices
13          ;
14 011670 010146          10$:  MOV      R1, -(SP)
15 011672 010246          MOV      R2, -(SP)
16 011674 010346          MOV      R3, -(SP)
17 011676 010446          MOV      R4, -(SP)
18 011700 010546          MOV      R5, -(SP)
19          ;
20          ;   Open each spooled device
21          ;
22 011702 105037 000000G          CLRB     NSPLDV      ; INIT COUNT OF # ACTUAL SPOOLED DEVICES
23 011706 012701 000000G          MOV      #SDCB, R1    ; POINT TO 1ST SDCB
24 011712 012703 000000G          MOV      #SPLDEV, R3  ; POINT TO TABLE OF RAD50 DEV NAMES
25 011716 012704 000000G          MOV      #SPLANM, R4  ; POINT TO TABLE OF ASCII DEV NAMES
26 011722 004737 012730'          2$:  CALL     FORCEO    ; FORCE UNIDENTIFIED UNIT #S TO 0
27 011726 011302          MOV      (R3), R2          ; GET RAD50 NAME OF SPOOLED DEVICE
28 011730 010261 000000G          MOV      R2, SDNAME(R1) ; SET NAME IN SDCB
29 011734 010100          MOV      R1, R0            ; GET ADDRESS OF SDCB
30 011736 062700 000000G          ADD      #SDANAM, R0   ; POINT TO CELL FOR ASCII NAME
31 011742 112420          MOVB     (R4)+, (R0)+        ; MOVE IN ASCII DEVICE NAME
32 011744 112420          MOVB     (R4)+, (R0)+
33 011746 112420          MOVB     (R4)+, (R0)+
34 011750 020227 000000G          CMP      R2, #DMYDEV   ; Is this a dummy entry for later patching?
35 011754 001451          BEQ      1$              ; Br if yes -- Ignore it
36 011756 010200          MOV      R2, R0            ; Get name to R0
37 011760 004737 012632'          CALL     CVTDVU      ; Convert name to device # and unit #
38 011764 010061 000000G          MOV      R0, SDDVU(R1) ; Store device # and unit # in SDCB
39 011770 010200          MOV      R2, R0            ; Get device name
40 011772 004737 012526'          CALL     CHKCLD      ; See if this is a CL device?
41 011776 103406          BCS      14$             ; Br if not
42 012000 004737 012442'          CALL     SPLCLD      ; Set up for spooling to CL device
43 012004 103414          BCS      3$              ; Br if invalid unit
44 012006 105237 000000G          INCB     NSPLDV      ; Count # of actual spooled devices
45 012012 000432          BR       1$              ; Process next device
46 012014 010100          14$:  MOV      R1, R0            ; Get address of SDCB
47 012016 062700 000000G          ADD      #SDCHAN, R0   ; Point to channel block within SDCB
48 012022 004737 027040'          CALL     OPNCHN      ; Set TSX-Plus channel block open to device
49 012026 103403          BCS      3$              ; Br if did not recognize device
50 012030 105237 000000G          INCB     NSPLDV      ; Count # actual spooled devices
51 012034 000421          BR       1$              ; GO PROCESS NEXT DEVICE
52          ;
53          ;   Error on opening spooled device
54          ;   Determine if we should print an error message or simply
55          ;   mark the spooled device as unavailable.
56          ;
57 012036 012761 000000G 000000G 3$:  MOV      #DMYDEV, SDNAME(R1); SAY THIS DEVICE IS NOT SPOOLED

```

```

58 012044 105737 0000000      TSTB    VINABT      ;ABORT OR CONTINUE ON ERRORS?
59 012050 001413              BEQ      1$           ;BR TO IGNORE DEVICE AND CONTINUE INIT
60 012052              .PRINT  #TSXHD      ;PRINT ERROR MESSAGE
61 012060              .PRINT  #BDSPOP
62 012066 010200              MOV      R2,R0      ;GET RAD50 DEVICE NAME
63 012070 004737 030012'      CALL     PRTR50     ;PRINT DEVICE NAME
64 012074 000137 004216'      JMP      INISTP     ;ABORT INITIALIZATION
65                          ;
66                          ; Process next spooled device
67                          ;
68 012100 062701 0000000      1$:    ADD     #SDCBSZ,R1      ;POINT TO NEXT SDCB
69 012104 005723              TST      (R3)+      ;POINT TO NEXT DEVICE NAME
70 012106 020327 0000000      CMP      R3,#SPLDVN    ;OPENED ALL SPOOLED DEVICES?
71 012112 103703              BLO      2$           ;BR IF MORE TO DO
72                          ;
73                          ; Open the spool file
74                          ;
75 012114 105737 0000000      TSTB    NSPLDV      ;ARE THERE ANY ACTUAL SPOOLED DEVICES?
76 012120 001521              BEQ      12$          ;BR IF THERE ARE NO ACTUAL SPOOLED DEVICES
77 012122 013700 0000000      MOV      SPLBLK,R0    ;Get name of device for spool file
78 012126 004737 027506'      CALL     RTFTCH      ;Fetch the RT-11 device handler
79 012132 103532              BCS      11$          ;Br if cannot fetch handler
80 012134 013702 0000000      MOV      NSPLBL,R2    ;GET # BLOCKS TO ALLOCATE FOR FILE
81 012140 005202              INC      R2           ;Add 1 extra block
82                          ;
83                          ; See if spool file already exists
84                          ;
85 012142              5$:    .LOOKUP #AREA,#1,#SPLBLK;SEE IF SPOOL FILE ALREADY EXISTS
86 012162 103415              BCS      6$           ;BR IF IT DOES NOT EXIST
87 012164 020002              CMP      R0,R2        ;IS IT THE RIGHT SIZE?
88 012166 001453              BEQ      7$           ;BR IF YES
89 012170              .CLOSE  #1                  ;IT EXISTS BUT IS OF WRONG SIZE
90 012176              .DELETE #AREA,#1,#SPLBLK;DELETE CURRENT FILE AND OPEN NEW ONE
91                          ;
92                          ; Open new spool file
93                          ;
94 012216              6$:    .ENTER #AREA,#1,#SPLBLK,R2;CREATE A NEW SPOOL FILE
95 012242 103456              BCS      8$           ;BR IF ERROR ON ENTER
96                          ; Write to last block in file to reserve full file space
97 012244 010203              MOV      R2,R3        ;Get # of blocks in file
98 012246 005303              DEC      R3           ;Get # of last block in file
99 012250              .WRITW #AREA,#1,#TSINIT,#256.,R3
100                          ; Now close and reopen using a lookup
101 012306              .CLOSE  #1                  ;CLOSE SPOOL FILE
102 012314 000712              BR      5$           ;GO BACK AND REOPEN USING LOOKUP
103                          ;
104                          ; Spool file has been successfully opened with a lookup.
105                          ; Save the channel status.
106                          ;
107 012316 012700 0000000      7$:    MOV      #SPLCHN,R0      ;SAVE CHANNEL STATUS HERE
108 012322 013702 0000000      MOV      SPLBLK,R2      ;GET DEVICE NAME
109 012326 004737 027126'      CALL     SETCHN      ;SAVE CHANNEL STATUS
110 012332              .RELEASE #SPLBLK      ;Release the RT-11 device handler
111                          ;
112                          ; Set number of free public blocks in spool file
113                          ;
114 012342 113703 0000000      MOVVB   NSPLDV,R3      ;Get # spooled devices

```

115	012346	070327	0000000	MUL	#PVSPBL,R3	;Times number of private blocks per dev
116	012352	005403		NEG	R3	
117	012354	063703	0000000	ADD	NSPLBL,R3	;Get # public spool blocks
118	012360	010337	0000000	MOV	R3,NFRESB	;This is number of public free spool blocks
119						
120						
121						
122	012364	012605		12\$:	MOV	(SP)+,R5
123	012366	012604			MOV	(SP)+,R4
124	012370	012603			MOV	(SP)+,R3
125	012372	012602			MOV	(SP)+,R2
126	012374	012601			MOV	(SP)+,R1
127	012376	000207		9\$:	RETURN	
128						
129						
130						
131	012400			8\$:	.PRINT	#TSXHD ;PRINT ERROR MESSAGE
132	012406				.PRINT	#BOSF
133	012414	000137	004216'		JMP	INISTP ;ABORT INITIALIZATION
134						
135						
136						
137	012420	010001		11\$:	MOV	RO,R1 ;Save device name
138	012422				.PRINT	#TSXHD ;Print error message
139	012430				.PRINT	#BOSF
140	012436	004737	011324'		CALL	BADDEV ;Print invalid device specification

SPLCLD -- Set up spooling to a CL device

```

1          .SBTTL  SPLCLD -- Set up spooling to a CL device
2          ;-----
3          ; SPLCLD is called to set up a spool device control block when
4          ; spooling is being directed to a Communication Line (CL) device.
5          ;
6          ; Inputs:
7          ;   R0 = CL unit number
8          ;   R1 = Address of SDCB
9          ;
10         ; Outputs:
11         ;   C-flag set ==> Invalid CL unit
12         ;
13 012442   010546   SPLCLD: MOV      R5,-(SP)
14
15         ; Make sure CL unit number is valie
16         ;
17 012444   010005           MOV      R0,R5           ;Get CL unit number
18 012446   020527   0000000   CMP      R5,#CLTOTL      ;Is this a valid unit #
19 012452   103022           BHIS     B$              ;Br if invalid
20
21         ; Set up channel control block in SDCB
22         ;
23 012454   005061   0000000   CLR      SDCHAN+C.SBLK(R1);Starting block # = 0
24 012460   020527   0000007   CMP      R5,#7          ;Is this a CL or C1 unit?
25 012464   101405           BLOS     1$              ;Br if CL unit
26 012466   162705   0000010   SUB      #8.,R5         ;Remove C1 unit # bias
27 012472   013700   0000000   MOV      C1DEVX,R0       ;Get C1 device index number
28 012476   000402           BR       2$
29 012500   013700   0000000   1$: MOV      CLDEVX,R0    ;Get CL device index number
30 012504   010061   0000000   2$: MOV      R0,SDCHAN+C.CSW(R1);Set device index number
31 012510   110561   0000000   MOV      R5,SDCHAN+C.DEVQ(R1);Set unit #
32
33         ; We successfully set up a CL unit
34         ;
35 012514   000241           CLC              ;Signal success on error
36 012516   000401           BR       9$
37
38         ; We cannot open this CL unit
39         ;
40 012520   000261   8$: SEC              ;Signal error
41
42         ; Finished
43         ;
44 012522   012605   9$: MOV      (SP)+,R5
45 012524   000207           RETURN

```

CHKCLD -- See if a device name is a CL or C1 unit

```

1          .SBTTL  CHKCLD -- See if a device name is a CL or C1 unit
2          ;-----
3          ; Determine if a rad50 device and unit name is a CL or C1 device.
4          ;
5          ; Inputs:
6          ;   R0 = Rad50 device spec
7          ;
8          ; Outputs:
9          ;   C-flag set      ==> Not a CL or C1 unit
10         ;   C-flag cleared ==> This is a CL or C1 unit
11         ;   R0 = CL unit number (0-15)
12         ;
13 012526   CHKCLD:
14         ;
15         ; See if this is a CL unit
16         ;
17 012526   020037   000202'      CMP      R0,R50CL      ; Is name "CL"?
18 012532   001411          BEQ      1$                ; Br if yes
19 012534   020037   000204'      CMP      R0,R50CL0     ; Is name in the range CL0 to CL7?
20 012540   103432          BLD      8$                ; Br if not
21 012542   020037   000206'      CMP      R0,R50CL7
22 012546   101005          BHI      2$
23 012550   163700   000204'      SUB      R50CL0,R0     ; Get CL unit number
24 012554   000422          BR       7$
25 012556   005000   1$:      CLR      R0                ; CL = CL0
26 012560   000420          BR       7$
27         ;
28         ; See if this is a C1 unit
29         ;
30 012562   020037   000210'      2$:      CMP      R0,R50C1      ; Is name "C1"?
31 012566   001413          BEQ      3$                ; Br if yes
32 012570   020037   000212'      CMP      R0,R50C10     ; Is name in the range C10 to C17?
33 012574   103414          BLD      8$                ; Br if not
34 012576   020037   000214'      CMP      R0,R50C17
35 012602   101011          BHI      8$
36 012604   163700   000212'      SUB      R50C10,R0     ; Get unit number
37 012610   062700   000010      ADD      #8.,R0        ; Bias by 8 for C1 units
38 012614   000402          BR       7$
39 012616   012700   000010      3$:      MOV      #8.,R0        ; C1 = CL8
40         ;
41         ; This is a CL or C1 unit
42         ;
43 012622   000241          7$:      CLC                      ; Signal success on return
44 012624   000401          BR       9$
45         ;
46         ; This is not a CL or C1 unit
47         ;
48 012626   000261          8$:      SEC                      ; Signal failure on return
49         ;
50         ; Finished
51         ;
52 012630   000207          9$:      RETURN

```

CVTDVU -- Convert device name to dev index and unit #

```

1          .SBTTL  CVTDVU -- Convert device name to dev index and unit #
2          ;-----
3          ; CVTDVU is called to convert a RAD50 device name into the corresponding
4          ; device index number and unit number.
5          ;
6          ; Inputs:
7          ; R0 = RAD50 device name.
8          ;
9          ; Outputs:
10         ; C-flag cleared ==> Conversion successful.
11         ; C-flag set      ==> Unable to find device name in tables.
12         ; R0 = Device index number (low byte), device unit number (high byte).
13         ;
14 012632 010246 CVTDVU: MOV      R2,-(SP)
15 012634 010346      MOV      R3,-(SP)
16         ;
17         ; Split the unit number off of the full device name
18         ;
19 012636 010003      MOV      R0,R3          ;Get full device name
20 012640 005002      CLR      R2            ;Set up for divide
21 012642 071227 000050 DIV      #50,R2      ;Split name and unit (R0=name, R1=unit)
22 012646 005703      TST      R3            ;Was a unit number specified?
23 012650 001402      BEQ      1$            ;Br if not
24 012652 162703 000036 SUB      #36,R3      ;Convert unit number to binary value
25 012656 010300 1$:  MOV      R3,R0          ;Get unit number
26 012660 000300      SWAB     R0            ;Position to high-order byte
27         ;
28         ; Look up the device name to get the device index
29         ;
30 012662 070227 000050 MUL      #50,R2      ;Now get the device name without unit number
31 012666 013702 000000G MOV      NUMDEV,R2      ;Get index number of last device
32 012672 020362 000000G 2$:  CMP      R3,PNAME(R2) ;Search for device in name table
33 012676 001407      BEQ      3$            ;Br if found it
34 012700 162702 000002 SUB      #2,R2      ;Try next device
35 012704 002372      BGE      2$            ;Loop if more to check
36         ;
37         ; Error, cannot find device name in tables
38         ;
39 012706 012700 177777      MOV      #-1,R0      ;Set device # = unit # = -1
40 012712 000261      SEC              ;Signal error on return
41 012714 000402      BR       9$
42         ;
43         ; Found the device in the tables
44         ;
45 012716 050200 3$:  BIS      R2,R0          ;Combine device # and unit #
46 012720 000241      CLC              ;Signal success on return
47         ;
48         ; Finished
49         ;
50 012722 012603 9$:  MOV      (SP)+,R3
51 012724 012602      MOV      (SP)+,R2
52 012726 000207      RETURN

```

FORCE0 -- Force a 2-char dev name to unit 0

```

1          .SBTTL  FORCE0 -- Force a 2-char dev name to unit 0
2          ; -----
3          ; Inputs: R3 points to a RAD50 device name
4          ;
5          ; Outputs: If the 3rd char of the device name pointed to by R3 is
6          ;           blank, then it is changed to 0
7          ;
8 012730 010346  FORCE0: MOV     R3, -(SP)
9 012732 010446      MOV     R4, -(SP)
10 012734 010546      MOV     R5, -(SP)
11 012736 011305      MOV     (R3), R5      ; MOVE CURRENT DEV NAME TO R5
12 012740 005004      CLR     R4           ; SET UP FOR DIVIDE
13 012742 071427 000050 DIV     #50, R4     ; SEPARATE INTO NAME AND UNIT
14 012746 005705      TST     R5           ; WAS 3RD CHAR BLANK?
15 012750 001012      BNE     9$           ; RETURN IF NOT
16 012752 010405      MOV     R4, R5       ; GET HIGH 2 CHARS
17 012754 005004      CLR     R4           ; SET UP FOR ANOTHER DIVIDE
18 012756 071427 000050 DIV     #50, R4     ; SEPARATE 1 & 2 CHARS
19 012762 005704      TST     R4           ; WAS CHAR 1 BLANK?
20 012764 001404      BEQ     9$           ; EMPTY OR INVALID DEV NAME!
21 012766 005705      TST     R5           ; WAS CHAR 2 BLANK?
22 012770 001402      BEQ     9$           ; 1-CHAR DEV NAME SHOULD BE INVALID???
23 012772 062713 000036 ADD     #^R 0, (R3) ; FORCE TO UNIT 0
24 012776 012605 9$:  MOV     (SP)+, R5
25 013000 012604      MOV     (SP)+, R4
26 013002 012603      MOV     (SP)+, R3
27 013004 000207      RETURN

```

ALOCBF -- Allocate buffer space

```

1          .SBTTL  ALOCBF -- Allocate buffer space
2          ;-----
3          ; ALOCBF is called to allocate space for buffers.  The allocated space
4          ; is not initialized but simply reserved.
5          ;
6          ; Inputs:
7          ;   R5 = Start of area to allocate buffer space in.
8          ;
9          ; Outputs:
10         ;   R5 = Address beyond end of buffer area.
11         ;   CHNBAS = Address of base of I/O channel space.
12         ;   CHNEND = Address past end of I/O channel space.
13         ;
14 013006   010146  ALOCBF: MOV      R1, -(SP)
15         ;
16         ; Assign space for I/O queue elements.
17         ;
18 013010   010537   000000G      MOV      R5, FREIOQ      ; START OF I/O QUEUE SPACE
19 013014   062705   000000C      ADD      #IOQSIZ*NUMIOQ, R5; RESERVE SPACE FOR I/O QUEUE ELEMENTS
20         ;
21         ; Assign space for shared PLAS region control blocks
22         ;
23 013020   010537   000000G      MOV      R5, SHRRCB      ; Start of area for RCB's
24 013024   013701   000000G      MOV      VNGR, R1        ; Get number of RCB's wanted
25 013030   020137   000000G      CMP      R1, VMXWIN      ; Must have one for each display window
26 013034   103002           BHS      13$                ; Br if ok
27 013036   013701   000000G      MOV      VMXWIN, R1      ; Force one for each window
28 013042   070127   000000G  13$:  MUL      #RC$$SZ, R1    ; Multiply by size of each block
29 013046   060105           ADD      R1, R5              ; Allocate space for RCB's
30 013050   010537   000000G      MOV      R5, SHRRCN      ; Address of end of region
31         ;
32         ; Assign space for fork blocks
33         ;
34 013054   012700   000000C      MOV      #<NUMFRK-FRKGEN>, R0 ; Get # fork blocks to allocate
35 013060   003404           BLE      11$                ; Br if none to allocate
36 013062   010537   000000G      MOV      R5, FRKINI      ; Set pointer to start of area
37 013066   062705   000000C      ADD      #<<NUMFRK-FRKGEN>*FQ$$SZ>, R5 ; Reserve space for fork blocks
38         ;
39         ; Assign space for job monitoring control blocks
40         ;
41 013072   013701   000000G  11$:  MOV      VMXMON, R1    ; Any job monitoring blocks wanted?
42 013076   001405           BEQ      10$                ; Br if not
43 013100   010537   000000G      MOV      R5, MONFQH      ; Start of job monitoring control blocks
44 013104   070127   000000G      MUL      #JM$$SZ, R1    ; Compute space needed for control blocks
45 013110   060105           ADD      R1, R5              ; Allocate the space
46         ;
47         ; Allocate space for system message buffers.
48         ;
49 013112   010537   000000G  10$:  MOV      R5, SNMSHD      ; HEAD OF SYSTEM MESSAGE BUFFER AREA
50 013116   062705   000000C      ADD      #<NMSNMB*SB$$SZ>, R5; RESERVE ROOM FOR MESSAGE BUFFERS
51         ;
52         ; Allocate space for INSTALLED program table
53         ;
54 013122   010537   000000G      MOV      R5, INSTBL      ; Base of table
55 013126   013701   000000G      MOV      VNUIP, R1      ; # slots for user installed programs
56 013132   062701   000000G      ADD      #NSIP, R1      ; Add # slots for system programs
57 013136   070127   000000G      MUL      #II$$SZ, R1    ; Multiply by size of each slot

```

```

58 013142 060105          ADD      R1,R5          ;Allocate space for table
59 013144 010537 0000000  MOV      R5,INSTBN        ;Pointer past end of INSTALL table
60                                     ;
61                                     ; Allocate space for device mount entries
62                                     ;
63 013150 010537 0000000  MOV      R5,CSHDEV        ;Point to start of area
64 013154 012701 0000000  MOV      #CD*$SZ,R1      ;Get size of each entry
65 013160 070137 0000000  MUL      VMXCSH,R1      ;Multiply by number of entries
66 013164 060105          ADD      R1,R5          ;Reserve space for table
67 013166 010537 0000000  MOV      R5,CSHDVN        ;Save pointer past end of table
68                                     ;
69                                     ; Allocate space for data cache control blocks
70                                     ;
71 013172 005737 0000000  TST      CSHALC          ;Is data caching wanted?
72 013176 001404          BEQ      12$             ;Br if not
73 013200 010537 0000000  MOV      R5,CCBHD        ;Head of free list area
74 013204 062705 0000000  ADD      #NUMCCB*CC*$SZ,R5 ;Allocate space for control blocks
75                                     ;
76                                     ; Allocate space for spool file control blocks
77                                     ;
78 013210 013701 0000000 12$:  MOV      NSPLFL,R1      ;Get # spool file control blocks needed
79 013214 001407          BEQ      1$              ;Br if none needed
80 013216 070127 0000000  MUL      #SFCBSZ,R1      ;Compute space needed by control blocks
81 013222 010537 0000000  MOV      R5,SFCB         ;Base of control block area
82 013226 060105          ADD      R1,R5          ;Allocate space for control blocks
83 013230 010537 0000000  MOV      R5,SFCBND        ;End of control block area
84                                     ;
85                                     ; Allocate space for a 16 byte vector for each multiplexer.
86                                     ; This vector is used to map from the mux line number to the
87                                     ; TSX-Plus logical line number.
88                                     ;
89 013234 012701 0000002 1$:  MOV      #2,R1         ;START WITH FIRST MUX
90 013240 020127 0000000 2$:  CMP      R1,#LSTMX      ;HAVE WE DONE ALL MUX'S?
91 013244 101007          BHI      5$              ;BR IF YES
92 013246 010561 0000000  MOV      R5,MXLNT(R1)    ;SET ADDRESS OF START OF VECTOR
93 013252 062705 0000020  ADD      #16.,R5         ;RESERVE SPACE FOR VECTOR
94 013256 062701 0000002  ADD      #2,R1         ;ADVANCE TO NEXT MUX
95 013262 000766          BR       2$
96                                     ;
97                                     ; Allocate buffers to hold characters for DMA transfers to DH11 multiplexers
98                                     ;
99 013264 012701 0000002 5$:  MOV      #2,R1         ;Start with first line
100 013270 020127 0000000 6$:  CMP      R1,#LSTPL      ;Is this a primary time-sharing line?
101 013274 101403          BLOS     3$              ;Br if yes
102 013276 020127 0000000  CMP      R1,#FSTIOL      ;Is this a CL line?
103 013302 103422          BLO      7$              ;Br if not
104 013304 026127 0000000 0000000 3$:  CMP      LCDTYP(R1),#CDX$DH ;Is this line connected to a DH11?
105 013312 001404          BEQ      8$              ;Br if yes
106 013314 026127 0000000 0000000  CMP      LCDTYP(R1),#CDX$VH ;Is this line connected to a DHV11?
107 013322 001012          BNE      7$              ;Br if not
108 013324 010561 0000000 8$:  MOV      R5,LDHB1B(R1) ;Set address of start of buffer 1
109 013330 010561 0000000  MOV      R5,LDHB1P(R1) ;Initialize pointer into buffer 1
110 013334 062705 0000000  ADD      #DHBFSZ,R5     ;Reserve space for buffer
111 013340 010561 0000000  MOV      R5,LDHB2B(R1) ;Set address of start of buffer 2
112 013344 062705 0000000  ADD      #DHBFSZ,R5     ;Reserve space for buffer
113 013350 062701 0000002 7$:  ADD      #2,R1         ;Get # of next line
114 013354 020127 0000000  CMP      R1,#LSTHL      ;Have we checked all lines?

```

115	013360	101743		BLOS	6\$		; Br if not
116	013362	005205		INC	R5		; Bound up to next word
117	013364	042705	000001	BIC	#1, R5		
118							
119							; Allocate space for tables that keep track of space in swap file
120							
121	013370	105737	000000G	TSTB	VSWPFL		; Is this a swapping system?
122	013374	001415		BEQ	14\$		; Br if not
123	013376	013700	000000G	MOV	VSWPSL, R0		; Get # slots in swap file
124	013402	006300		ASL	R0		; Allocate 2 bytes per slot
125	013404	010537	000000G	MOV	R5, SWPPOS		; Start of table with starting block #'s
126	013410	060005		ADD	R0, R5		; Allocate space
127	013412	010537	000000G	MOV	R5, SWPJOB		; Start of table with job #'s
128	013416	060005		ADD	R0, R5		; Allocate space
129	013420	010537	000000G	MOV	R5, SCPFHD		; Pointer to area with command packets
130	013424	062705	000000C	ADD	#NSCP*SP##SZ, R5		; Allocate space for swap command packets
131							
132							; Allocate a 512-byte buffer to use to access job context blocks
133							
134	013430	010537	000000G	14\$: MOV	R5, CXTBUF		; Set address of buffer
135	013434	062705	001400	ADD	#1400, R5		; Reserve space for the buffer
136							
137							; Make sure TSX is not too big.
138							
139	013440	010500		MOV	R5, R0		; GET CURRENT MEMORY ADDRESS
140	013442	004737	027622'	CALL	CHKMEM		; CHECK FOR SPACE OVERFLOW
141							
142							; Finished
143							
144	013446	012601		MOV	(SP)+, R1		
145	013450	000207		RETURN			

```

1          .SBTTL  ALCSLO -- Allocate silo buffers for lines
2          ;-----
3          ; Allocate the silo buffers that are used to hold characters as they
4          ; are received from serial lines.
5          ;
6          ; Inputs:
7          ;   R5 = Current pointer to start of free memory.
8          ;
9          ; Outputs:
10         ;   R5 = New pointer to start of free memory.
11         ;
12 013452 010146 ALCSLO: MOV     R1, -(SP)
13 013454 010246      MOV     R2, -(SP)
14         ;
15         ; Begin loop to check each line
16         ;
17 013456 012701 000000G      MOV     #LSTHL, R1      ;Get index to last hardware line
18         ;
19         ; Only allocate silo buffers for real lines
20         ;
21 013462 012702 000040 1$:   MOV     #32., R2      ;Set minimum size for time-sharing lines
22 013466 020127 000000G      CMP     R1, #LSTPL      ;Is this a primary line?
23 013472 101405      BLOS     2$      ;Br if yes
24 013474 020127 000000G      CMP     R1, #FSTIOL      ;Is this a CL line?
25 013500 103463      BLD      9$      ;Br if not
26 013502 012702 000020      MOV     #16., R2      ;Set minimum size for CL lines
27         ;
28         ; Determine how much space to allocate
29         ;
30 013506 016100 000000G 2$:   MOV     LHIRBA(R1), R0 ;Get requested size
31 013512 001002      BNE      8$      ;Br if a size was specified
32 013514 013700 000000G      MOV     VNCSLO, R0      ;Use default value
33 013520 020027 000000G 8$:   CMP     R0, #MAXSLO      ;Constrain size to 255 bytes
34 013524 101402      BLOS     3$      ;Br if ok
35 013526 012700 000000G      MOV     #MAXSLO, R0      ;Reduce size
36 013532 020002 3$:   CMP     R0, R2      ;Compare with acceptable minimum
37 013534 103001      BHIS     4$      ;Br if ok
38 013536 010200      MOV     R2, R0      ;Use minimum size allowed
39 013540 010061 000000G 4$:   MOV     R0, LHIRBA(R1) ;Set # bytes to be allocated for silo
40         ;
41         ; Allocate the space
42         ;
43 013544 010561 000000G      MOV     R5, LHIRBB(R1) ;Set base address of buffer
44 013550 010561 000000G      MOV     R5, LHIRBP(R1) ;Init pointer where to store next char
45 013554 010561 000000G      MOV     R5, LHIRBG(R1) ;Init pointer where to get next char
46 013560 010061 000000G      MOV     R0, LHIRBS(R1) ;Set # free bytes in buffer
47 013564 060005      ADD      R0, R5      ;Allocate space for buffer
48 013566 010561 000000G      MOV     R5, LHIRBE(R1) ;Set address beyond end of buffer
49         ;
50         ; Set up control information about when to send XON and XOFF
51         ;
52 013572 006200      ASR      R0      ;Get 1/2 of buffer size
53 013574 162700 000002      SUB      #2, R0      ;Minus two characters
54 013600 116102 000000G      MOVB    LHIRBC(R1), R2 ;Get specified size for XOFF point
55 013604 001002      BNE      5$      ;Br if value specified
56 013606 113702 000000G      MOVB    VNCXOF, R2      ;Try default
57 013612 020200 5$:   CMP      R2, R0      ;Is specified value ok?

```

```

58 013614 101401          BLOS 6$          ;Br if yes
59 013616 010002          MOV  R0,R2        ;No, use size/2-2
60 013620 110261 000000G  6$:  MOVB R2,LHIRBC(R1) ;Set # of chars when XOFF sent
61 013624 116102 000001G  MOVB LHIRBC+1(R1),R2 ;Get specified size for XON point
62 013630 001002          BNE 7$          ;Br if a value was specified
63 013632 113702 000000G  MOVB VNCXON,R2    ;Try default
64 013636 020200          7$:  CMP  R2,R0    ;Is specified value ok?
65 013640 101401          BLOS 10$         ;Br if ok
66 013642 010002          MOV  R0,R2        ;No, use size/2-2
67 013644 110261 000001G  10$: MOVB R2,LHIRBC+1(R1) ;Set # of chars when XON sent
68                               ;
69                               ; Do the next line
70                               ;
71 013650 162701 000002  9$:  SUB  #2,R1    ;Get next line index number
72 013654 003302          BGT  1$          ;Loop if more to do
73                               ;
74                               ; Finished
75                               ;
76 013656 005205          INC  R5          ;Force R5 to be even
77 013660 042705 000001  BIC  #1,R5
78 013664 012602          MOV  (SP)+,R2
79 013666 012601          MOV  (SP)+,R1
80 013670 000207          RETURN

```

ALBFX -- Allocate buffers in extended memory region

```

1          .SBTTL  ALBFX  -- Allocate buffers in extended memory region
2          ;-----
3          ;  ALBFX is called to allocate space for buffers that are not constrained
4          ;  to fit in the 40Kb region that TSX-Plus occupies.
5          ;
6          ;  Inputs:
7          ;    R5 = 64-Byte address of base of free memory region.
8          ;
9          ;  Outputs:
10         ;    R5 = Address above top of buffers allocated.
11         ;
12 013672 010146      ALBFX:  MOV     R1, -(SP)
13 013674 010246      MOV     R2, -(SP)
14 013676 010346      MOV     R3, -(SP)
15         ;
16         ;  Allocate character buffers for all lines
17         ;  Note: Character buffer space will be accessed by mapping through PAR 6.
18         ;
19 013700 012701 000002      MOV     #2, R1          ; GET 1ST JOB INDEX NUMBER
20 013704 032761 000000G 000000G 3#:  BIT     $$DEAD, LSW3(R1) ; IS THIS LINE INSTALLED?
21 013712 001047      BNE     2#                    ; BR IF NOT -- DON'T ALLOCATE ANY BUFFER SPACE
22 013714 020127 000000G      CMP     R1, #FSTD L      ; IS THIS A DETACHED JOB LINE?
23 013720 103403      BLD     1#                    ; BR IF NOT
24 013722 020127 000000G      CMP     R1, #LSTD L      ; DETACHED JOB LINE?
25 013726 101441      BLOS    2#                    ; BR IF DETACHED JOB -- DON'T ALLOCATE BUFFERS
26 013730 010561 000000G      1#:  MOV     R5, LTPAR(R1) ; SET PHYSICAL MEMORY PAR OFFSET FOR BUFFER
27 013734 012702 000000G      MOV     #VPAR6, R2      ; GET VIRTUAL MEMORY ADDRESS FOR BASE OF PAR6
28 013740 010261 000000G      MOV     R2, LINBUF(R1)  ; INPUT BUFFER STARTS AT BASE OF PAR6 REGION
29 013744 016100 000000G      MOV     LINSIZ(R1), R0   ; GET # BYTES FOR INPUT BUFFER
30 013750 010061 000000G      MOV     R0, LINSPC(R1)  ; SET # FREE BYTES IN INPUT BUFFER
31 013754 060002      ADD     R0, R2                  ; ADVANCE VIRTUAL ADDRESS
32 013756 010261 000000G      MOV     R2, LINEND(R1)  ; POINTS PAST END OF INPUT BUFFER
33 013762 010003      MOV     R0, R3                  ; Get # bytes in input buffer
34 013764 062703 000007      ADD     #7, R3           ; Bound up to multiple of 8
35 013770 072327 177775      ASH     #-3, R3          ; Get # bytes needed in activation-flag buffer
36 013774 060302      ADD     R3, R2                  ; Reserve space for activation-flag buffer
37 013776 060300      ADD     R3, R0                  ; Accumulate total buffer space
38 014000 010261 000000G      MOV     R2, LOTBUF(R1)  ; POINTS TO START OF OUTPUT BUFFER AREA
39 014004 066100 000000G      ADD     LOTSIZ(R1), R0   ; ACCUMULATE # BYTES IN BOTH BUFFERS
40 014010 066102 000000G      ADD     LOTSIZ(R1), R2   ; ADVANCE VIRTUAL ADDRESS COUNTER
41 014014 010261 000000G      MOV     R2, LOTEND(R1)  ; SAVE ADDRESS OF END OF OUTPUT BUFFER
42 014020 062700 000077      ADD     #77, R0          ; BOUND UP TO MULTIPLE OF 64 BYTES
43 014024 072027 177772      ASH     #-6, R0          ; CONVERT TO # 64-BYTE BLOCKS ALLOCATED
44 014030 060005      ADD     R0, R5                  ; ADVANCE PHYSICAL MEMORY PAR ADDRESS
45 014032 062701 000002      2#:  ADD     #2, R1      ; ADVANCE LINE NUMBER
46 014036 020127 000000G      CMP     R1, #LSTSL      ; HAVE WE DONE ALL LINES YET?
47 014042 101720      BLOS    3#                    ; BR IF MORE TO DO
48         ;
49         ;  Allocate space for shared file record locking data structures
50         ;
51 014044 005737 000000G      TST     VMXSF           ; Shared file support wanted?
52 014050 001451      BEQ     5#                    ; Br if not
53 014052 013737 000000G 000000G      MOV     VNUMDC, NUMDCD ; Set number of data cache blocks
54 014060 013737 000000G 000000G      MOV     VMXSFC, NUMCDB ; Set number of free CDB's
55 014066 010537 000000G      MOV     R5, LOKMEM      ; Set phys address of base of area
56 014072 012701 000000G      MOV     #FF##SZ, R1    ; Size of an FDB
57 014076 070137 000000G      MUL     VMXSF, R1       ; Times number of FDB's

```

```

58 014102 062701 000000C      ADD      #<NLINES*FW##SZ>,R1 ; Space needed for wait blocks
59 014106 013703 000000G      MOV      VMLBLK,R3      ; Max blocks a CDB may hold locked
60 014112 006303              ASL      R3              ; Two bytes per entry
61 014114 062703 000000G      ADD      #FC$LBN,R3      ; Add base size of a CDB
62 014120 070337 000000G      MUL      VMXSFC,R3      ; Times number of shared file channels
63 014124 060301              ADD      R3,R1          ; Accumulate space needed
64 014126 012703 000000G      MOV      #DC##SZ,R3      ; Size of a data cache descriptor
65 014132 070337 000000G      MUL      VNUMDC,R3      ; Times number of data cache entries
66 014136 060301              ADD      R3,R1          ; Reserve space for data cache descriptors
67 014140 062701 000100      ADD      #64.,R1          ; Bound up to 64 byte unit
68 014144 072127 177772      ASH      #-6,R1          ; Convert to # 64 byte units
69 014150 042701 176000      BIC      #176000,R1        ; Clear sign extension
70 014154 060105              ADD      R1,R5          ; Reserve space for data structures
71 014156 010537 000000G      MOV      R5,LOKCSH      ; Save pointer to start of cache buffer area
72 014162 013701 000000G      MOV      VNUMDC,R1      ; # shared-file data cache blocks wanted
73 014166 070127 000010      MUL      #8.,R1          ; 8 64-byte blocks each (512 bytes each)
74 014172 060105              ADD      R1,R5          ; Reserve space for data cache buffers
75                               ;
76                               ; Allocate space for mapped I/O buffers
77                               ;
78 014174 105737 000000G      5$:      TSTB      MIOFLG      ; Are any mapped I/O buffers needed?
79 014200 001415              BEQ      7$              ; Br if not
80 014202 013701 000000G      MOV      MIOBHD,R1      ; Point to 1st mapped I/O control block
81 014206 001412              BEQ      7$              ; Br if no more buffers needed
82 014210 010561 000000G      6$:      MOV      R5,MI$SBP(R1) ; Set address of base of buffer
83 014214 113703 000000G      MOV      VMIOSZ,R3      ; Get # blocks for buffer
84 014220 072327 000003      ASH      #3,R3          ; Convert to # 64-byte pages
85 014224 060305              ADD      R3,R5          ; Allocate space for buffer
86 014226 016101 000000G      MOV      MI$LNK(R1),R1    ; Get address of next control block
87 014232 001366              BNE      6$              ; Loop if more to allocate
88                               ;
89                               ; Allocate space for performance monitor data buffer if it is wanted.
90                               ;
91 014234 012701 000000G      7$:      MOV      #PMSIZE,R1 ; DID USER GEN IN PERFORMANCE MONITOR FEATURE?
92 014240 001407              BEQ      9$              ; BR IF NOT
93 014242 010537 000000G      MOV      R5,PMPAR      ; SET BASE ADDRESS OF PM BUFFER
94 014246 062701 000077      ADD      #77,R1          ; BOUND SIZE UP TO 64-BYTE MULTIPLE
95 014252 072127 177772      ASH      #-6,R1          ; CONVERT TO # 64-BYTE BLOCKS
96 014256 060105              ADD      R1,R5          ; ADVANCE FREE MEMORY POINTER
97                               ;
98                               ; Finished
99                               ;
100 014260 012603      9$:      MOV      (SP)+,R3
101 014262 012602      MOV      (SP)+,R2
102 014264 012601      MOV      (SP)+,R1
103 014266 000207      RETURN

```

OPNKMN -- Open channel to TSKMON

```

1          .SBTTL  OPNKMN -- Open channel to TSKMON
2          ;-----
3          ; OPNKMN is called to open an I/O channel to TSKMON SAV file and to
4          ; set up information about TSKMON.
5          ;
6          ; Inputs:
7          ;   R5 = Address of base of free memory area
8          ;
9          ; Outputs:
10         ;   KMNCHN = Saved status of channel to use to access TSKMON SAV file.
11         ;   KMNTOP = Top of memory address for TSKMON.
12         ;   KMNHI  = Top address of TSKMON - KMNBAS.
13         ;   KMNPGS = Number of 256-word memory pages needed for TSKMON & context area
14         ;   KMNSTK = Address of stack to use while TSKMON running.
15         ;   KMNSTR = Starting address of TSKMON.
16         ;
17 014270 010246  OPNKMN: MOV      R2,-(SP)
18
19         ; Lookup TSKMON file.
20
21 014272          .LOOKUP #AREA,#1,#KMNNAM ;TRY TO FILE KMON SAV FILE
22 014312 103517  BCS      9$             ;BR IF NOT THERE
23
24         ; Read block 0 of save file and extract some information.
25
26 014314 013702 000152'  MOV      WRKBUF,R2          ;Point to work buffer
27 014320          .READW #AREA,#1,R2,#256.,#0 ;READ BLOCK 0 OF SAV FILE
28
29         ; Determine size of kmon
30 014354 016200 000050  MOV      50(R2),R0          ;GET TOP ADDRESS OF KMON
31 014360 062700 000003  ADD      #3,R0             ;BOUND UP TO NEXT WORD
32 014364 042700 000001  BIC      #1,R0             ;FORCE EVEN
33 014370 010037 000000G  MOV      R0,KMNTOP
34
35         ; Determine number of 256-word memory pages needed while kmon running.
36 014374 162700 000000G  SUB      #KMNBAS,R0          ;BASE ADDRESS OF KMON
37 014400 010037 000000G  MOV      R0,KMNHI          ;TOP OF TSKMON - KMNBAS
38 014404 062700 000777  ADD      #511.,R0          ;BOUND UP TO PAGE SIZE
39 014410 000241  CLC
40 014412 006000  ROR      R0             ;CVT TO # WORDS
41 014414 000300  SWAB     R0             ;CVT TO # PAGES
42 014416 042700 177400  BIC      #^C377,R0
43 014422 063700 000000G  ADD      CXTAG,R0          ;# PAGES NEEDED FOR JOB CONTEXT AREA
44 014426 010037 000000G  MOV      R0,KMNPGS          ;# 256-wd pages needed for kmon + context area
45
46         ; Determine Kmon stack pointer.
47 014432 016237 000042 000000G  MOV      42(R2),KMNSTK ;INITIAL STACK POINTER FOR KMON
48
49         ; Determine Kmon starting address.
50 014440 016237 000040 000000G  MOV      40(R2),KMNSTR ;STARTING ADDRESS
51
52         ; Set up demo-system time limit
53         ; (If this is a demo version of TSX-Plus, the number of minutes the system
54         ; is to run before it crashes is stored in location 300 of TSKMON)
55
56 014446 016237 000300 000000G  MOV      300(R2),DTLX ;SET DEMO TIME-LIMIT
57
58         ; Now save status of channel so we can do a reopen when we need kmon.
59
60 014454 012700 000000G  MOV      #KMNCHN,R0          ;GET KMON CHANNEL SAVE AREA
61 014460 013702 000072'  MOV      KMNNAM,R2          ;GET KMON DEVICE NAME

```

```
58 014464 004737 027126'          CALL    SETCHN          ; SAVE CHANNEL STATUS
59                                ;
60                                ; Lookup CCL.SAV and save channel status for it.
61                                ;
62 014470                                . LOOKUP #AREA,#1,#CCLNAM; LOOKUP SY:CCL.SAV
63 014510 103410                      BCS      B#              ; BR IF CAN'T FIND CCL
64 014512 012700 000000G              MOV      #CCLSAV,R0      ; CHANNEL SAVE AREA
65 014516 013702 000102'              MOV      CCLNAM,R2      ; DEVICE NAME
66 014522 004737 027126'              CALL    SETCHN          ; SAVE CHANNEL STATUS
67                                ;
68                                ; Finished
69                                ;
70 014526 012602                      10$:    MOV      (SP)+,R2
71 014530 000207                      RETURN
72                                ;
73                                ; Error: We could not find SY:CCL.SAV
74                                ;
75 014532                      B$:      . PRINT  #TSXHD
76 014540                      . PRINT  #NOCCCL          ; PRINT ERROR MESSAGE
77 014546 000137 004216'              JMP      INISTP          ; ABORT INITIALIZATION
78                                ;
79                                ; Error: We could not locate TSKMON SAV file.
80                                ;
81 014552                      9$:      . PRINT  #TSXHD          ; PRINT ERROR MESSAGE
82 014560                      . PRINT  #NOKMON
83 014566 000137 004216'              JMP      INISTP          ; ABORT INITIALIZATION
```

```

1          .SBTTL  CLINIT -- Initialize CL handler
2          ;-----
3          ; Perform initialization for CL (Communication Line) handler
4          ;
5          ; Inputs:
6          ;   R5 = Address of start of free memory area.
7          ;
8          ; Outputs:
9          ;   R5 = Address of new start of free memory area.
10         ;
11 014572 010146 CLINIT: MOV     R1,-(SP)
12 014574 010246      MOV     R2,-(SP)
13 014576 010346      MOV     R3,-(SP)
14         ;
15         ; Initialize tables for each CL unit
16         ;
17 014600 005003      CLR     R3          ; Accumulate ring buffer sizes in R3
18 014602 012701 000000C      MOV     #2*<CLTOTL-1>,R1 ; Get index # of last CL unit
19         ;
20         ; See if this CL unit is connected to hardware or is free to be
21         ; connected later to a time-sharing line.
22         ;
23 014606 016102 000000G 1$:      MOV     CL$LIX(R1),R2 ; Is this CL unit associated with a line?
24 014612 001416          BEQ     5$          ; Br if not
25 014614 012762 000000G 000000G      MOV     #$SXON,LSW10(R2) ; Send XON when we start the line
26 014622 010162 000000G      MOV     R1,LCLUNT(R2) ; Associate the CL unit with this line
27 014626 005762 000000G      TST     RSR(R2)      ; Does this unit have a specified RSR addr?
28 014632 001006          BNE     5$          ; Br if yes
29 014634 005762 000000G      TST     LMXNUM(R2)    ; Is this a mux line?
30 014640 001003          BNE     5$          ; Br if yes
31 014642 005061 000000G      CLR     CL$EPS(R1)    ; Say no endstring buffer
32 014646 000472          BR      4$          ; Line is not genned in
33         ;
34         ; Allocate and set up pointers for the output ring buffers
35         ;
36 014650 010561 000000G 5$:      MOV     R5,CL$ORB(R1) ; Start of output ring buffer
37 014654 010561 000000G      MOV     R5,CL$ORP(R1) ; Input character pointer
38 014660 010561 000000G      MOV     R5,CL$ORG(R1) ; Next available character pointer
39 014664 012700 000000G      MOV     #CLORSZ,R0      ; Get default output ring buffer size
40 014670 005702          TST     R2          ; Is this CL unit connected to a line?
41 014672 001402          BEQ     6$          ; Br if not
42 014674 016200 000000G      MOV     LOTSIZ(R2),R0 ; Get size of output ring buffer
43 014700 010061 000000G 6$:      MOV     R0,CL$ORA(R1) ; Set size of output ring buffer
44 014704 010061 000000G      MOV     R0,CL$ORS(R1) ; Available space in ring buffer
45 014710 060005          ADD     R0,R5        ; Point beyond end of ring buffer
46 014712 010561 000000G      MOV     R5,CL$ORE(R1) ; Address past end of ring buffer
47 014716 060003          ADD     R0,R3        ; Accumulate size of output ring buffers
48         ;
49         ; Allocate space for end-of-file string buffer
50         ;
51 014720 010561 000000G      MOV     R5,CL$EPS(R1) ; Set pointer to end-of-file string buffer
52 014724 005061 000000G      CLR     CL$EPP(R1)    ; No string to print yet
53 014730 062705 000001G      ADD     #<CLEDFS+1>,R5 ; Reserve space for buffer
54 014734 062703 000001G      ADD     #<CLEDFS+1>,R3 ; Accumulate buffer space
55         ;
56         ; Initialize end-of-file form-feed count
57         ;

```

```

58 014740 005061 0000000 CLR CL$EPN(R1) ; Init ENDPAGE=0
59 ;
60 ; Initialize option word
61 ;
62 014744 012700 0000000 MOV #<CD$DEF>,R0 ; Get default option flags
63 014750 005702 TST R2 ; Is this CL unit connected with a line?
64 014752 001421 BEQ 7$ ; Br if not
65 014754 016202 0000000 MOV ILSW2(R2),R2 ; Get line options
66 014760 032702 0000000 BIT $$TAB,R2 ; Does hardware support tabs?
67 014764 001402 BEQ 2$ ; Br if not
68 014766 052700 0000000 BIS #CD$TAB,R0 ; Set hardware-tab flag
69 014772 032702 0000000 2$: BIT $$FORM,R2 ; Does hardware support form feeds?
70 014776 001402 BEQ 3$ ; Br if not
71 015000 052700 0000000 BIS #CD$FF,R0 ; Set hardware-form-feed flag
72 015004 032702 0000000 3$: BIT $$B8BIT,R2 ; Does hardware want B bit support?
73 015010 001402 BEQ 7$ ; Br if not
74 015012 052700 0000000 BIS #CD$B8T,R0 ; Enable B bit support for CL line
75 015016 010061 0000000 7$: MOV R0,CL$OPT(R1) ; Set options for this CL line
76 ;
77 ; Initialize page length
78 ;
79 015022 012761 000102 0000000 MOV #66,CL$LEN(R1) ; Say page length = 66 lines
80 ;
81 ; Initialize status flags
82 ;
83 015030 005061 0000000 CLR CL$STA(R1) ; Initialize status flags
84 ;
85 ; Do next line
86 ;
87 015034 162701 000002 4$: SUB #2,R1 ; Get index # of next unit
88 015040 002262 BGE 1$ ; Loop if more units to initialize
89 ;
90 ; Make a device table entry for "CL" device
91 ;
92 015042 062737 000002 0000000 ADD #2,NUMDEV ; One more device
93 015050 013701 0000000 MOV NUMDEV,R1 ; Get device table index
94 015054 010137 0000000 MOV R1,CLDEVX ; Remember index number of CL device
95 015060 013761 000202' 0000000 MOV R5OCL,PNAME(R1) ; Set device name ("CL")
96 015066 012761 0000000 0000000 MOV #CLSTS,DVSTAT(R1) ; Set dev status flags
97 015074 012761 0000000 0000000 MOV #<DX$NCA!DX$NMT!DX$NRD>,DVFLAG(R1) ; Device info flags
98 015102 005061 0000000 CLR DEVSIZ(R1) ; Clear device size
99 015106 012761 0000060 0000000 MOV #CLHEAD+6,HANENT(R1) ; Set handler entry point (4th word)
100 015114 062703 0000000 ADD #CLSIZE,R3 ; Get size of handler
101 015120 062703 0000000 0000000 ADD #<<CLTOTL*46.>+<NIOL*32.>>,R3 ; Add size of tables in TSGEN
102 015124 010361 0000000 MOV R3,HANSIZ(R1) ; Set size of handler
103 015130 005205 INC R5 ; Make sure free-memory pointer is even
104 015132 042705 000001 BIC #1,R5
105 ;
106 ; Make a device table entry for C1 if there are more than 8 CL units
107 ;
108 015136 022727 0000000 000010 CMP #CLTOTL,#8. ; Are there more than 8 CL units?
109 015144 101430 BLOS 13$ ; Br if not -- Don't need C1
110 015146 062737 000002 0000000 ADD #2,NUMDEV ; One more device
111 015154 013701 0000000 MOV NUMDEV,R1 ; Get device table index
112 015160 010137 0000000 MOV R1,C1DEVX ; Remember index number of CL device
113 015164 013761 000210' 0000000 MOV R5OC1,PNAME(R1) ; Set device name ("C1")
114 015172 012761 0000000 0000000 MOV #CLSTS,DVSTAT(R1) ; Set dev status flags

```

```

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

```

LDINIT -- Determine LD translation table format

```

1
2
3
4
5
6
7
8 015312 010146
9 015314 113701 000000G
10 015320 001022
11 015322 010246
12 015324 010346
13 015326 013703 000000G
14 015332 012701 000001
15 015336 012702 000556'
16 015342 022203
17 015344 001404
18 015346 005712
19 015350 001374
20 015352 012701 000002
21 015356 110137 000000G
22 015362 012603
23 015364 012602
24 015366 012601
25 015370 000207

.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
;
LDINIT: MOV R1, -(SP)
        MOVB LDVERS, R1 ; See if value was specified in TSGEN
        BNE 9$ ; If value specified in TSGEN, use it
        MOV R2, -(SP)
        MOV R3, -(SP)
        MOV SYSVER, R3 ; Get current RT version and update
        MOV #1, R1 ; Assume version format RT5.3 or earlier
        MOV #LD1TBL, R2 ; Point to table of versions using format 1
1$: CMP (R2)+, R3 ; Does this entry match actual version?
     BEQ 5$ ; Exit if so
     TST @R2 ; End of table?
     BNE 1$ ; Keep looking if more entries
     MOV #2, R1 ; If not in format 1 table, use format 2
5$: MOVB R1, LDVERS ; Remember for LD SPFUN 372
     MOV (SP)+, R3
     MOV (SP)+, R2
9$: MOV (SP)+, R1
     RETURN

```

```

1          .SBTTL  INDINI -- Initialize IND program
2          ;-----
3          ; Perform initialization for IND program.
4          ;
5          ; Outputs:
6          ; If IND is available, the following information is set up:
7          ;   INDSAV = 5 word .SAVESTATUS block for SY:IND.SAV file
8          ;   INDDBL = Lowest block # within IND.SAV file of data overlay segment.
9          ;   INDDBS = Number of blocks used for data overlay segment.
10         ;   INDTSV = 5 word .SAVESTATUS block for SY:TSXIND.TSX file
11         ;
12 015372 010246 INDINI: MOV     R2,-(SP)
13 015374 010346         MOV     R3,-(SP)
14         ;
15         ; Determine if IND support is wanted
16         ;
17 015376 005037 000000G         CLR     INDSAV         ; ASSUME IND SUPPORT NOT WANTED
18         ;
19         ; Lookup SY:IND.SAV file
20         ;
21 015402 013737 000000G 000264'         MOV     SYNAME,INDNAM     ; LOOK UP IND ON BOOT DEVICE
22 015410         .LOOKUP #AREA,#1,#INDNAM ; TRY TO FIND SY:IND.SAV
23 015430 103002         BCC     4$             ; BR IF FOUND IND
24 015432 000137 016064'         JMP     9$             ; IF CAN'T FIND IND, THEN NO IND SUPPORT
25         ;
26         ; Set up information about IND overlay data segment
27         ;
28 015436 013703 000152' 4$:      MOV     WRKBUF,R3         ; Get pointer to work buffer
29 015442         .READW #AREA,#1,R3,#256.,#0 ; READ IN BLOCK 0 OF SAV FILE
30 015476 016302 000064         MOV     64(R3),R2         ; GET POINTER TO OVERLAY TABLE
31 015502         .READW #AREA,#1,R3,#256.,#1 ; READ IN BLOCK 1 WITH OVERLAY TABLE
32 015540 162702 001000         SUB     #1000,R2         ; GET ADDRESS OF OVERLAY TABLE REL TO BLOCK 1
33 015544 060302         ADD     R3,R2             ; ADD BASE ADDRESS WHERE BLOCK 1 DATA IS
34 015546 011203         MOV     (R2),R3         ; Get virtual address of segment 0
35 015550 020312 5$:      CMP     R3,(R2)         ; Search for 1st segment with different addr
36 015552 001003         BNE     6$             ; Br if found it (this is the data segment)
37 015554 062702 000006         ADD     #6,R2         ; Point to overlay table entry for next seg
38 015560 000773         BR     5$
39 015562 005722 6$:      TST     (R2)+         ; Point to word with block # if SAV file
40 015564 012237 000000G         MOV     (R2)+,INDDBL     ; GET BLOCK # IN SAV FILE OF DATA OVERLAY
41 015570 011202         MOV     (R2),R2         ; GET # OF WORDS IN OVERLAY SEGMENT
42 015572 062702 000377         ADD     #255.,R2         ; ROUND UP TO NEXT BLOCK
43 015576 000302         SWAB    R2             ; CONVERT # WORDS TO # BLOCKS
44 015600 042702 177400         BIC     #^C<377>,R2
45 015604 010237 000000G         MOV     R2,INDDBS         ; SAVE # BLOCKS USED FOR DATA OVERLAY
46         ;
47         ; Do .SAVESTATUS on channel opened to IND.SAV file so that we
48         ; can do a reopen to access it from KMON.
49         ;
50 015610 012700 000000G         MOV     #INDSAV,R0         ; GET ADDRESS OF SAVESTATUS BLOCK
51 015614 013702 000264'         MOV     INDNAM,R2         ; GET RAD50 DEVICE NAME
52 015620 004737 027126'         CALL    SETCHN         ; SAVE FILE STATUS
53         ;
54         ; Determine how much space is needed for SY:TSXIND.TSX swap file
55         ;
56 015624 013703 000000G         MOV     INDDBS,R3         ; GET # BLOCKS NEEDED PER JOB
57 015630 070327 000000C         MUL     #<LSTSL/2>,R3     ; TIMES TOTAL NUMBER OF JOBS

```

```

58
59
60      ; Load Rt-11 device handler for ind swap file.
61
62 015634 013700 0000000      MOV      INDFIL,R0      ;Get name of the device
63 015640 004737 027506'      CALL      RTFTCH      ;Try to fetch the RT-11 device handler
64 015644 103522              BCS      11$      ;Br if error on handler fetch
65
66      ; See if TSXIND file already exists
67
68 015646              . LOOKUP #AREA,#1,#INDFIL ; DOES SY:TSXIND.TSX FILE EXIST NOW?
69 015666 103415              BCS      1$      ;BR IF NOT
70 015670 020003              CMP      R0,R3      ; IS IT OF THE CORRECT SIZE?
71 015672 001462              BEQ      2$      ;BR IF YES
72 015674              . PURGE #1      ; FILE IS OF WRONG SIZE
73 015702              . DELETE #AREA,#1,#INDFIL;DELETE OLD FILE
74
75      ; File does not now exist
76      ; Create new file
77
78 015722      1$:      . ENTER #AREA,#1,#INDFIL,R3 ; CREATE NEW TSXIND FILE
79 015746 103451              BCS      10$      ;BR IF ERROR ON CREATE
80 015750 010302              MOV      R3,R2      ; # BLOCKS IN FILE
81 015752 005302              DEC      R2      ; GET # OF LAST BLOCK IN FILE
82 015754              . WRITW #AREA,#1,WRKBUF,#256.,R2 ; WRITE TO LAST BLOCK OF FILE
83 016012              . CLOSE #1      ; NOW CLOSE THE FILE
84 016020              . LOOKUP #AREA,#1,#INDFIL ; REOPEN TSXIND FILE WITH LOOKUP
85
86      ; Do .SAVESTATUS for SY:TSXIND.TSX file
87
88 016040 012700 0000000      2$:      MOV      #INDTSV,R0      ; POINT TO SAVESTATUS BLOCK
89 016044 013702 0000000      MOV      INDFIL,R2      ; GET RAD50 DEVICE NAME
90 016050 004737 027126'      CALL      SETCHN      ; SAVE FILE INFO
91 016054              . RELEAS #INDFIL      ; Release device handler
92
93      ; Finished
94
95 016064 012603      9$:      MOV      (SP)+,R3
96 016066 012602      MOV      (SP)+,R2
97 016070 000207      RETURN
98
99      ; Error occurred while opening SY:TSXIND.TSX file
100
101 016072      10$:      . PRINT #TSXHD      ; Print error message
102 016100      . PRINT #INDOPN
103 016106 004737 011276'      CALL      SPNEED      ; Print info about number of blocks needed
104
105      ; Error: Invalid device specification.
106
107 016112 010001      11$:      MOV      R0,R1      ; Save device name
108 016114      . PRINT #TSXHD      ; Print error message
109 016122      . PRINT #INDOPN
110 016130 004737 011324'      CALL      BADDEV      ; Print invalid device specification

```

UCLINI -- Initialize TSXUCL data file

```

1          .SBTTL  UCLINI -- Initialize TSXUCL data file
2          ;-----
3          ; UCLINI is called to initialize the TSXUCL data file which is used
4          ; to store user-defined commands.
5          ;
6          ; Outputs:
7          ;   TSXUCL data file is initialized.
8          ;   UCLBLK = Number of blocks in data file for each job.
9          ;
10         016134 010246 UCLINI: MOV     R2,-(SP)
11         016136 010346      MOV     R3,-(SP)
12         ;
13         ; Determine if TSXUCL data file is needed
14         ;
15         016140 105737 000000G      TSTB     VU$CL      ; Is TSXUCL being used at all?
16         016144 001536      BEQ      9$              ; Br if not
17         016146 013702 000000G      MOV     VUCLMC,R2    ; Get maximum number of commands
18         016152 001533      BEQ      9$              ; Br if none allowed
19         ;
20         ; Determine number of blocks needed in data file for each job
21         ;
22         016154 012700 000000G      MOV     #UK$$SZ,R0    ; Size of each keyword descriptor
23         016160 062700 000000G      ADD     #US$$SZ,R0    ; Size of each command string descriptor
24         016164 070200      MUL     R0,R2                ; Compute total # bytes for keywords+commands
25         016166 062703 000777G      ADD     #UC$$SZ+511.,R3 ; Add space for control information & round up
26         016172 005502      ADC     R2                  ; Propagate carry
27         016174 071227 001000      DIV     #512.,R2       ; Convert to # of blocks needed
28         016200 010237 000000G      MOV     R2,UCLBLK     ; Save number of blocks needed per job
29         ;
30         ; Multiply by number of jobs to get total file size
31         ;
32         016204 070227 000000C      MUL     #<LSTSL/2>,R2  ; Times total number of jobs
33         ;
34         ; Load Rt-11 device handler for ind swap file.
35         ;
36         016210 013700 000000G      MOV     UCLDAT,R0     ; Get name of the device
37         016214 004737 027506'      CALL    RTFTCH       ; Try to fetch the RT-11 device handler
38         016220 103523      BCS     11$                 ; Br if error on handler fetch
39         ;
40         ; The total required file size is now in R3.
41         ; See if the file already exists.
42         ;
43         016222      .LOOKUP #AREA,#1,#UCLDAT ; See if the file exists now
44         016242 103415      BCS     1$                 ; Br if file does not exist
45         016244 020003      CMP     R0,R3                ; Is existing file of correct size?
46         016246 001446      BEQ     2$                 ; Br if yes -- use the old file
47         016250      .PURGE #1                  ; Purge the channel
48         016256      .DELETE #AREA,#1,#UCLDAT ; Delete the old file
49         ;
50         ; Create a new data file
51         ;
52         016276      1$: .ENTER #AREA,#1,#UCLDAT,R3 ; Create new data file
53         016322 103452      BCS     10$                ; Br if error creating the file
54         016324 005303      DEC     R3                  ; Get # of last block in the file
55         016326      .WRITW #AREA,#1,WRKBUF,#256.,R3 ; Write to last block of file
56         ;
57         ; Translate possible logical device name to physical name and close

```

```
58      ; (Physical name is needed for TSXUCL program.)
59      ;
60 016364      2$:      .CSTAT  #AREA,#1,#NFSBLK      ;GET CHANNEL STATUS INFORMATION
61 016404      MOV      <NFSBLK+12>,R2      ;FETCH DEVICE NAME IN RAD50
62 016410      ADD      <NFSBLK+10>,R2      ;ADD IN DEVICE UNIT NUMBER
63 016414      ADD      #^R 0,R2      ;CONVERT UNIT NUMBER TO RAD50
64 016420      MOV      R2,UCLDAT      ;SET PHYSICAL NAME BACK INTO TSGEN CELL
65 016424      .CLOSE  #1      ;Close the file
66 016432      .RELEAS #UCLDAT      ;Release device handler
67      ;
68      ; Finished
69      ;
70 016442      9$:      MOV      (SP)+,R3
71 016444      MOV      (SP)+,R2
72 016446      RETURN
73      ;
74      ; Error creating the data file
75      ;
76 016450      10$:     .PRINT  #TSXHD      ;Print error message
77 016456      .PRINT  #UCLOPN
78 016464      CALL    SPNEED      ;Print info about number of blocks needed
79      ;
80      ; Error: Invalid device specification.
81      ;
82 016470      11$:     MOV      R0,R1      ;Save device name
83 016472      .PRINT  #TSXHD      ;Print error message
84 016500      .PRINT  #UCLOPN
85 016506      CALL    BADDEV      ;Print invalid device specification
```

MEMINI -- Initialize memory management

```

1          .SBTTL  MEMINI -- Initialize memory management
2          ; -----
3          ; Initialize memory management registers for a 1-to-1 mapping.
4          ; But leave memory management turned off.
5          ;
6 016512 010146 MEMINI: MOV      R1,-(SP)
7 016514 010246      MOV      R2,-(SP)
8 016516 010346      MOV      R3,-(SP)
9 016520 010446      MOV      R4,-(SP)
10 016522 010546      MOV      R5,-(SP)
11          ;
12          ; Initialize all pages for a 1-to-1 mapping.
13          ;
14 016524 012700 000000G 12$:  MOV      #KPAR0,R0      ; Kernel mode PAR 0
15 016530 012701 000000G      MOV      #UPAR0,R1      ; User mode PAR 0
16 016534 012702 000000G      MOV      #KPDR0,R2      ; Kernel mode PDR 0
17 016540 012703 000000G      MOV      #UPDR0,R3      ; User mode PDR 0
18 016544 012704 000010      MOV      #8,R4           ; Initialize 8 pages
19 016550 005005      CLR      R5                     ; Set initial PAR value
20 016552 010520 2$:      MOV      R5,(R0)+           ; Set kernel PAR
21 016554 010521      MOV      R5,(R1)+           ; Set user PAR value
22 016556 012722 077406      MOV      #077406,(R2)+   ; Set kernel PDR
23 016562 012723 077406      MOV      #077406,(R3)+   ; Set user PDR value
24 016566 062705 000200      ADD      #200,R5         ; Advance block number
25 016572 077411      SOB      R4,2$                 ; Init all pages
26          ;
27          ; Map kernel mode I/O page (160000) to 17760000.
28          ;
29 016574 012737 000000G 000000G      MOV      #IOPAGE,@#KPAR7 ; Map I/O page
30          ;
31          ; Finished
32          ;
33 016602 012605      MOV      (SP)+,R5
34 016604 012604      MOV      (SP)+,R4
35 016606 012603      MOV      (SP)+,R3
36 016610 012602      MOV      (SP)+,R2
37 016612 012601      MOV      (SP)+,R1
38 016614 000207      RETURN

```

MEMTST -- Set up information about available memory space

```

1          .SBTTL  MEMTST -- Set up information about available memory space
2          -----
3          ; MEMTST is called to set up information related to memory management.
4          ; MEMTST performs the following functions:
5          ;   1. Determine how much memory is installed on machine.
6          ;   2. Load Kernel mode mapping registers.
7          ;
8          ; Inputs:
9          ;   R5 = top of memory currently allocated for TSX and low memory buffers.
10         ;
11         ; Outputs:
12         ;   PHYMEM = 64-byte block # above top of physical memory.
13         ;   FMEMHI = 64-byte block # above top of memory available for system.
14         ;   Kernel mode mapping registers loaded.
15         ;   Memory management is left turned off.
16         ;
17         ;
18         ;
19         ; Offset word to test for memory wrap - choose a location which will not
20         ; effect RT-11 or TSX-Plus initialization.
21         ;
22         000110      TSTWRD = 110          ; Offset word to test for memory wrap
23
24 016616 010146      MEMTST: MOV      R1, -(SP)
25 016620 010246      MOV      R2, -(SP)
26 016622 010346      MOV      R3, -(SP)
27 016624 010446      MOV      R4, -(SP)
28 016626 010546      MOV      R5, -(SP)
29 016630 013746 000004      MOV      @#4, -(SP)      ; Save illegal mem. ref. trap vector
30
31         ; Determine if this machine has a memory management register # 3.
32         ; If it does not, then machine cannot possibly have more than 256Kb.
33         ;
34 016634 012737 017170' 000004      MOV      #TRCSET, @#4      ; Catch trap
35 016642 000240      NOP                      ; Clean out 11/73 pipeline
36 016644 000240      NOP                      ; Before attempting trap
37 016646 005737 000000G      TST      @#SR3MMR      ; Try to access status register 3
38 016652 103402      BCS      22$      ; Br if MMU 3 status register is non-existent
39 016654 105237 000000G      INCB     SR3FLG      ; No trap. We must have SR3
40
41         ;
42         ; If we are running on a Professional, there is a register that tells
43         ; us how much memory is installed on the machine.
44         ;
45 016660      22$:
46         .IF      NE, PROASM
47         TSTB     PROFLG      ; Are we running on a Professional?
48         BEQ      26$      ; Br if not
49         CLR      R5          ; Load byte without sign extension
50         BISB     @#173050, R5 ; Get 32Kb top of system RAM boundary
51         ASH      #9, R5      ; Convert to # 64 byte blocks
52         SUB      #10, R5     ; Don't use the last 512 bytes of memory
53         BR       7$
54         26$:
55         .ENDC      ; NE, PROASM
56         .IF      NE, <PROASM-1>      ; Assemble if could be on a PDP-11
57         ;
58         ; We are not running on a Professional.

```

```

58          ; Test each page above TSX to see where the top of memory is.
59          ;
60 016660 012737 017200' 000020      MOV    #RTNKM,@#20      ; Use IOT instruction to get out of user mode
61 016666 005037 000022              CLR    @#22
62 016672 052737 000000G 000000G    BIS    #MMENBL,@#SR0MMR; Enable memory management
63 016700 105737 000000G          TSTB   SR3FLG          ; Does maching have mem management reg # 3?
64 016704 001403              BEQ     4$                ; Br if non-existent
65 016706 052737 000000G 000000G    BIS    #EMMAP,@#SR3MMR ; Enable 22-bit extended memory
66          ;
67          ; Map user page 7 to each successive 256-word block and attempt to access.
68          ;
69 016714 012705 002000      4$: MOV    #1024,R5          ; Start checking at 64Kb
70 016720 010537 000000G    5$: MOV    R5,@#UPAR7        ; Map user page 7 to page to be tested
71 016724 052737 000000G 000000G    BIS    #UMODE,@#PSW    ; Go into user mode
72 016732 005737 160000      TST     @#160000            ; Can we access the page?
73          ;
74          ; Use IOT to get back into kernel mode.
75          ;
76 016736 000004              IOT                     ; Return to kernel mode
77 016740 103405              BCS     6$                ; Br if memory is non-existent
78 016742 062705 000010      ADD     #10,R5              ; Go try next page
79 016746 020527 177600      CMP     R5,#177600          ; Don't enter I/O page
80 016752 103762              BLD     5$
81          ;
82          ; Check for potential memory wrap (on 18-bit 256K byte computers).
83          ;
84 016754 020527 010000    6$: CMP     R5,#10000          ; Is physical memory above 256K bytes
85 016760 101421              BLOS    7$                ; Br if below 256K bytes
86 016762 005037 000110      CLR     @#TSTWRD           ; Clear physical location
87 016766 012737 010000 000000G    MOV    #10000,@#UPAR7 ; Map to 256K byte boundary
88 016774 052737 000000G 000000G    BIS    #UMODE,@#PSW    ; Go into user mode
89 017002 012737 177777 160110      MOV    #-1,@#160000+TSTWRD ; Store -1 at 256K physical location
90 017010 000004              IOT                     ; Return to kernel mode
91 017012 005737 000110      TST     @#TSTWRD           ; Test physical location
92 017016 001402              BEQ     7$                ; Br if physical location is clear
93 017020 012705 010000      MOV     #10000,R5          ; Constrain memory to 256K byte total
94          . ENDC      ; NE,<PROASM-1>
95          ;
96          ; Reached end of available memory.
97          ;
98 017024 010537 000000G    7$: MOV     R5,PHYMEM        ; set physical memory size
99 017030 020537 000000G      CMP     R5,MAPSIZ          ; Constrain kernel to user specified cutoff
100 017034 101402              BLOS    8$                ; Br if below user specified
101 017036 013705 000000G      MOV     MAPSIZ,R5          ; Only use this much memory
102 017042 010537 000000G    8$: MOV     R5,$MEMSZ        ; Set # 64-byte blocks of total memory
103 017046 010537 000134'      MOV     R5,FMEMHI         ; Save base 64-byte block # of top of free mem
104          ;
105          ; Turn off memory management
106          ;
107 017052 105737 000000G      TSTB   SR3FLG          ; Do we have memory management reg # 3?
108 017056 001403              BEQ     9$                ; Br if non-existent
109 017060 042737 000000G 000000G    BIC    #EMMAP,@#SR3MMR ; Disable extended memory management
110 017066 042737 000000G 000000G    9$: BIC    #MMENBL,@#SR0MMR; Turn off memory management
111          ;
112          ; If this is a Q-bus machine with >256Kb then set EXTLSI flag in ICONFG
113          ;
114 017074 023727 000134' 010000      CMP     FMEMHI,#4096. ; Does machine have at least 256Kb?

```

MEMTST -- Set up information about available memory space

```

115 017102 103411          BLD      25$          ;Br if not
116 017104 105237 0000000  INCB     MEM256       ;Remember machine has at least 256kb
117 017110 123727 0000000 0000000  CMPB     VBUSTP,#QBUS   ;Is this a Q-bus machine?
118 017116 001003          BNE      25$          ;Br if not
119 017120 052737 000001 000306'  BIS      #EXTLSI,ICONFG ;Set extended-LSI flag in ICONFG
120
121          ; See if this machine needs UNIBUS mapping
122
123 017126 123727 0000000 0000000 25$:  CMPB     VBUSTP,#UNIBUS ;Is this a UNIBUS machine?
124 017134 001005          BNE      29$          ;Br if not
125 017136 105737 0000000          TSTB     MEM256       ;Does machine have at least 256kb of memory?
126 017142 001402          BEQ      29$          ;Br if not
127 017144 105237 0000000          INCB     UBUSMP       ;Say UNIBUS mapping is needed
128
129          ; Finished
130
131 017150 012637 000004      29$:  MOV      (SP)+,@#4      ;Reset trap vector
132 017154 012605          MOV      (SP)+,R5
133 017156 012604          MOV      (SP)+,R4
134 017160 012603          MOV      (SP)+,R3
135 017162 012602          MOV      (SP)+,R2
136 017164 012601          MOV      (SP)+,R1
137 017166 000207          RETURN
138
139          ; Trap - return with C-bit set.
140
141 017170 052766 000001 000002  TRCSET: BIS      #1,2(SP)      ;Set c-bit for return
142 017176 000002          RTI          ;Return from trap
143
144          ; IOT - return at kernel mode with c-bit preserved.
145
146 017200 042766 0000000 000002  RTNKM: BIC      #UMODE,2(SP)   ;Clear user mode - return to kernel
147 017206 000002          RTI          ;Return from trap
148
149          ; Error: System does not have memory management hardware.
150
151 017210          NOXM:  .PRINT  #TSXHD      ;PRINT ERROR MESSAGE
152 017216          .PRINT  #NXMMMSG
153 017224 000137 004216'      JMP      INISTP      ;ABORT INITIALIZATION

```

```

1          .SBTTL  CXTALC -- Set up info about job context area
2          ;-----
3          ; Set up information about the size of the job context area.
4          ;
5          ; Outputs:
6          ;   CXTWDS = Number of words needed for job context area.
7          ;   CXTPAG = Number of 512-byte pages needed for context area.
8          ;   CXTPDR = Value to load into PDR when mapping job context area.
9          ;   CXTRMN = Address of simulated RMON in context area.
10         ;   RMNPDR = Value to load into PDR to map to simulated RMON.
11         ;
12 017230   CXTALC:
13         ;
14         ; Get size of base portion of job context area
15         ;
16 017230   012700   000000G      MOV      #CXTSIZ,RO      ;Get # bytes for base context area
17         ;
18         ; Bound up to 64-byte boundary and add size of simulated RMON
19         ; which is allocated above the base job context data.
20         ;
21 017234   062700   000077      ADD      #63.,RO      ;Bound up to 64 byte boundary
22 017240   042700   000077      BIC      #77,RO
23 017244   010037   000000G      MOV      RO,CXTRMN      ;Offset to start of simulated RMON
24 017250   062737   000000G 000000G  ADD      #CXTBAS,CXTRMN ;Add base virtual address of context area
25 017256   062700   000001G      ADD      #MVSIZ+1,RO    ;Add space for simulated RMON & channels
26         ;
27         ; Save number of words needed for context area
28         ;
29 017262   006200           ASR      RO      ;Convert to # words
30 017264   010037   000000G      MOV      RO,CXTWDS      ;This is # words for whole job context area
31         ;
32         ; Compute PDR value to use to map to job context area
33         ;
34 017270   062700   000037      ADD      #31.,RO      ;Bound up to # 32 word units
35 017274   072027   177773      ASH      #-5.,RO      ;Get # 32-word units for context area
36 017300   000300           SWAB      RO      ;Put # 32-word units in high-order byte
37 017302   052700   000006      BIS      #6,RO      ;Set PDR control flags
38 017306   010037   000000G      MOV      RO,CXTPDR      ;This is the PDR value
39         ;
40         ; Compute # 512-byte pages needed for job context block
41         ;
42 017312   013700   000000G      MOV      CXTWDS,RO      ;Get back # words for context area
43 017316   062700   000377      ADD      #255.,RO      ;Bound up to # 256-word blocks
44 017322   072027   177770      ASH      #-8.,RO      ;Get # 256-word pages for context area
45 017326   010037   000000G      MOV      RO,CXTPAG      ;# pages for job context area
46         ;
47         ; Set up PDR value used when mapping to simulated RMON
48         ;
49 017332   012700   000000G      MOV      #MVSIZ,RO      ;Get size of monitor vector table
50 017336   062700   000077      ADD      #63.,RO      ;Round up to # 32 word blocks
51 017342   072027   177772      ASH      #-6,RO      ;Cvt to # 32-word blocks
52 017346   005300           DEC      RO      ;Get # blocks - 1
53 017350   000300           SWAB      RO      ;Put # blocks in left byte
54 017352   052700   000006      BIS      #6,RO      ;Allow read and write access
55 017356   042700   100261      BIC      #100261,RO      ;Make sure unused PDR bits are zero
56 017362   010037   000000G      MOV      RO,RMNPDR      ;This is PDR value to map to sim. RMON
57

```

TSINIT -- TSX startup initializ MACRO V05.04 Monday 14-Dec-87 08:35 Page 45-1
CXTALC -- Set up info about job context area

58 ; Finished
59 ;
60 017366 000207 RETURN

```

1          .SBTTL  MAPALC -- Allocate memory usage table
2          ;-----
3          ; MAPALC is called to allocate a table that keeps track of which pages
4          ; of memory are currently in use by user jobs and which are free.
5          ; Each byte in the table corresponds to a 512-byte block of physical memory.
6          ; The portion of physical memory used by the system is not represented
7          ; in the memory allocation table.
8          ;
9          ; Inputs:
10         ; R5      = 64-byte block number of top of free memory area.
11         ; FMEMLO = 64-byte block number of base of free memory area.
12         ;
13         ; Outputs:
14         ; FMEMHI = 64-byte block number of top of free memory area.
15         ; MAPPAR = 64-byte block number used to map to the memory alloc table.
16         ; BASMAP = Virtual address of memory allocation table that would
17         ;           correspond to physical address 0. Note, the entries
18         ;           in the allocation table between BASMAP and LOMAP are
19         ;           actually not allocated.
20         ; LOMAP  = Virtual address of memory allocation table that corresponds
21         ;           to 1st physical 512-byte page that is available to user jobs.
22         ;           Note, LOMAP always contains 120000 because we access the
23         ;           allocation table by mapping it through PAR 5.
24         ; HIMAP  = Virtual address of memory allocation table that corresponds
25         ;           to 512-byte page above the top of the user area.
26         ;
27 017370 010246 MAPALC: MOV      R2, -(SP)
28 017372 010346      MOV      R3, -(SP)
29 017374 010446      MOV      R4, -(SP)
30
31         ; Determine how many bytes will be required for the memory allocation table.
32         ; One byte in the table is required for each 512-byte physical page.
33         ;
34 017376 010503      MOV      R5, R3          ; Get 64-byte block # of top of free mem
35 017400 072327 177775 ASH      #-3, R3       ; Convert to 512-byte page #
36 017404 042703 160000 BIC      #160000, R3    ; Kill possible sign extension
37 017410 013702 000136' MOV      FMEMLO, R2      ; Get 64-byte block # of base of free memory
38 017414 062702 000007 ADD      #7, R2        ; Round up
39 017420 072227 177775 ASH      #-3, R2       ; Convert to 512-byte page #
40 017424 042702 160000 BIC      #160000, R2    ; Kill possible sign extension
41 017430 160203      SUB      R2, R3          ; Get # bytes needed for allocation table
42 017432 003440      BLE      10$,           ; Br if memory overflow
43 017434 010304      MOV      R3, R4          ; Get # bytes for allocation table
44 017436 062704 001000 ADD      #512, R4        ; Add 1 extra byte and round up to 512-byte
45 017442 072427 177767 ASH      #-9, R4        ; Get # 512-byte units needed for alloc table
46
47         ; Set up virtual address pointers for the allocation table
48         ;
49 017446 012700 0000000 MOV      #VPAR5, R0      ; We will map to alloc table through PAR 5
50 017452 010037 0000000 MOV      R0, LOMAP      ; Pointer to 1st entry in alloc table
51 017456 160200      SUB      R2, R0          ; Get pseudo virtual address for page # 0
52 017460 010037 0000000 MOV      R0, BASMAP      ; This would point to alloc entry for page 0
53 017464 012700 0000000 MOV      #VPAR5, R0      ; Get back base address of table
54 017470 060300      ADD      R3, R0          ; Add # bytes used by table
55 017472 160400      SUB      R4, R0          ; Subtract space used by table itself
56 017474 010037 0000000 MOV      R0, HIMAP      ; Virtual address of 1st entry for system page
57

```

```

58          ; Allocate space for the allocation table
59          ;
60 017500 072427 000003          ASH      #3,R4          ;Get # 64-byte units for alloc table
61 017504 160405          SUB      R4,R5          ;Compute physical 64-byte base for table
62 017506 020537 000136'        CMP      R5,FMEMLO      ;Did we run out of memory space?
63 017512 101410          BLOS     10$          ;Br if memory overflow
64 017514 010537 000000G        MOV      R5,MAPPAR      ;Use this value to map PAR 5 to alloc table
65 017520 010537 000134'        MOV      R5,FMEMHI      ;Save new top of free memory area
66          ;
67          ; Finished
68          ;
69 017524 012604          MOV      (SP)+,R4
70 017526 012603          MOV      (SP)+,R3
71 017530 012602          MOV      (SP)+,R2
72 017532 000207          RETURN
73          ;
74          ; Error: Generated system is too large
75          ;
76 017534          10$: .PRINT  #TSXHD          ;Print error message heading
77 017542          .PRINT  #PHSOVF          ;Physical memory overflow
78 017550 000137 004216'        JMP      INISTP          ;Abort the initialization

```

SETJSZ -- Set up information about maximum job sizes

```

1          .SBTTL  SETJSZ -- Set up information about maximum job sizes
2          ;-----
3          ; SETJSZ is called to set up some information about the maximum
4          ; job sizes to be allowed. The maximum job size is chosen so that
5          ; we are guaranteed to be able to get at least one job logged on.
6          ;
7          ; Inputs:
8          ;   LOMAP = Address of 1st MEMMAP entry available to user jobs.
9          ;   HIMAP = Address of 1st MEMMAP entry above top of user job area.
10         ;
11         ; Outputs:
12         ;   FREPGS = Total number of 512-byte pages available to user jobs.
13         ;   MXJMEM = max # K bytes available to a job
14         ;   DFJMEM = Default job memory size (kb)
15         ;
16 017554 010546 SETJSZ: MOV      R5, -(SP)
17         ;
18         ; Determine total number of pages of memory available to user jobs
19         ;
20 017556 013705 000000G      MOV      HIMAP, R5      ; POINTER ABOVE LAST FREE PAGE ENTRY
21 017562 163705 000000G      SUB      LOMAP, R5      ; GET TOTAL # OF FREE PAGES
22 017566 010537 000000G      MOV      R5, FREPGS      ; # FREE PAGES AVAILABLE TO USER JOBS
23         ;
24         ; Make sure there is enough free space to run TSKMON.
25         ;
26 017572 020537 000000G      CMP      R5, KMNPGS      ; COMPARE # FREE PAGES TO # PAGES FOR TSKMON
27 017576 103436      BLO      1$      ; BR IF INSUFFICIENT MEMORY TO RUN TSKMON
28         ;
29         ; Set up max memory limit for jobs
30         ;
31 017600 013700 000000G      MOV      CXTPAG, R0      ; # PAGES NEEDED FOR JOB CONTEXT AREA
32 017604 010037 000000G      MOV      R0, JCXPGS
33 017610 160005      SUB      R0, R5      ; LEAVE ROOM FOR JOB CONTEXT AREA
34 017612 006205      ASR      R5      ; Convert # pages to # KB
35 017614 020537 000000G      CMP      R5, VHIMEM      ; COMPARE WITH TSGEN SPECIFIED MAX SIZE
36 017620 101402      BLOS      10$      ; BR IF CONSTRAINED BY PHYSICAL SIZE
37 017622 013705 000000G      MOV      VHIMEM, R5      ; LIMIT BY VALUE SPECIFIED IN TSGEN
38 017626 010537 000000G 10$: MOV      R5, MXJMEM      ; MAX # K BYTES AVAILABLE TO A JOB
39 017632 010500      MOV      R5, R0
40 017634 072027 000012      ASH      #10., R0      ; CONVERT # KB TO BYTE ADDRESS
41 017640 001002      BNE      2$      ; BR IF DIDN'T OVERFLOW 64KB
42 017642 012700 177774      MOV      #177774, R0      ; GET 64KB TOP ADDRESS
43 017646 010037 000000G 2$: MOV      R0, MXJADR      ; ADDRESS ABOVE TOP OF JOB
44         ;
45         ; Set default memory size of jobs
46         ;
47 017652 020537 000000G      CMP      R5, VDFMEM      ; COMPARE TO DEFAULT SPECIFIED IN TSGEN
48 017656 101402      BLOS      11$      ; BR IF CONSTRAINED BY PHYSICAL LIMIT
49 017660 013705 000000G      MOV      VDFMEM, R5      ; CONSTRAIN BY TSGEN PARAMETER
50 017664 010537 000000G 11$: MOV      R5, DFJMEM      ; SET DEFAULT JOB MEMORY SIZE
51         ;
52         ; Finished
53         ;
54 017670 012605      MOV      (SP)+, R5
55 017672 000207      RETURN
56         ;
57         ; Error -- Insufficient memory space available to run TSKMON.

```

TSINIT -- TSX startup initializ MACRO V05.04 Monday 14-Dec-87 08:35 Page 47-1
SETJSZ -- Set up information about maximum job sizes

58

59 017674 004737 027642'

1\$:

CALL SIZERR

;Generated system is too big -- abort

SETJSZ -- Set up information about maximum job sizes

```

1          . IF      NE,<PROASM-1>      ; No parity control if PRO only
2          . SBTTL   PARSET -- Setup memory parity control
3          ; -----
4          ; PARSET is called to set up memory parity control.
5          ; Currently this consists of disabling memory parity.
6          ;
7 017700 005727 000000G PARSET: TST      #MPARFL      ; Does he want to disable memory parity?
8 017704 001027          BNE      20$          ; Br if not
9 017706 010246          MOV      R2,-(SP)
10 017710 013746 000004 MOV      @#4,-(SP)      ; Save contents of trap vector
11          ;
12          ; Catch traps that occur when we access unimplemented parity registers
13          ;
14 017714 012737 017740' 000004 MOV      #2@, @#4      ; Send traps to 2$
15 017722 000240          NOP                      ; Clean out instruction pipeline
16 017724 000240          NOP
17          ;
18          ; Disable parity for each block of memory
19          ;
20 017726 012702 000000G          MOV      #MPARO,R2      ; Point to 1st memory control register
21 017732 042712 000000G 1$:      BIC      #PARENL,(R2)    ; Disable memory parity
22 017736 000402          BR       3$              ; We did not trap
23 017740 062706 000004 2$:      ADD      #4,SP          ; Clean trap PS and PC off of stack
24 017744 062702 000002 3$:      ADD      #2,R2          ; Point to next parity control register
25 017750 020227 000000G          CMP      R2,#MPAR16      ; Have we cleared all registers?
26 017754 101766          BLOS     1$              ; Loop if not
27          ;
28          ; Finished
29          ;
30 017756 012637 000004          MOV      (SP)+, @#4      ; Restore trap vector
31 017762 012602          MOV      (SP)+, R2
32 017764 000207 20$:      RETURN
33          . IFF      ; NE,<PROASM-1>
34 PARSET: RETURN
35          . ENDC      ; NE,<PROASM-1>

```

```

1          .SBTTL  GETHNL -- Load device handlers into memory
2          ;-----
3          ; GETHNL performs two functions:
4          ;   1. Set up information in the device tables about all devices.
5          ;   2. Load those handlers that reside in low memory.
6          ;
7          ; Inputs:
8          ;   R5 = Address of start of free memory.
9          ;
10         ; Outputs:
11         ;   R5 = Address of new start of free memory.
12         ;   NMXHAN = Number of handlers to load into extended memory.
13         ;
14 017766 010146  GETHNL: MOV     R1,-(SP)
15 017770 010246      MOV     R2,-(SP)
16 017772 010446      MOV     R4,-(SP)
17
18         ; Begin loop to check all handlers specified in TSGEN with DEVDEF.
19         ;
20 017774 005001      CLR      R1          ; Init device table index
21 017776 020127 000000C 1$:      CMP     R1,#<AHEND-AUTHAN> ; Done all devices?
22 020002 103015      BHIS     2$          ; Br if yes
23 020004 016102 000000G      MOV     AUTHAN(R1),R2 ; Get the name of the device
24 020010 001407      BEQ      3$          ; Ignore null devices
25 020012 020227 000000G      CMP     R2,#DMYDEV    ; Is this a dummy device entry?
26 020016 001404      BEQ      3$          ; Skip it if yes
27 020020 016104 000000G      MOV     DTYPE(R1),R4 ; Get flags specified in TSGEN
28
29         ; Load this handler
30
31 020024 004737 020076'      CALL     LDHAND      ; Try to load handler into memory
32
33         ; Check next device
34         ;
35 020030 062701 000002      3$:      ADD     #2,R1          ; Advance device index
36 020034 000760      BR       1$          ; See if more devices to load
37
38         ; Now see if there are spooled devices to contend with
39         ;
40 020036 012704 000000G      2$:      MOV     #SPLND,R4      ; Are there any spooled devices?
41 020042 001411      BEQ      9$          ; Br if not
42 020044 012701 000000G      MOV     #SPLDEV,R1      ; Point to spooled device name table
43 020050 012102      5$:      MOV     (R1)+,R2      ; Get the name of the next spooled device
44 020052 010446      MOV     R4,-(SP)      ; Save device count
45 020054 005004      CLR      R4          ; Say no TSGEN flags for device
46 020056 004737 020076'      CALL     LDHAND      ; Load the handler
47 020062 012604      MOV     (SP)+,R4      ; Recover the device count
48 020064 077407      SDB      R4,5$      ; Loop if more handlers to load
49
50         ; Finished
51         ;
52 020066 012604      9$:      MOV     (SP)+,R4
53 020070 012602      MOV     (SP)+,R2
54 020072 012601      MOV     (SP)+,R1
55 020074 000207      RETURN

```

```

1          .SBTTL  LDHAND -- Load a device handler
2          ;-----
3          ; LDHAND sets up the device tables for a handler and loads into memory
4          ; those handlers that reside in low memory.
5          ; The device interrupt vectors are NOT set up by LDHAND.
6          ;
7          ; Inputs:
8          ;   R2 = Rad-50 name of device.
9          ;   R4 = TSX-Plus DX$xxx status flags for device from TSGEN.
10         ;   R5 = Address where handler is to be loaded.
11         ;
12         ; Outputs:
13         ;   R5 = New free memory address.
14         ;   NUMDEV = Incremented by 2.
15         ;   PNAME(i) = Rad-50 name of device.
16         ;   ENTRY(i) = Handler entry point.
17         ;   DVSTAT(i) = Device status flags.
18         ;   DVFLAG(i) = TSX-Plus device status flags.
19         ;   HANPAR(i) = PAR offset if this is a mapped handler.
20         ;
21 020076 010446 LDHAND: MOV      R4, -(SP)
22         ;
23         ; Determine if we should ignore this device
24         ;
25 020100 004737 020246'      CALL    IN$CK1      ; Should we ignore this device?
26 020104 103456              BCS      9$          ; Br if yes
27         ;
28         ; The initial tests indicate that this handler should be loaded.
29         ; Now open the handler file and perform some additional checks.
30         ;
31 020106 004737 020424'      CALL    IN$CK2      ; Perform some additional checks on handler
32 020112 103450              BCS      8$          ; Br if we should not load this device
33         ;
34         ; At this point channel 1 is open to the handler file and block 0
35         ; of the handler is in WRKBUF.
36         ; Set up information tables for this device.
37         ;
38 020114 004737 021036'      CALL    STDVTB      ; Set up info in tables for this device
39         ;
40         ; Determine if this handler is to be loaded into low memory or
41         ; extended memory.
42         ;
43 020120 016400 000000G      MOV      DVFLAG(R4), R0 ; Get TSX-Plus status flags for device
44 020124 032700 000000G      BIT      #DX$NHM, R0   ; Are we never to map this handler?
45 020130 001035              BNE      1$           ; Br if handler cannot be mapped
46 020132 032700 000000G      BIT      #DX$MPH, R0   ; Is mapping wanted for this handler?
47 020136 001432              BEQ      1$           ; Br if not
48 020140 032700 000000G      BIT      #DX$IBH, R0   ; Does this handler have an internal I/O buff?
49 020144 001412              BEQ      2$           ; Br if not
50 020146 105737 000000G      TSTB     UBUSMP        ; Does this machine have a mapped UNIBUS?
51 020152 001024              BNE      1$           ; Br if yes -- Don't map this handler
52 020154 032700 000000G      BIT      #DX$MAP, R0   ; Does this handler require I/O mapping?
53 020160 001404              BEQ      2$           ; Br if not
54 020162 032737 000001 000306' BIT      #EXTLSI, ICONF$ ; Is this a Q-bus system with more than 256Kb?
55 020170 001015              BNE      1$           ; Br if yes -- Don't map this handler
56         ;
57         ; This handler can be mapped and will be loaded in extended memory

```

```
58
59 020172 005237 000124' 2$: INC NMXHAN ; Count # of mapped handlers
60 020176 012764 000001 000000G MOV #1,HANPAR(R4) ; Set flag saying handler should be mapped
61
62 ; Make sure size of mapped handler does not exceed 8KB
63
64 020204 026427 000000G 020000 CMP HANSIZ(R4),#8192. ; Is mapped handler too big?
65 020212 101410 BLOS B$ ; Br if not too big
66 020214 012700 002452' MOV #HN2BIG,R0 ; Get error message address
67 020220 000137 022124' JMP HLERR ; Abort initialization
68
69 ; This handler must be loaded into low memory
70
71 020224 005064 000000G 1$: CLR HANPAR(R4) ; Say this handler is not mapped
72 020230 004737 021146' CALL LDHNLO ; Load handler into low memory
73
74 ; Close the handler file
75
76 020234 B$: .CLOSE #1 ; Close the handler file
77
78 ; Finished
79
80 020242 012604 9$: MOV (SP)+,R4
81 020244 000207 RETURN
```

INSCK1 -- Determine if a handler should be installed

```

1          .SBTTL  INSCK1 -- Determine if a handler should be installed
2          ;-----
3          ;  INSCK1 is called to determe if a certain device handler should be
4          ;  loaded.
5          ;
6          ;  Inputs:
7          ;    R2 = Rad50 name of the device.
8          ;    R4 = Initial DX$xxx flags as specified in TSGEN.
9          ;
10         ;  Outputs:
11         ;    C-flag cleared ==> Load the handler.
12         ;    C-flag set      ==> Do not load the handler.
13         ;    R2 = Device name with unit number removed.
14         ;    R4 = DX$xxx combined with default flags for the device.
15         ;
16 020246 010146  INSCK1: MOV      R1, -(SP)
17         ;
18         ;  Strip off any specified unit number
19         ;
20 020250 010201          MOV      R2, R1          ; Get full device name
21 020252 005000          CLR      R0              ; Set for divide
22 020254 071027 000050   DIV      #50, R0        ; Split off last digit
23 020260 070027 000050   MUL      #50, R0        ; Now correct for divide
24 020264 010102          MOV      R1, R2          ; Get device name less 3rd digit
25 020266 010237 000146'  MOV      R2, CURNAM      ; Set name of handler being loaded
26         ;
27         ;  See if this is a device such as DK, SY, or TT which we don't
28         ;  need to load as a device handler.
29         ;
30 020272 020237 000176'  CMP      R2, R5OLD        ; Is device name LD?
31 020276 001004          BNE      1$              ; Br if not
32 020300 105737 000000G   TSTB     VLDSYS        ; Is standard system LD support included?
33 020304 001044          BNE      5$              ; Br if yes -- Don't load LD
34 020306 000417          BR       3$              ; Load LD
35 020310 012701 000224'  1$: MOV      #SKPDEV, R1   ; Point to table of devices to skip
36 020314 020221          2$: CMP      R2, (R1)+    ; Is this a device to be skipped?
37 020316 001437          BEQ      5$              ; Br if yes
38 020320 005711          TST      (R1)            ; Reached end of skip list?
39 020322 001374          BNE      2$              ; Loop if not
40         ;
41         ;  See if we have already loaded the handler for this device
42         ;
43 020324 013701 000000G   MOV      NUMDEV, R1      ; Get index for last device
44 020330 001406          BEQ      3$              ; Br if no devices installed yet
45 020332 020261 000000G   4$: CMP      R2, PNAME(R1) ; See if this device is already installed
46 020336 001427          BEQ      5$              ; Br if already installed
47 020340 162701 000002   SUB      #2, R1          ; More installed devices to check?
48 020344 003372          BGT      4$              ; Loop if yes
49         ;
50         ;  This handler is to be loaded.
51         ;  Get default TSX-Plus control flags for this device.
52         ;
53 020346 012701 000342'  3$: MOV      #DVFLBS, R1   ; Point to start of table
54 020352 020261 000000   6$: CMP      R2, DV$NAM(R1) ; Search for device in the table
55 020356 001003          BNE      7$              ; Br if this is not it
56 020360 056104 000002   BIS      DV$FLG(R1), R4   ; Combine default flags
57 020364 000405          BR       8$

```

INSCK1 -- Determine if a handler should be installed

```
58 020366 062701 000004      7$:      ADD      #DV$SZ,R1      ;Point to next entry
59 020372 020127 000522'      CMP      R1,#DVFLND      ;Checked all entries?
60 020376 103765              BLD      6$              ;Loop if not
61                          ;
62                          ; If this is a DMA device, set flag saying buffers must be on
63                          ; even byte boundaries.
64                          ;
65 020400 032704 000000G      8$:      BIT      #DX$DMA,R4      ;Is this a DMA device?
66 020404 001402              BEQ      10$              ;Br if not
67 020406 052704 000000G      BIS      #DX$EBA,R4      ;Set even-buffer-boundary flag
68                          ;
69                          ; Load this handler
70                          ;
71 020412 000241              10$:     CLC                      ;Set flag saying to load the handler
72 020414 000401              BR      9$
73                          ;
74                          ; Do not load this handler
75                          ;
76 020416 000261              5$:      SEC                      ;Set flag saying not to load the handler
77                          ;
78                          ; Finished
79                          ;
80 020420 012601              9$:      MOV      (SP)+,R1
81 020422 000207              RETURN
```

```

1          .SBTTL  INSCK2 -- Additional checking for handler installation
2          ;-----
3          ;  INSCK2 is called to determine if a device handler should be installed.
4          ;
5          ;  Inputs:
6          ;    R2 = Rad50 device name (without unit number).
7          ;
8          ;  Outputs:
9          ;    C-flag cleared ==> Load this handler.
10         ;    C-flag set      ==> Do not load this handler.
11         ;    If the handler is to be loaded, its block 0 is in WRKBUF and channel
12         ;    number 1 is opened to the handler file.
13         ;
14 020424 010146  INSCK2: MOV      R1,-(SP)
15 020426 010246      MOV      R2,-(SP)
16 020430 010346      MOV      R3,-(SP)
17 020432 010446      MOV      R4,-(SP)
18 020434 010546      MOV      R5,-(SP)
19 020436 013746 000004      MOV      @#4,-(SP)      ; Save the bus timeout vector
20 020442 013746 000010      MOV      @#10,-(SP)     ; Save illegal instruction vector
21
22         ;  Try to lookup handler file on system disk
23         ;
24 020446 010237 000246'      MOV      R2,HANNAM+2      ; Set the device name for the lookup
25 020452          .LOOKUP #AREA,#1,#HANNAM; Try to open the handler file
26 020472 103011      BCC      1$      ; Br if we found the handler file
27
28         ;  Error -- Cannot find handler file
29         ;
30 020474 105737 000000G      TSTB     VINABT          ; Abort or continue on errors?
31 020500 001002          BNE      2$      ; Br if abort on errors
32 020502 000137 021004'      JMP      10$          ; Say not to load this handler
33 020506 012700 001636' 2$:      MOV      #CFHMSG,R0      ; Can't find handler
34 020512 000137 022124'      JMP      HLERR          ; Abort initialization
35
36         ;  We were able to open the handler file.
37         ;  Read in block 0 of handler.
38         ;
39 020516          1$:      .READW   #AREA,#1,WRKBUF,#256.,#0 ; Read block 0 into WRKBUF
40 020554 103004          BCC      3$      ; Br if read ok
41 020556 012700 001700'      MOV      #ERHMSG,R0      ; Error during read
42 020562 000137 022124'      JMP      HLERR
43
44         ;  Determine if the handler is supported under the current RT-11 version
45         ;
46 020566 012700 000330' 3$:      MOV      #HVTBL,R0      ; Point to table with handler version info
47 020572 013701 000152'      MOV      WRKBUF,R1      ; Point to buffer with block 0 of handler
48 020576 116101 000000G      MOVB     H.DSTS(R1),R1    ; Get device ID code from handler
49 020602 120160 000000      51$:      CMPB     R1,HV$ID(R0) ; Compare handler ID code with table entry
50 020606 001020          BNE      53$      ; Br if this entry not for this handler
51 020610 123760 000000G 000001      CMPB     SYSVER,HV$VER(R0); Compare curr RT-11 vers with min acceptable
52 020616 103405          BLO      52$      ; Br if version is not adequate
53 020620 101020          BHI      54$      ; Br if version is ok
54 020622 123760 000000G 000002      CMPB     SYSUPD,HV$UPD(R0); Compare update level
55 020630 103014          BHIS     54$      ; Br if update level is ok
56 020632 105737 000000G 52$:      TSTB     VINABT      ; Handler not loadable - abort or continue?
57 020636 001462          BEQ      10$      ; Br if continue but don't allow loading

```

```

58 020640 012700 001744'      MOV      #ERHNDV,RO      ;Wrong version of RT for handler
59 020644 000137 022124'      JMP      HLERR        ;Report error and abort
60 020650 062700 000003      53$:  ADD      #HV$SZ,RO      ;Point to next handler version table entry
61 020654 020027 000341'      CMP      RO,#HVEND        ;Are there more entries?
62 020660 103750              BLD      51$             ;Loop if more to check
63                          ;
64                          ; Check handler sysgen options
65                          ;
66 020662 013700 000152'      54$:  MOV      WRKBUF,RO      ;Point to buffer with handler block 0
67 020666 032760 000000G 000000G  BIT      #SG$MMU,H.GEN(RO);Was handler genned with XM support?
68 020674 001004              BNE      4$              ;Br if yes
69 020676 012700 002045'      MOV      #HSGER,RO      ;Error if not XM version of handler
70 020702 000137 022124'      JMP      HLERR
71 020706 016037 000000G 000304' 4$:  MOV      H.GEN(RO),HGENFL;Save handler sysgen flags for later
72                          ;
73                          ; Check the CSR address specified in the handler to see if the
74                          ; hardware device for this handler exists.
75                          ;
76 020714 012737 017170' 000004      MOV      #TRCSET,@#4      ;Catch bus timeout traps
77 020722 012737 017170' 000010      MOV      #TRCSET,@#10     ;Catch illegal instruction traps
78 020730 013700 000152'      MOV      WRKBUF,RO      ;Point to start of block 0 of handler
79 020734 016001 000000G      MOV      H.CSR(RO),R1      ;Get address of CSR for device
80 020740 001402              BEQ      5$              ;Br if no CSR specified
81 020742 005711              TST      (R1)             ;Is CSR accessible?
82 020744 103422              BCS      13$            ;Br if trap occurred while accessing CSR
83                          ;
84                          ; Execute the device installation code.
85                          ; The installation code will set the C-flag if the handler should
86                          ; not be loaded.
87                          ;
88 020746 062700 000000G      5$:  ADD      #H.INS,RO      ;Offset 200 in block 0
89 020752 005710              TST      @RO              ;Does any installation code exist?
90 020754 001415              BEQ      11$            ;Br if no driver installation code
91 020756 013746 000000G      MOV      @#RMON,-(SP)      ;Save RT-11 RMON pointer
92 020762 012737 000000G 000000G  MOV      #MONVEC,@#RMON ;Set TSX-Plus RMON pointer
93 020770 013703 000000G      MOV      RPRVEC,R3        ;Get pointer to Pro vec addr routine
94 020774 004710              CALL     @RO              ;Call the installation code
95 020776 012637 000000G      MOV      (SP)+,@#RMON      ;Restore RT-11 RMON pointer
96 021002 000403              BR       13$             ;C-flag now indicates handler load status
97                          ;
98                          ; Finished with installation verification.
99                          ;
100 021004 000261      10$:  SEC                      ;Set c-bit for indicating handler
101 021006 000401              BR       13$            ;should not be installed
102 021010 000241      11$:  CLC                      ;Clear the c-bit for driver installation
103 021012 012637 000010      13$:  MOV      (SP)+,@#10     ;Restore illegal instruction vector
104 021016 012637 000004      MOV      (SP)+,@#4        ;Restore the bus timeout vector
105 021022 012605      MOV      (SP)+,R5
106 021024 012604      MOV      (SP)+,R4
107 021026 012603      MOV      (SP)+,R3
108 021030 012602      MOV      (SP)+,R2
109 021032 012601      MOV      (SP)+,R1
110 021034 000207      RETURN

```

STDVTB -- Set up device table entries for a device

```

1          .SBTTL  STDVTB -- Set up device table entries for a device
2          ;-----
3          ; STDVTB is called to set up device table entries for a device whose
4          ; handler is being loaded.
5          ;
6          ; Inputs:
7          ;   R2 = Rad50 name of device (less unit number).
8          ;   R4 = DX$xxx device flags for DVFLAG table.
9          ;   Block 0 of the handler must be in WRKBUF.
10         ;
11         ; Outputs:
12         ;   R4 = Device table index number for this device.
13         ;   NUMDEV = Incremented by 2.
14         ;   PNAME(i) = Rad50 name of the device.
15         ;   DVSTAT(i) = Device status flags.
16         ;   DVFLAG(i) = TSX-Plus control flags.
17         ;   HANSIZ(i) = Size of handler (bytes).
18         ;   DEVSIZ(i) = Size of device (blocks).
19         ;
20 021036   STDVTB:
21         ;
22         ; Increment device counter
23         ;
24 021036   062737 000002 000000G      ADD     #2,NUMDEV      ; Say another device added to tables
25 021044   013700 000000G      MOV      NUMDEV,R0          ; Get device index number
26         ;
27         ; Set up PNAME and DVFLAG.
28         ;
29 021050   010260 000000G      MOV      R2,PNAME(R0)        ; Set permanent device name
30 021054   010460 000000G      MOV      R4,DVFLAG(R0)       ; Set up TSX-Plus control flags for the device
31         ;
32         ; Set HANDSK entry to 1.
33         ; This entry is supposed to hold the absolute block number on the disk where
34         ; block 1 of the handler is located. We set to 1 because all I/O we do
35         ; on behalf of the handler is relative to the base of the handler rather than
36         ; relative to the start of the disk.
37         ;
38 021060   012760 000001 000000G      MOV      #1,HANDSK(R0) ; Set block # of block 1 of handler file
39         ;
40         ; Extract parameters from handler block 0
41         ;
42 021066   010004      MOV      R0,R4          ; Carry device index in R4
43 021070   013700 000152'      MOV      WRKBUF,R0      ; Point to block 0 of handler
44 021074   016064 000000G 000000G      MOV      H.SIZ(R0),HANSIZ(R4) ; Set handler size
45 021102   016064 000000G 000000G      MOV      H.DVSZ(R0),DEVSIZ(R4) ; Number of blocks on device
46 021110   016064 000000G 000000G      MOV      H.DSTS(R0),DVSTAT(R4) ; Set device status flags
47         ;
48         ; Disable MOUNTs and data caching for certain devices
49         ;
50 021116   032764 000000G 000000G      BIT      #DS$DIR,DVSTAT(R4) ; Is this a directory structured device?
51 021124   001404      BEQ      1$          ; Br if not -- No mounts allowed
52 021126   032764 000000G 000000G      BIT      #DS$NRD,DVSTAT(R4) ; Non RT-11 directory structure (mag tape)?
53 021134   001403      BEQ      9$          ; Br if not
54 021136   052764 000000C 000000G 1$:  BIS      #<DX$NMT!DX$NCA>,DVFLAG(R4) ; Disable mounts and data caching
55         ;
56         ; Finished
57         ;

```

TSINIT -- TSX startup initializ MACRO V05.04 Monday 14-Dec-87 08:35 Page 53-1
STDVTB -- Set up device table entries for a device

5B 021144 000207

9#: RETURN

```

1          .SBTTL  LDHNLO -- Load device handler into low memory
2          ;-----
3          ; LDHNLO is called to load a device handler into low memory.
4          ;
5          ; Inputs:
6          ;   R4 = Device index number.
7          ;   R5 = Address of start of free memory area.
8          ;
9          ; Outputs:
10         ;   R5 = Address of new start of free memory area.
11         ;
12 021146 010346 LDHNLO: MOV      R3,-(SP)
13         ;
14         ; Determine if we have enough free memory space to read the handler
15         ;
16 021150 005064 000000G      CLR      HANPAR(R4)      ; Say this handler is not mapped
17 021154 010500      MOV      R5,R0      ; Get current top of memory address
18 021156 016403 000000G      MOV      HANSIZ(R4),R3    ; Get size of handler
19 021162 060300      ADD      R3,R0      ; Get address above top of handler
20 021164 004737 027622'      CALL     CHKMEM      ; See if handler will fit in memory
21         ;
22         ; Handler will fit. Read it into memory.
23         ;
24 021170 006203      ASR      R3      ; Get number of words to read
25 021172      . READW  #AREA,#1,R5,R3,#1
26 021226 103004      BCC      1$      ; Br if read ok
27 021230 012700 001700'      MOV      #ERHMSG,R0      ; Error reading handler
28 021234 000137 022124'      JMP      HLERR      ; Abort initialization
29         ;
30         ; Set address of handler entry point and compute address beyond
31         ; end of the handler.
32         ;
33 021240 010564 000000G 1$: MOV      R5,HANENT(R4)      ; Set address of handler entry point
34 021244 062764 000006 000000G ADD      #6,HANENT(R4)      ; (Point to fourth word of handler)
35 021252 006303      ASL      R3      ; Convert handler size to bytes
36 021254 060305      ADD      R3,R5      ; Point beyond end of handler
37         ;
38         ; Set up table of addresses of support routines at end of handler.
39         ;
40 021256 010503      MOV      R5,R3      ; Get address past end of handler
41 021260 004737 022032'      CALL     STHNPV      ; Set up pointer vector in handler
42         ;
43         ; If handler has any load-time execution code, run it now
44         ;
45 021264 004737 022162'      CALL     DOHNLC      ; Run any load-time code for handler
46         ;
47         ; Finished
48         ;
49 021270 012603      MOV      (SP)+,R3
50 021272 000207      RETURN

```

```

1          .SBTTL  GETHNH -- Load handlers into extended memory
2          ;-----
3          ; GETHNH is called to load those handlers that can be placed in extended
4          ; memory.  The status tables for these devices have already been set up
5          ; by GETHNL.
6          ;
7          ; Inputs:
8          ;   R5 = 64-byte block number of top of free memory area.
9          ;
10         ; Outputs:
11         ;   R5 = 64-byte block number of new top of free memory area.
12         ;
13 021274 010446  GETHNH: MOV      R4, -(SP)
14         ;
15         ; Begin looking for handlers that are to be loaded into extended memory.
16         ; GETHNL stored a non-zero (but meaningless) value in the HANPAR entry
17         ; for each handler that is to be mapped.
18         ;
19 021276 012704 000002      MOV      #2, R4          ; Get index for first device entry
20         ;
21         ; See if this device has a mapped handler
22         ;
23 021302 005764 000000G  1$:      TST      HANPAR(R4)      ; Is this handler mapped?
24 021306 001402          BEQ      2$              ; Br if not
25         ;
26         ; We found an entry for a device with a mapped handler.
27         ; Load the handler.
28         ;
29 021310 004737 021332'      CALL      LDHNHI          ; Load a mapped handler
30         ;
31         ; Look for more mapped handlers
32         ;
33 021314 062704 000002  2$:      ADD      #2, R4          ; Increment device index
34 021320 020437 000000G      CMP      R4, NUMDEV      ; Checked all devices?
35 021324 101766          BLOS     1$              ; Loop if not
36         ;
37         ; Finished
38         ;
39 021326 012604          MOV      (SP)+, R4
40 021330 000207          RETURN

```

```

1          .SBTTL LDHNHI -- Load device handler into extended memory
2          ;-----
3          ; LDHNHI is called to load a device handler into extended memory.
4          ;
5          ; Inputs:
6          ;   R4 = Device index number.
7          ;   R5 = 64-byte block number of top of free memory area.
8          ;
9          ; Outputs:
10         ;   R5 = 64-byte block number of new top of free memory area.
11         ;
12 021332 010146 LDHNHI: MOV     R1,-(SP)
13 021334 010246      MOV     R2,-(SP)
14 021336 010346      MOV     R3,-(SP)
15 021340 010446      MOV     R4,-(SP)
16         ;
17         ; Open channel 1 to the handler file.
18         ;
19 021342 016437 000000G 000246'      MOV     PNAME(R4),HANNAM+2 ;Set the device name for the lookup
20 021350 016437 000000G 000146'      MOV     PNAME(R4),CURNAM;Set name in case we have an error
21 021356      .LOOKUP #AREA,#1,#HANNAM;Try to open the handler file
22 021376 103004      BCC     B$          ;Br if we found the handler file
23 021400 012700 001636'      MOV     #CFHMSG,R0      ;Can't find handler
24 021404 000137 022124'      JMP     HLERR          ;Abort initialization
25         ;
26         ; Read block 0 of the handler file and extract some information
27         ;
28 021410 013702 000152'      B$:      MOV     WRKBUF,R2      ;Get address of work buffer
29 021414      .READW #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
30 021450 016237 000000G 000304'      MOV     H.GEN(R2),HGENFL;Save handler sysgen flags
31         ;
32         ; Set virtual address of handler entry point
33         ;
34 021456 012764 000006G 000000G      MOV     #VPAR5+6,HANENT(R4) ;Set virtual addr of handler entry point
35         ;
36         ; Get information about the size of the handler and determine the
37         ; address in extended memory where the handler is to be loaded.
38         ;
39 021464 016402 000000G      MOV     HANSIZ(R4),R2      ;Get size of handler (bytes)
40 021470 005202      INC     R2          ;Make sure handler size is even
41 021472 042702 000001      BIC     #1,R2
42 021476 010200      MOV     R2,R0
43 021500 062700 000077      ADD     #63.,R0      ;Round up to # 64-byte blocks
44 021504 072027 177772      ASH     #-6,R0      ;Get # 64-byte blocks for handler
45 021510 060037 000000G      ADD     R0,MHNSIZ      ;Accumulate total space for mapped handlers
46 021514 160005      SUB     R0,R5      ;Reserve room for handler
47 021516 010564 000000G      MOV     R5,HANPAR(R4)      ;Set mapping value for handler
48 021522 010537 000126'      MOV     R5,HMAP      ;Set initial PAR base for handler
49 021526 012737 000001 000142'      MOV     #1,FILBLK      ;Set # of block to read from file
50         ;
51         ; Begin loop to read handler into memory
52         ;
53 021534 010203      1$:      MOV     R2,R3      ;Get remaining size of handler
54 021536 020327 001000      CMP     R3,#512.      ;Compare with max we can read at one time
55 021542 101402      BLOS     2$          ;Br if we can read remainder of handler
56 021544 012703 001000      MOV     #512.,R3      ;Read one block
57 021550 160302      2$:      SUB     R3,R2      ;Reduce amt of handler left to read

```

```

58      ;
59      ; Read next block of handler
60      ;
61 021552 006203      ASR      R3      ;Get # words to read
62 021554 013701 000152'  MOV      WRKBUF,R1      ;Get address of buffer for read
63 021560      . READW  #AREA,#1,R1,R3,FILBLK ;Read a block
64 021614 103004      BCC      3$      ;Br if read ok
65 021616 012700 001700'  MOV      #ERHMSG,R0      ;Get error message
66 021622 000137 022124'  JMP      HLERR      ;Abort initialization
67      ;
68      ; Move the code we just read into the XM area for the handler
69      ;
70 021626 012700 000000G  3$:      MOV      #VPAR5,R0      ;Get virtual address of mapped region
71 021632      DISABL      ;** Disable interrupts **
72 021640 013746 000000G  MOV      @#KPAR5,-(SP) ;;;Save current mapping of PAR 5
73 021644 013737 000126' 000000G  MOV      HMAP,@#KPAR5 ;;;Set up mapping to get to XM area
74 021652 052737 000000G 000000G  BIS      #MMENBL,@#SROMMR ;;;Enable memory management
75 021660 105737 000000G  TSTB     MEM256 ;;;Does machine have > 256KB?
76 021664 001403      BEQ      4$      ;;;Br if not
77 021666 052737 000000G 000000G  BIS      #EMMAP,@#SR3MMR ;;;Enable extended memory addressing
78 021674 012120      4$:      MOV      (R1)+,(R0)+ ;;;Move from WRKBUF to XM region
79 021676 077302      SOB      R3,4$ ;;;Loop till all moved
80 021700 105737 000000G  TSTB     MEM256 ;;;Does machine have > 256KB?
81 021704 001403      BEQ      5$      ;;;Br if not
82 021706 042737 000000G 000000G  BIC      #EMMAP,@#SR3MMR ;;;Disable extended memory addressing
83 021714 042737 000000G 000000G  5$:      BIC      #MMENBL,@#SROMMR ;;;Enable memory management
84 021722 012637 000000G  MOV      (SP)+,@#KPAR5 ;;;Replace PAR 5 mapping
85 021726      ENABL      ; ** Enable interrupts **
86      ;
87      ; See if there is more to read
88      ;
89 021734 062737 000010 000126'  ADD      #8,HMAP ;Increase XM region base
90 021742 005237 000142'  INC      FILBLK ;Increment file block number
91 021746 005702      TST      R2 ;Is there more to read?
92 021750 001271      BNE      1$ ;Loop if more to read
93      ;
94      ; We have finished moving the handler into its XM region.
95      ; Set up addresses of system routines in a vector at the end of the handler
96      ;
97 021752 012703 000000G  MOV      #VPAR5,R3 ;Get virtual address of handler base
98 021756 066403 000000G  ADD      HANSIZ(R4),R3 ;Get virtual address beyond end of handler
99 021762 004737 022662'  CALL     HANMAP ;;;Map KPAR5 to the handler
100 021766 004737 022032'  CALL     STHNPV ;;;Set up handler pointer vector
101 021772 004737 022736'  CALL     HANUMP ;Restore mapping
102      ;
103      ; If handler has any load-time code, run it now
104      ;
105 021776 010537 000134'  MOV      R5,FMEMHI ;Set addr of top of free memory area
106 022002 004737 022162'  CALL     DOHNLC ;Run any load-time code for handler
107 022006 013705 000134'  MOV      FMEMHI,R5 ;Get new top of free memory address
108      ;
109      ; Close the handler file
110      ;
111 022012      .CLOSE  #1 ;Close the handler file
112      ;
113      ; Finished
114      ;

```

TSINIT -- TSX startup initializ MACRO V05.04 Monday 14-Dec-87 08:35 Page 56-2
LDHNHI -- Load device handler into extended memory

115	022020	012604	MOV	(SP)+, R4
116	022022	012603	MOV	(SP)+, R3
117	022024	012602	MOV	(SP)+, R2
118	022026	012601	MOV	(SP)+, R1
119	022030	000207	RETURN	

```

1          .SBTTL  STHNPV -- Initialize pointer vector in a handler
2          ;-----
3          ; STHNPV is called to initialize the pointer vector at the end of a
4          ; handler which provides the addresses of various system routines to the
5          ; handler.
6          ;
7          ; Inputs:
8          ;   R3 = Address beyond the end of the handler.
9          ;   HGENFL = Sysgen option flags for the handler being loaded.
10         ;
11 022032 010346 STHNPV: MOV      R3,-(SP)
12         ;
13         ; Set up addresses in the pointer vector
14         ;
15 022034 012743 000000G          MOV      #FORK,-(R3)      ; Address of fork routine
16 022040 012743 000000G          MOV      #INTEN,-(R3)     ; Address of inten routine
17 022044 032737 000000G 000304' BIT      #SG$IOT,HGENFL   ; Does handler want timeout support?
18 022052 001402          BEQ      2$                      ; Br if not
19 022054 012743 000000G          MOV      #IOTIMR,-(R3)    ; Set address of timeout support routine
20 022060 032737 000000G 000304' 2$: BIT      #SG$ELG,HGENFL ; Does handler want error logging support?
21 022066 001402          BEQ      3$                      ; Br if not
22 022070 012743 000000G          MOV      #ERRLOG,-(R3)    ; Set address of error logging routine
23 022074 012743 000000G          3$: MOV      #PTWRD,-(R3)
24 022100 012743 000000G          MOV      #PTBYT,-(R3)
25 022104 012743 000000G          MOV      #GTBYT,-(R3)
26 022110 012743 000000G          MOV      #MPPHY,-(R3)
27 022114 012743 000000G          MOV      #RELOC,-(R3)
28         ;
29         ; Finished
30         ;
31 022120 012603          MOV      (SP)+,R3
32 022122 000207          RETURN
  
```

STHNPV -- Initialize pointer vector in a handler

```

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

```

DOHNLC -- Execute and handler load/fetch code

```

1          .SBTTL DOHNLC -- Execute and handler load/fetch code
2          ;-----
3          ; If the handler being loaded has any Load-time execution code, read it
4          ; into our work buffer and execute it now.
5          ;
6          ; Inputs:
7          ;   R4 = Device index number of handler that is being loaded.
8          ;
9          ; Outputs:
10         ;   C-flag is set on return if load code signals an error during its
11         ;   execution.
12         ;
13 022162 010146 DOHNLC: MOV     R1,-(SP)
14 022164 010246      MOV     R2,-(SP)
15 022166 010346      MOV     R3,-(SP)
16 022170 010446      MOV     R4,-(SP)
17 022172 010546      MOV     R5,-(SP)
18         ;
19         ; Examine 1st word of handler to see if it could have any load-time code.
20         ;
21 022174 016405 0000006 MOV     HANENT(R4),R5 ;Get address of handler entry point
22 022200 004737 022662' CALL     HANMAP      ;;;Map Kpar5 to handler if mapped handler
23 022204 016500 0000004 MOV     4(R5),R0      ;;;Get 1st instruction located at 4 in handler
24 022210 004737 022736' CALL     HANUMP      ;Restore normal mapping
25 022214 020027 000240 CMP     R0,#240      ;Is it a NOP?
26 022220 103516 BLD     7$          ;Br if can't be any load code
27 022222 020027 000277 CMP     R0,#277      ;
28 022226 101113 BHI     7$          ;Br if can't be any load code
29 022230 132700 0000004 BITB    #4,R0      ;Is there load code?
30 022234 001510 BEQ     7$          ;Br if not
31         ;
32         ; Handler may have load code.
33         ; Read block 0 of handler and get offset to load code.
34         ;
35 022236 013702 000152' MOV     WRKBUF,R2      ;Get addr of our work buffer
36 022242      .READW    #AREA,#1,R2,#256.,#0 ;Read block 0 of handler
37 022276 022227 031066 CMP     (R2)+,#^RHAN ;Is this a new type handler?
38 022302 001065 BNE     7$          ;Br if not
39 022304 016203 0000004 MOV     4(R2),R3      ;Get offset to load code
40 022310 001462 BEQ     7$          ;Br if there is none
41         ;
42         ; There is load-time code.
43         ; Read into WRKBUF the portion of the handler with the load code.
44         ;
45 022312 020327 001000 CMP     R3,#1000      ;Is load code in block 0 of handler?
46 022316 103424 BLD     1$          ;Br if yes
47 022320 010302 MOV     R3,R2          ;Get offset to start of load code
48 022322 072227 177767 ASH     #-9.,R2      ;Convert to a block number
49 022326 042702 177400 BIC     #^C377,R2      ;Clear all but block number
50 022332      .READW    #AREA,#1,WRKBUF,#512.,R2 ;Read 2 blocks from handler file
51         ;
52         ; The load code is now in WRKBUF. Set up and execute it.
53         ;
54 022370 010437 000144' 1$: MOV     R4,CURDEV      ;Save current device index number
55 022374 042703 177000 BIC     #^C777,R3      ;Get offset within block of load code entry pt
56 022400 010300 MOV     R3,R0          ;Get entry point offset
57 022402 063700 000152' ADD     WRKBUF,R0      ;Add base address

```

```

58 022406 013701 000000G      MOV      RPRVEC,R1      ;Get pointer to GETVEC routine for Pro
59 022412 012702 000000C      MOV      #MAXDEV*2,R2    ;Get 2** entries in device tables
60 022416 012703 0000004      MOV      #4,R3          ;Set code saying this is load code
61 022422 012705 000000G      MOV      #HANENT,R5      ;Point to handler entry address vector
62 022426 060405              ADD      R4,R5          ;Point to entry cell for this handler
63 022430 004737 022662'      CALL     HANMAP        ;;;Map Kpar5 to handler if it is a mapped
64 022434 012704 022512'      MOV      #LDREAD,R4      ;;;Get pointer to Read routine
65 022440 004710              CALL     (R0)          ;;;Execute the load code
66 022442 103407              BCS      2$            ;;;Br if handler load code signaled an error
67                          ;
68                          ; Fetch/load code ran ok.
69                          ; Turn off handler mapping.
70                          ;
71 022444 012737 001400 000000G      MOV      #1400,@#KPAR6 ;;;Restore original mapping for RT-11
72 022452 004737 022736'      CALL     HANUMP        ;Unmap the handler
73 022456 000241              7$: CLC              ;Clear the carry flag for return
74 022460 000406              BR       9$
75                          ;
76                          ; Error occurred in fetch/load code
77                          ;
78 022462 012737 001400 000000G 2$: MOV      #1400,@#KPAR6 ;;;Restore original mapping for RT-11
79 022470 004737 022736'      CALL     HANUMP        ;Unmap the handler
80 022474 000261              SEC              ;Set the carry flag for return
81                          ;
82                          ; Finished
83                          ;
84 022476 012605              9$: MOV      (SP)+,R5
85 022500 012604              MOV      (SP)+,R4
86 022502 012603              MOV      (SP)+,R3
87 022504 012602              MOV      (SP)+,R2
88 022506 012601              MOV      (SP)+,R1
89 022510 000207              RETURN

```

```

1          .SBTTL LDREAD -- Perform I/O for handler load code
2          ;-----
3          ; This routine performs Read operations for handler load code.
4          ; It simulates the operation of the bootstrap read routine.
5          ; When called, Channel 1 must be open to the handler file.
6          ;
7          ; Inputs:
8          ;   R0 = Block number within handler file to be read.
9          ;   R1 = Number of words to read.
10         ;   R2 = Buffer address
11         ;
12         ; Outputs:
13         ;   C-flag is set if a read error occurs
14         ;
15 022512 010046 LDREAD: MOV     R0,-(SP)
16 022514 010446      MOV     R4,-(SP)      ;;;
17 022516 010004      MOV     R0,R4        ;;;Get starting block number
18         ;
19         ; Save current mapping information
20         ;
21 022520 105737 000000G      TSTB     MEM256      ;;;Does machine have > 256?
22 022524 001402      BEQ      1$           ;;;Br if not
23 022526 013746 000000G      MOV     @#SR3MMR,-(SP) ;;;Save extended memory address register
24 022532 013746 000000G 1$: MOV     @#SROMMR,-(SP) ;;;Save memory mapping
25 022536 013746 000000G      MOV     @#KPAR5,-(SP) ;;;Save current KPAR5 mapping
26 022542 013746 000000G      MOV     @#KPAR6,-(SP) ;;;Save current KPAR6 mapping
27 022546 012737 001400 000000G      MOV     #1400,@#KPAR6 ;;;Restore original mapping for RT-11
28         ;
29         ; Turn off handler mapping
30         ;
31 022554 004737 022736'      CALL     HANUMP      ;Turn off handler mapping
32         ;
33         ; Read the requested data from the handler
34         ;
35 022560      .READW  #AREA,#1,R2,R1,R4 ;Read the blocks
36 022612 103420      BCS      9$           ;Br if read error
37         ;
38         ; Restore handler mapping
39         ;
40 022614 013704 000144'      MOV     CURDEV,R4      ;Get current device index
41 022620 004737 022662'      CALL     HANMAP      ;;;Map Kpar5 if necessary
42         ;
43         ; Restore mapping information
44         ;
45 022624 012637 000000G      MOV     (SP)+,@#KPAR6 ;;;Restore KPAR6 mapping
46 022630 012637 000000G      MOV     (SP)+,@#KPAR5 ;;;Restore KPAR5 mapping
47 022634 012637 000000G      MOV     (SP)+,@#SROMMR ;;;Restore memory mapping
48 022640 105737 000000G      TSTB     MEM256      ;;;Does machine have > 256?
49 022644 001402      BEQ      2$           ;;;Br if not
50 022646 012637 000000G      MOV     (SP)+,@#SR3MMR ;;;Restore extended memory address register
51         ;
52         ; Finished
53         ;
54 022652 000241 2$: CLC                      ;;;Signal success on return
55 022654 012604 9$: MOV     (SP)+,R4
56 022656 012600      MOV     (SP)+,R0
57 022660 000207      RETURN

```

HANMAP -- Set up KPAR5 to access a mapped handler

```

1          .SBTTL  HANMAP -- Set up KPAR5 to access a mapped handler
2          -----
3          ; This routine is called to determine if a handler is mapped and if so
4          ; to turn on mapping and set up KPAR5 to access the mapped handler.
5          ; If the handler is not mapped, mapping is not turned on and KPAR5 is
6          ; not altered.
7          ; Interrupts are left disabled by this routine.
8          ; In addition to setting up mapping, this routine also changes the RMON
9          ; pointer to point to the TSX-Plus simulated RMON vector.
10         ;
11         ; Inputs:
12         ;   R4 = Device index number
13         ;
14 022662   HANMAP:
15         ;
16         ; Disable interrupts
17         ;
18 022662   DISABL          ;;;Disable interrupts
19         ;
20         ; Change RMON pointer to point to TSX-Plus vector
21         ;
22 022670   012737  000000G 000000G      MOV      #MONVEC, @RMON ;;;Say TSX-Plus is the monitor
23         ;
24         ; See if this handler is mapped
25         ;
26 022676   005764  000000G      TST      HANPAR(R4)      ;;;Is this handler mapped?
27 022702   001403          BEQ      9$          ;;;Br if not
28         ;
29         ; This handler is mapped.
30         ; Set up mapping to access it.
31         ;
32 022704   016437  000000G 000000G      MOV      HANPAR(R4), @KPAR5; ;Map KPAR5 to the handler code
33 022712   052737  000000G 000000G 9$:   BIS      #MMENBL, @SR0MMR; ;Enable memory mapping
34 022720   105737  000000G      TSTB     MEM256          ;;;Does machine have > 256KB?
35 022724   001403          BEQ      10$          ;;;Br if not
36 022726   052737  000000G 000000G      BIS      #EMMAP, @SR3MMR ;;;Enable extended memory addressing
37         ;
38         ; Finished
39         ;
40 022734   000207      10$:   RETURN
41         ;
42         .SBTTL  HANUMP -- Turn off memory mapping to a handler
43         -----
44         ; This routine is the companion to HANMAP. It turns off memory mapping
45         ; and restores KPAR5 to its normal mapping value.
46         ; Enter with interrupts disabled. Interrupts are enabled on return.
47         ; This routine also changes the RMON pointer back to RT-11.
48         ;
49 022736   HANUMP:
50         ;
51         ; Turn off memory management
52         ;
53 022736   105737  000000G      TSTB     MEM256          ;;;Does machine have > 256KB?
54 022742   001403          BEQ      1$          ;;;Br if not
55 022744   042737  000000G 000000G      BIC      #EMMAP, @SR3MMR ;;;Turn off extended memory addressing
56 022752   042737  000000G 000000G 1$:   BIC      #MMENBL, @SR0MMR; ;Turn off memory mapping
57 022760   012737  001200  000000G      MOV      #1200, @KPAR5 ;;;Reset KPAR5 to its normal mapping

```

HANUMP -- Turn off memory mapping to a handler

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

```

1          .SBTTL  FNDHRB -- Try to find a handler global region
2          ;-----
3          ; This routine is called to try to locate an allocated XM region with
4          ; a specified name.
5          ; If a region control block with the specified name cannot be found,
6          ; the address of a free one is returned and the specified name is stored
7          ; into the free block.
8          ;
9          ; Inputs:
10         ; R5 = Pointer to 2-word cell containing Rad50 name of region to be found.
11         ;
12         ; Outputs:
13         ; C-flag cleared ==> Found the specified RCB.
14         ; R1 = Address of the RCB
15         ; C-flag set ==> Could not find the specified RCB.
16         ; R1 = Pointer to a free RCB or 0 if no available RCB's.
17         ;
18 023004 010246 FNDHRB: MOV     R2,-(SP)
19         ;
20         ; Search for specified RCB and also remember if we see a free RCB
21         ;
22 023006 005002          CLR     R2          ; Say no free RCB found
23 023010 013701 00000000 MOV     HANRCB,R1      ; Point to start of RCB area
24 023014 005721          TST     (R1)+      ; Skip over -1 word at front
25 023016 005711 1$:      TST     (R1)       ; What is the status of this RCB
26 023020 001002          BNE     2$        ; Br if this is not a free RCB
27 023022 010102          MOV     R1,R2      ; Remember address of a free RCB
28 023024 000412          BR      3$
29 023026 021127 177777 2$:      CMP     (R1), #-1      ; Are we at the end of the list?
30 023032 001412          BEQ     4$        ; Br if yes
31 023034 021561 000006          CMP     (R5), 6(R1)     ; Compare the names
32 023040 001004          BNE     3$        ; Br if don't match
33 023042 026561 000002 000010 CMP     2(R5), 10(R1)    ; Compare 2nd half of name
34 023050 001417          BEQ     6$        ; Br if found the RCB we were searching for
35 023052 062701 000012 3$:      ADD     #12,R1      ; Point to the next RCB
36 023056 000757          BR      1$        ; Continue searching
37         ;
38         ; We could not find the specified RCB.
39         ; If there was a free one, initialize the name.
40         ;
41 023060 010201 4$:      MOV     R2,R1      ; Was there a free RCB?
42 023062 001410          BEQ     5$        ; Br if not
43 023064 011561 000006          MOV     (R5), 6(R1)     ; Set name in the RCB
44 023070 016561 000002 000010 MOV     2(R5), 10(R1)    ; (2nd word of name)
45 023076 063761 000144' 000010 ADD     CURDEV, 10(R1)   ; Make name unique to device
46 023104 000261 5$:      SEC             ; Signal that we did not find the RCB
47 023106 000401          BR      9$
48         ;
49         ; We found the RCB
50         ;
51 023110 000241 6$:      CLC             ; Signal success on return
52         ;
53         ; Finished
54         ;
55 023112 012602 9$:      MOV     (SP)+, R2
56 023114 000207          RETURN

```

```

1          .SBTTL  HANXMR -- Allocate XM region during handler load
2          -----
3          ; This routine can be called by a handler as its is being loaded to
4          ; allocate an XM region for the handler.
5          ;
6          ; Inputs:
7          ;   R2 = Number of 64-byte units needed for XM region
8          ;
9          ; Outputs:
10         ;   C-flag cleared ==> Successfully allocated a region
11         ;   R1 = 64-byte address of base of allocated region
12         ;   R2 = Requested size
13         ;   C-flag set ==> Could not allocate the region
14         ;   R2 = Largest available region size
15         ;
16         ; Notes: FMEMLO and FMEMHI are used by this routine to indicate the
17         ; bottom and top of the free memory area that can be allocated.
18         ; The allocation is done from the top of free memory downward.
19         ; FMEMHI is updated to have the new top of free memory after the
20         ; allocation has been done.
21         ;
22 023116 010046 HANXMR: MOV      R0,-(SP)
23         ;
24         ; Get the total amount of free memory space available now
25         ; and see if the requested region can be allocated.
26         ;
27 023120 013700 000134'      MOV      FMEMHI,R0      ;Top of free memory
28 023124 163700 000136'      SUB      FMEMLO,R0      ;-Base of free memory
29 023130 020200              CMP      R2,R0           ;Do we have room for the requested region?
30 023132 101006              BHI      B$              ;Br if not
31         ;
32         ; There is room for the region so allocate it from the top of memory
33         ;
34 023134 160237 000134'      SUB      R2,FMEMHI      ;Allocate the region
35 023140 013701 000134'      MOV      FMEMHI,R1      ;Return the address of the base of the region
36 023144 000241              CLC                      ;Signal success on return
37 023146 000402              BR       9$
38         ;
39         ; We are not able to allocate the region.
40         ; Return with C-flag set and the size of the largest possible region in R2.
41         ;
42 023150 010002 8$:          MOV      R0,R2           ;Get the size of the largest possible region
43 023152 000261              SEC                      ;Signal failure on return
44         ;
45         ; Finished
46         ;
47 023154 012600 9$:          MOV      (SP)+,R0
48 023156 000207              RETURN

```

```

1          .IF      NE,<PROASM-1>      ; If assembling for PDP-11
2          .SBTTL   SETMIO -- Set up information about mapped devices
3          -----
4          ; SETMIO is called to set up information about which devices have to have
5          ; their I/O mapped through system buffers. I/O mapping is done for DMA
6          ; devices with 18-bit controllers being used on Q-bus systems with more
7          ; than 256Kb of memory.
8          ; The DX$MAP flag is set in the DVFLAG word for devices that require mapping.
9          ;
10         ; Inputs:
11         ;   R5 = Pointer to low-memory free area
12         ;
13         ; Outputs:
14         ;   MIOFLAG = 0==>I/O mapping not required for any device;
15         ;               1==>I/O mapping required for some device.
16         ;   R5 = Pointer to new low-memory free area.
17         ;
18 023160 010146 SETMIO: MOV      R1,-(SP)
19
20         ; Determine if this machine requires mapping at all
21         ;
22 023162 005727 000000G          TST      #MIODBG          ; Are we debugging mapped I/O system?
23 023166 001017          BNE      2$          ; Br if yes
24 023170 032737 000001 000306'  BIT      #EXTLSI,ICONFG    ; Is this a Q-bus machine with more than 256Kb?
25 023176 001013          BNE      2$          ; Br if yes
26
27         ; This is not a Q-bus system with more than 256Kb.
28         ; Mapping is not required at all.
29         ;
30 023200 012701 000002          MOV      #2,R1          ; Get initial device index number
31 023204 042761 000000G 000000G 1$: BIC      #DX$MAP,DVFLAG(R1) ; Clear mapped-I/O flag
32 023212 062701 000002          ADD      #2,R1          ; Get next device index
33 023216 020137 000000G          CMP      R1,NUMDEV       ; More to do?
34 023222 101770          BLOS      1$          ; Br if yes
35 023224 000464          BR        9$
36
37         ; This is a Q-bus system with more than 256Kb.
38         ; See if any devices have requested mapped I/O.
39         ;
40 023226 005000 2$: CLR      R0          ; Clear composite flag word
41 023230 012701 000002          MOV      #2,R1          ; Initialize device index number
42 023234 056100 000000G 3$: BIS      DVFLAG(R1),R0       ; Combine flags from all devices
43 023240 062701 000002          ADD      #2,R1          ; Get next device index number
44 023244 020137 000000G          CMP      R1,NUMDEV       ; Checked all devices?
45 023250 101771          BLOS      3$          ; Br if not
46 023252 032700 000000G          BIT      #DX$MAP,R0       ; Does any device require mapping?
47 023256 001447          BEQ      9$          ; Br if not
48
49         ; I/O mapping is required
50         ;
51 023260 105237 000000G          INCB     MIOFLG          ; Remember that mapping is required
52
53         ; Zero the area where we will build the control structures
54         ;
55 023264 113701 000000G          MOVB     VMIOBF,R1        ; Get # buffers wanted
56 023270 070127 000000G          MUL      #MI$$SZ,R1        ; Times size for each control block
57 023274 062701 000000C          ADD      #MIONWB*MW$$SZ,R1 ; Add space for MIO wait blocks

```

Line	Address	Hex	Dec	Op	Opnd	Comment
58	023300	006201		ASR	R1	;Get # words to zero
59	023302	010500		MOV	R5,R0	;Get pointer to start of area
60	023304	005020		4\$: CLR	(R0)+	;Zero the entire area
61	023306	077102		SOB	R1,4\$	
62						
63						; Allocate and initialize the control structures
64						
65	023310	010537	000000G	MOV	R5,MIOBHD	;Start of area for I/O mapping control blks
66	023314	113700	000000G	MOVB	VMIOBF,R0	;Get # buffers wanted
67	023320	010501		5\$: MOV	R5,R1	;Get pointer to current control block
68	023322	062701	000000G	ADD	#MI\$\$SZ,R1	;Get pointer to next control block
69	023326	005300		DEC	R0	;Need to link more together?
70	023330	003404		BLE	6\$	;Br if not
71	023332	010165	000000G	MOV	R1,MI\$LNK(R5)	;Set pointer to next control block
72	023336	010105		MOV	R1,R5	;Advance pointer to next block
73	023340	000767		BR	5\$	;See if more to do
74	023342	010105		6\$: MOV	R1,R5	;Set pointer past last block
75	023344	010537	000000G	MOV	R5,MIOWHD	;Start of wait blocks
76	023350	012700	177777G	MOV	#MIONWB-1,R0	;Get # wait blocks wanted - 1
77	023354	010501		7\$: MOV	R5,R1	;Get pointer to current block
78	023356	062701	000000G	ADD	#MW\$\$SZ,R1	;Get pointer to next block
79	023362	010165	000000G	MOV	R1,MW\$LNK(R5)	;Set pointer to next wait block
80	023366	010105		MOV	R1,R5	;Advance pointer to next block
81	023370	077007		SOB	R0,7\$	;Loop if more to allocate
82	023372	062705	000000G	ADD	#MW\$\$SZ,R5	;Allocate space for last block (with 0 link)
83						
84						; Finished
85						
86	023376	012601		9\$: MOV	(SP)+,R1	
87	023400	000207		RETURN		
88				. IFF	;NE,<PROASM-1>	
89						
90						; Define dummy routine for Pro
91						
92				SETMIO:	RETURN	
93				. ENDC	;NE,<PROASM-1>	

```

1          .SBTTL  OVLPOS -- Determine which overlays go over TSINIT
2          ;-----
3          ; OVLPOS is called to determine which system overlays are to be placed
4          ; over the TSINIT code (specifically, between @OVLBAS and INITOP).
5          ;
6          ; Inputs:
7          ;   R5 = Base address in TSINIT where overlays may be loaded.
8          ;
9          ; Outputs:
10         ;   Overlay segment information is set up in OSTABL.
11         ;   OS$FLG(seg) = 0==>Load seg into high memory; 1==>Load over TSINIT.
12         ;   OVLBAS = Address of location within TSINIT where overlays start.
13         ;   R5 = Pointer past last overlay loaded over TSINIT.
14         ;
15 023402 010146 OVLPOS: MOV     R1,-(SP)
16 023404 010246      MOV     R2,-(SP)
17 023406 010346      MOV     R3,-(SP)
18 023410 010446      MOV     R4,-(SP)
19 023412 010504      MOV     R5,R4          ;Get address where we may load overlays
20
21         ; Build the table that holds information about the overlays
22         ;
23 023414 004737 023612'      CALL    OVLBLD          ;Build overlay information table
24
25         ; First determine how much space will be used by those overlays that are
26         ; forced to be loaded over TSINIT.
27         ;
28 023420 062704 000077      ADD     #63.,R4          ;Bound address to 64-byte boundary
29 023424 042704 000077      BIC     #77,R4
30 023430 010437 000140'      MOV     R4,OVLBAS          ;Remember address where we load overlays
31 023434 012702 000576'      MOV     #OSTABL,R2        ;Point to start of table
32 023440 005762 000000      1$:    TST     OS$SIZ(R2)    ;Is this overlay to be loaded?
33 023444 001423              BEQ     2$                ;Br if not
34 023446 016200 000004      MOV     OS$OVL(R2),R0      ;Point to linker-built entry
35 023452 016000 000000G      MOV     O.ADR(R0),R0      ;Get Rad50 segment ID
36 023456 012701 001026'      MOV     #LOWOVL,R1       ;Point to table of overlays to go over TSINIT
37 023462 020021      6$:    CMP     R0,(R1)+          ;Must this overlay go over TSINIT?
38 023464 001404              BEQ     4$                ;Br if yes
39 023466 020127 001034'      CMP     R1,#LOWEND        ;End of low-overlay table?
40 023472 103773              BLO     6$                ;Br if not
41 023474 000407              BR      2$                ;This overlay is not forced over TSINIT
42 023476 005262 000002      4$:    INC     OS$FLG(R2)    ;Set flag saying load over TSINIT
43 023502 016200 000000      MOV     OS$SIZ(R2),R0      ;Get # 64-byte blocks needed for overlay
44 023506 072027 000006      ASH     #6,R0              ;Get # bytes needed for overlay
45 023512 060004              ADD     R0,R4              ;Advance address within TSINIT
46 023514 062702 000006      2$:    ADD     #OS$$SZ,R2    ;Point to entry for next segment
47 023520 020237 001024'      CMP     R2,OSLAST        ;Have we finished?
48 023524 103745              BLO     1$                ;Loop if not
49
50         ; Determine how much memory space is available in TSINIT for other overlays
51         ;
52 023526 020427 030004'      CMP     R4,#INITOP-50.    ;Any space left for other overlays?
53 023532 103021              BHIS    9$                ;Br if not
54 023534 012705 030066'      MOV     #INITOP,R5        ;Point to top of overlay area
55 023540 160405              SUB     R4,R5              ;Total space available for overlays
56 023542 072527 177772      ASH     #-6,R5            ;Convert to # 64-byte blocks
57

```

```
58      ; Now begin loop which determines which other overlays go over TSINIT.
59      ; We do this in the order of largest to smallest to try to fill
60      ; the overlay area as completely as possible.
61      ;
62 023546 004737 025004' 3$: CALL OVLTRY      ; Try to find largest overlay that will fit
63 023552 103411          BCS 9$           ; Br if no more overlays will fit
64 023554 005262 000002   INC OS$FLG(R2)  ; Remember to load over TSINIT
65 023560 016200 000000   MOV OS$SIZ(R2),R0 ; Get # 64-byte blocks needed for overlay
66 023564 160005          SUB R0,R5       ; Reduce remaining free space in TSINIT
67 023566 072027 000006   ASH #6,R0       ; Get # bytes needed for overlay
68 023572 060004          ADD R0,R4       ; Advance overlay address in TSINIT
69 023574 000764          BR 3$           ; See if we can find more segments to load
70      ;
71      ; Finished
72      ;
73 023576 010405 9$: MOV R4,R5           ; Return top-of-overlay address in R5
74 023600 012604          MOV (SP)+,R4
75 023602 012603          MOV (SP)+,R3
76 023604 012602          MOV (SP)+,R2
77 023606 012601          MOV (SP)+,R1
78 023610 000207          RETURN
```

OVLBLD -- Build overlay information table

```

1          .SBTTL  OVLBLD -- Build overlay information table
2          ;-----
3          ; OVLBLD is called to build an overlay information table that is used
4          ; by TSINIT while loading TSX overlays into memory.
5          ;
6          ; Outputs:
7          ; Overlay segment information is set up in OSTABL.
8          ; OSLAST = Pointer past last entry in OSTABL.
9          ;
10         OVLBLD: MOV     R1,-(SP)
11         MOV     R2,-(SP)
12         MOV     R3,-(SP)
13         ;
14         ; Read 1st block of SAV file to get pointer to overlay table
15         ;
16         MOV     WRKBUF,R2          ;Point to work buffer
17         .READW  #AREA,#17,R2,#256.,#0 ;read first block of the save file
18         BCS     22$                ;Br if error on read
19         MOV     64(R2),R1          ;point to the overlay table
20         BNE     15$                ;br if overlays exist
21         ;
22         ; Must be version 3B overlays structure at absolute location.
23         ;
24         MOV     #137,@#1000        ;position jump instruction over 3b ovly handler
25         MOV     #$OVRH,@#1002      ;position overlay intercept location
26         MOV     #1104,R1           ;point to the overlay table
27         MOV     R1,OVRADD           ;save the address of the overlay table
28         ;
29         ; Initialize the table that holds information about the overlays
30         ;
31         15$: MOV     #OSTABL,R3      ;Point to table for overlay info
32         11$: MOV     R1,OS$OVL(R3)  ;Save pointer to overlay control block
33         CLR     OS$FLG(R3)          ;Assume seg will be loaded in high memory
34         CALL    ALCOVL              ;Determine if we should load this overlay
35         MOV     R2,OS$SIZ(R3)       ;Remember total size of overlay+data
36         ADD     #OS$$SZ,R3          ;Point to next overlay table entry
37         12$: ADD     #6,R1           ;find the next region
38         CMP     (R1),#4537          ;compare with a <JSR R5,$OVRH> instruction
39         BNE     11$                ;Br if not at end
40         MOV     R3,OSLAST           ;Save pointer past last overlay table entry
41         ;
42         ; Finished
43         ;
44         MOV     (SP)+,R3
45         MOV     (SP)+,R2
46         MOV     (SP)+,R1
47         RETURN
48         ;
49         ; Error -- Read error occured while reading overlay table
50         ;
51         22$: .PRINT  #TSXHD
52         .PRINT  #RDERR
53         JMP     INISTP

```

GETMAP -- Load any mapped system code regions

```

1          .SBTTL  GETMAP -- Load any mapped system code regions
2          ;-----
3          ; GETMAP is called to load those system overlays that are placed
4          ; in high memory.
5          ;
6          ; Inputs:
7          ;   R5 = 64-byte block number of top of free memory.
8          ;
9          ; Outputs:
10         ;   R5 = New 64-byte block number of top of free memory.
11         ;
12 024012 010146 GETMAP: MOV     R1,-(SP)
13 024014 010246      MOV     R2,-(SP)
14 024016 010346      MOV     R3,-(SP)
15 024020 010537 000000G      MOV     R5,SMRSIZ      ; Save memory pointer at start of allocation
16
17         ; Now that most of the system initialization is completed, we must check
18         ; again to see which overlays need to be loaded.
19         ;
20 024024 012703 000576'      MOV     #OSTABL,R3      ; Point to 1st overlay table entry
21 024030 004737 024276' 1$:  CALL     OPTOVL      ; See if this segment should be loaded
22 024034 010263 000000      MOV     R2,OS$SIZ(R3)    ; Save # 64-byte blocks needed for overlay
23 024040 062703 000006      ADD     #OS$$SZ,R3      ; Point to next overlay table entry
24 024044 020337 001024'      CMP     R3,OSLAST     ; Checked all entries in overlay table?
25 024050 103767      BLO     1$      ; Br if not
26
27         ; Load those overlays that go into high memory
28         ;
29 024052 012702 000576'      MOV     #OSTABL,R2      ; Point to 1st overlay entry
30 024056 005762 000000      3$:  TST     OS$SIZ(R2)    ; Is this overlay segment wanted?
31 024062 001405      BEQ     4$      ; Br if not
32 024064 005762 000002      TST     OS$FLG(R2)      ; Load over TSINIT or into high memory?
33 024070 001002      BNE     4$      ; Br if load over TSINIT
34 024072 004737 025102'      CALL     GETOVL      ; Load overlay into high memory
35 024076 062702 000006      4$:  ADD     #OS$$SZ,R2    ; Point to next overlay table entry
36 024102 020237 001024'      CMP     R2,OSLAST     ; Have we done all overlays?
37 024106 103763      BLO     3$      ; Loop if not
38
39         ; Finished
40         ;
41 024110 013700 000000G      19$:  MOV     SMRSIZ,R0    ; Get memory pointer at start of allocation
42 024114 160500      SUB     R5,R0      ; Calc amt of space allocated
43 024116 010037 000000G      MOV     R0,SMRSIZ      ; Save total space used for mapped regions
44 024122 012603      MOV     (SP)+,R3
45 024124 012602      MOV     (SP)+,R2
46 024126 012601      MOV     (SP)+,R1
47 024130 000207      RETURN

```

```

1          .SBTTL  ALCOVL -- Allocate space for a system overlay region
2          ;-----
3          ; ALCOVL is called to determine if a system overlay region is wanted
4          ; (based on sysgen options), and if it is wanted to determine how
5          ; much space is needed for the code and data.
6          ;
7          ; Inputs:
8          ;   R3 = Pointer to overlay table entry (OS$xxx)
9          ;
10         ; Outputs:
11         ;   C-flag cleared ==> This segment is to be loaded.
12         ;   C-flag set    ==> Do not load this overlay segment.
13         ;   R2 = # 64-Byte blocks needed for segment including data areas within it.
14         ;
15 024132 010146 ALCOVL: MOV      R1,-(SP)
16         ;
17         ; Get pointer to linker-built overlay entry for segment
18         ;
19 024134 016301 000004      MOV      OS$OVL(R3),R1  ;Get pointer to linker-built entry for seg
20         ;
21         ; Read in the first block of the overlay segment
22         ;
23 024140 013702 000152'      MOV      WRKBUF,R2      ;Point to work buffer
24 024144      .READW  #AREA,#17,R2,#256.,0,BLK(R1) ;read the first block
25 024202 103415      BCS      3$                    ;Br if read error
26         ;
27         ; Save the 3 character Rad50 segment ID in the 0.ADR cell of the
28         ; linker-built overlay table entry for this segment.
29         ;
30 024204 016261 000002 000000G      MOV      2(R2),0.ADR(R1) ;save the rad50 overlay identifier
31         ;
32         ; Make sure the segment is not larger than 8Kb
33         ;
34 024212 016102 000000G      MOV      0.SIZ(R1),R2      ;get the word count of the code region
35 024216 006302      ASL      R2                        ;convert to byte count
36 024220 020227 020000      CMP      R2,#20000          ;check for 8kb overflow
37 024224 101014      BHI      21$                    ;Br if region is too big
38         ;
39         ; Don't load some optional segments if features were not selected
40         ; in TSGEN.
41         ;
42 024226 004737 024276'      CALL      OPTOVL          ;See if we want to load this segment
43         ;
44         ; Finished
45         ; The C-flag is set or reset by OPTOVL.
46         ;
47 024232 012601      MOV      (SP)+,R1
48 024234 000207      RETURN
49         ;
50         ; Error -- Error on reading from SAV file
51         ;
52 024236      3$:      .PRINT  #TSXHD                    ;Print heading
53 024244      .PRINT  #RDERR                    ;Read error
54 024252 000137 004216'      JMP      INISTP          ;Abort initialization
55         ;
56         ; Error -- Insufficient memory space to load run-time systems
57         ;

```

TSINIT -- TSX startup initializ MACRO V05.04 Monday 14-Dec-87 08:35 Page 68-1
ALCOVL -- Allocate space for a system overlay region

58 024256	21#:	. PRINT	#TSXHD	; PRINT HEADING
59 024264		. PRINT	#TSXSIZ	; PRINT ERROR MESSAGE
60 024272 000137 004216'		JMP	INISTP	; ABORT INITIALIZATION

```

1          .SBTTL  OPTOVL -- Check for optional system overlay regions
2          ;-----
3          ; OPTOVL is called to determine if a specific system overlay is or is
4          ; not to be loaded based on sysgen options.
5          ; This routine may also add space for buffers to the overlay regions size.
6          ;
7          ; Inputs:
8          ;   R3 = Pointer to overlay table entry for segment (OS$xxx)
9          ;
10         ; Outputs:
11         ;   C-flag cleared ==> Load this overlay.
12         ;   C-flag set    ==> Do not load this overlay.
13         ;   R2 = # 64-byte blocks needed for code + data for the segment.
14         ;
15 024276 010346 OPTOVL: MOV     R3, -(SP)
16 024300 010446      MOV     R4, -(SP)
17 024302 010546      MOV     R5, -(SP)
18
19         ; Get the name of the overlay segment
20
21 024304 016305 000004      MOV     OS$OVL(R3), R5 ; Get pointer to linker-built entry
22 024310 016504 000000G     MOV     O.ADR(R5), R4 ; Get name of the segment
23
24         ; Get size of code portion of overlay segment
25
26 024314 016502 000000G     MOV     O.SIZ(R5), R2 ; Get # words needed by code portion of seg
27 024320 006302              ASL     R2 ; Convert to # bytes
28
29         ; See if this is an optional segment that we need to deal with specially
30
31 024322 012700 024346'     MOV     #OVLST, R0 ; Point to overlay name list
32 024326 020420 1$: CMP     R4, (R0)+ ; Found name of overlay?
33 024330 001405             BEQ     2$ ; Br if yes
34 024332 005720             TST     (R0)+ ; No -- Skip over address word
35 024334 020027 024432'     CMP     R0, #OVLEND ; Checked all names in the list?
36 024340 103772             BLO     1$ ; Loop if not
37 024342 000577             BR      OXYYES ; Load this overlay
38
39         ; Branch off to processing routine
40
41 024344 000130 2$: JMP     @(R0)+ ; Enter processing routine for the overlay
42
43         ; Table of overlay names and processing routines
44
45         .MACRO  OVLTLBL  NAME
46         .RAD50  /'NAME'/
47         .WORD   OOR'NAME
48         .ENDM   OVLTLBL
49
50 OVLST:
51 024346      OVLTLBL  USR ; TSUSR -- File management
52 024352      OVLTLBL  SPL ; TSSPOL -- Spooling system
53 024356      OVLTLBL  LOK ; TSLOCK -- Shared file record locking
54 024362      OVLTLBL  MSG ; TSMMSG -- Inter-job message communication
55 024366      OVLTLBL  SWP ; TSSWAP -- Job swapper
56 024372      OVLTLBL  PLS ; TSPLAS -- PLAS support
57 024376      OVLTLBL  SLE ; TSSLE -- Single line editor

```

58 024402		OVLTLBL	WIN	; TSWIN -- Display window management
59 024406		OVLTLBL	MIO	; TSMIO -- Mapped I/O
60 024412		OVLTLBL	CLO	; TSCLO -- CL handler
61 024416		OVLTLBL	DBG	; TSDBG -- Program debugger
62 024422		OVLTLBL	CSH	; TSCASH -- Data caching
63 024426		OVLTLBL	DMP	; TSDUMP -- Crash dump generator
64 024432		OVLEND:		
65				
66				
67				
68 024432	013703	0000000	OORUSR: MOV VNFCSH, R3	; Get # file cache entries
69 024436	070327	0000000	MUL #FC##SZ, R3	; Multiply by size of each entry
70 024442	060302		ADD R3, R2	; Allocate space for directory cache
71 024444	000536		BR OOXYES	; Load the segment
72				
73				
74				
75 024446	005727	0000000	OORSPL: TST #SPLND	; Are there any spooled devices?
76 024452	001530		BEQ OOXNO	; Br if not
77 024454	062702	0000000	ADD #<SPLNB*512.>, R2	; Reserve room for spool buffers
78 024460	013703	0000000	MOV NSPLBL, R3	; Get # blocks for spool file
79 024464	062703	0000007	ADD #7, R3	; Bound up to byte boundary
80 024470	072327	177775	ASH #-3, R3	; Divide by 8 to get # bytes for table
81 024474	005203		INC R3	; Round up to word boundary
82 024476	042703	0000001	BIC #1, R3	
83 024502	060302		ADD R3, R2	; Add space for spool file allocation table
84 024504	000516		BR OOXYES	; Load the segment
85				
86				
87				
88 024506	005737	0000000	OORLOK: TST VMXSF	; Any shared files?
89 024512	001510		BEQ OOXNO	; Br if not
90 024514	005737	0000000	TST VNUMDC	; Shared file data caching wanted?
91 024520	001110		BNE OOXYES	; Br if yes
92 024522	162702	0000000	SUB #DCCSIZ, R2	; Reduce size of segment - Leave out cache code
93 024526	000505		BR OOXYES	; Load the segment
94				
95				
96				
97 024530	013703	0000000	OORMSG: MOV VMAXMC, R3	; Is message communication facility wanted?
98 024534	001477		BEQ OOXNO	; Br if not
99 024536	070327	0000000	MUL #MB##SZ, R3	; Space for message channel blocks
100 024542	060302		ADD R3, R2	
101 024544	013703	0000000	MOV VMXMRB, R3	; Number of message request blocks
102 024550	070327	0000000	MUL #MR##SZ, R3	; Times size of request block
103 024554	060302		ADD R3, R2	
104 024556	013703	0000000	MOV VMSCHR, R3	; Max # chars in a message
105 024562	005203		INC R3	; Bound up to word
106 024564	042703	0000001	BIC #1, R3	; Reserve whole number of words
107 024570	062703	0000000	ADD #MU#TXT, R3	; Plus space for message header
108 024574	070337	0000000	MUL VMXMSG, R3	; Times maximum number of messages
109 024600	060302		ADD R3, R2	; Space for message buffers
110 024602	000457		BR OOXYES	
111				
112				
113				
114 024604	013703	0000000	OORPLS: MOV VPLAS, R3	; PLAS support wanted?

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

```
172                                     ;
173 024734 005002      DOXNO:  CLR      R2          ; Say no space needed for overlay
174 024736 000261          SEC          ; Signal don't load the segment
175 024740 000415          BR      DOXFIN
176                                     ;
177                                     ; Load this segment
178                                     ;
179 024742 005202      DOXYES: INC      R2          ; Make sure size is even
180 024744 042702 000001      BIC      #1,R2
181 024750 020227 020000      CMP      R2,#8192.    ; Don't allow code + data to exceed 8Kb
182 024754 101402          BLOS      1$          ; Br if ok
183 024756 012702 020000      MOV      #8192.,R2    ; Note, init code in segment will truncate dat
184 024762 062702 000077      1$:  ADD      #63.,R2  ; Convert to # 64-byte blocks
185 024766 072227 177772          ASH      #-6,R2
186 024772 000241          CLC          ; Signal to load the segment
187                                     ;
188                                     ; Finished
189                                     ;
190 024774 012605      DOXFIN: MOV      (SP)+,R5
191 024776 012604          MOV      (SP)+,R4
192 025000 012603          MOV      (SP)+,R3
193 025002 000207          RETURN
```

OVLTRY -- Find an overlay to place over TSINIT

```

1          .SBTTL  OVLTRY -- Find an overlay to place over TSINIT
2          ;-----
3          ; OVLTRY is called to identify the largest overlay segment which
4          ; will fit in the TSINIT area and which is not already marked to go
5          ; over TSINIT.
6          ;
7          ; Inputs:
8          ;   R5 = # 64-byte blocks available for segment in TSINIT.
9          ;
10         ; Outputs:
11         ;   R2 = Pointer to OSTABL entry for segment
12         ;   C-flag set ==> No more segments will fit.
13         ;
14 025004 010346 OVLTRY: MOV      R3,-(SP)
15         ;
16         ; Begin loop to examine all segments
17         ;
18 025006 005002          CLR      R2          ; Say we haven't found any segment yet
19 025010 012703 000576'  MOV      #OSTABL,R3    ; Point to entry for 1st segment
20 025014 005763 000000  1$: TST      OS$SIZ(R3)  ; Is this segment to be loaded?
21 025020 001415          BEQ      2$          ; Br if not
22 025022 005763 000002  TST      OS$FLG(R3)    ; Is this segment already over TSINIT?
23 025026 001012          BNE      2$          ; Br if yes
24 025030 026305 000000  CMP      OS$SIZ(R3),R5 ; Will this segment fit?
25 025034 101007          BHI      2$          ; Br if not
26 025036 005702          TST      R2          ; Have we found any other seg yet?
27 025040 001404          BEQ      3$          ; Br if not
28 025042 026362 000000 000000  CMP      OS$SIZ(R3),OS$SIZ(R2) ; Is new seg larger than old?
29 025050 101401          BLOS     2$          ; Br if not
30 025052 010302 3$: MOV      R3,R2          ; Remember largest segment
31 025054 062703 000006  2$: ADD      #OS$SZ,R3    ; Point to entry for next segment
32 025060 020337 001024'  CMP      R3,OSLAST    ; Have we checked all segments?
33 025064 103753          BLO      1$          ; Loop if not
34         ;
35         ; Finished
36         ;
37 025066 000241          CLC          ; Assume we found a segment
38 025070 005702          TST      R2          ; Did we find a segment that will fit?
39 025072 001001          BNE      9$          ; Br if yes
40 025074 000261          SEC          ; Signal failure on return
41 025076 012603 9$: MOV      (SP)+,R3
42 025100 000207          RETURN

```

GETOVL -- Load system overlay into high memory

```

1          .SBTTL  GETOVL -- Load system overlay into high memory
2          ;-----
3          ; GETOVL is called to load a system overlay into high memory.
4          ;
5          ; Inputs:
6          ;   R2 = Pointer to overlay table entry for segment in OSTABL.
7          ;   R5 = 64-byte physical memory block number where seg is to be loaded.
8          ;
9          ; Outputs:
10         ;   R5 = Update 64-byte physical memory block pointer for next segment.
11         ;
12 025102   GETOVL:
13         ;
14         ; Allocate space for the overlay segment
15         ;
16 025102   166205 000000      SUB     OS$SIZ(R2),R5      ; Allocate space for overlay
17 025106   020527 001600      CMP     R5,#1600          ; Are we about to run over RT-11?
18 025112   103405              BLO     10$              ; Br if yes -- Insufficient memory
19         ;
20         ; Remember the base address of some key segments
21         ;
22 025114   004737 004604'     CALL    KEYSEG            ; Remember address of some segments
23         ;
24         ; Load the segment
25         ;
26 025120   004737 025146'     CALL    LODOVL           ; Load the segment
27         ;
28         ; Finished
29         ;
30 025124   000207              RETURN
31         ;
32         ; Error: Memory overflow
33         ;
34 025126   .PRINT #TSXHD
35 025134   .PRINT #TSXSIZ
36 025142   000137 004216'     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 025146 010146 LODOVL: MOV      R1,-(SP)
10 025150 010246      MOV      R2,-(SP)
11 025152 010346      MOV      R3,-(SP)
12 025154 010446      MOV      R4,-(SP)
13 025156 010546      MOV      R5,-(SP)
14
15          ;
16          ; Get info about size of the overlay and position within SAV file
17          ;
18 025160 016201 000004      MOV      OS$OVL(R2),R1 ;Get pointer to linker-built segment entry
19 025164 016103 000000G      MOV      0.SIZ(R1),R3 ;Get size of overlay segment (# words)
20 025170 016137 000000G 000142' MOV      0.BLK(R1),FILBLK;Get block in SAV file where segment starts
21 025176 010302      MOV      R3,R2 ;Get total number of words in segment
22 025200 062702 000377      ADD      #255.,R2 ;round to the nearest number of blocks
23 025204 000302      SWAB      R2 ;Divide by 256. words per segment
24 025206 042702 177400      BIC      #177400,R2 ;kill sign extension bits
25 025212 010561 000000G      MOV      R5,0.PAR(R1) ;Remember where segment is being loaded
26          ;
27          ; Read next block of overlay segment into low-memory buffer
28          ;
29 025216 013704 000152' 10$: MOV      WRKBUF,R4 ;Point to work buffer
30 025222      .READW #AREA,#17,R4,#256.,FILBLK ;read a block
31 025260 103466      BCS      22$ ;read error occurred
32          ;
33          ; Move from low buffer to high position in memory
34          ;
35 025262 012701 000000G      MOV      #VPAR5,R1 ;get the virtual address of the mapped region
36 025266 012700 000400      MOV      #256.,R0 ;obtain the number of words to move
37 025272 020300      CMP      R3,R0 ;Do we need to move as many as 256 words?
38 025274 103001      BHIS     2$ ;Br if yes
39 025276 010300      MOV      R3,R0 ;Get number of words to move for last block
40 025300 160003 2$: SUB      R0,R3 ;Get number of words left after this move
41 025302      DISABL ;** Disable interrupts **
42 025310 013746 000000G      MOV      @#KPAR5,-(SP) ;save the contents of the mapping register
43 025314 010537 000000G      MOV      R5,@#KPAR5 ;change the mapping register
44 025320 052737 000000G 000000G BIS      #MMENBL,@#SROMMR;enable memory management
45 025326 105737 000000G      TSTB     MEM256 ;Does machine have at least 256Kb of memory?
46 025332 001403      BEQ      11$ ;Br if not
47 025334 052737 000000G 000000G BIS      #EMMAP,@#SR3MMR ;enable extended memory addressing
48 025342 012421 11$: MOV      (R4)+,(R1)+ ;move into high memory
49 025344 077002      SOB      R0,11$
50 025346 105737 000000G      TSTB     MEM256 ;Does this machine have at least 256Kb?
51 025352 001403      BEQ      12$ ;Br if not
52 025354 042737 000000G 000000G BIC      #EMMAP,@#SR3MMR ;disable extended memory management
53 025362 042737 000000G 000000G 12$: BIC      #MMENBL,@#SROMMR;disable memory management
54 025370 012637 000000G      MOV      (SP)+,@#KPAR5 ;restore the mapping register
55 025374      ENABL ;** Enable interrupts **
56 025402 062705 000010      ADD      #10,R5 ;advance 64-byte block # by 512-bytes
57 025406 005237 000142'      INC      FILBLK ;increment file block #

```

```
58 025412 005302          DEC      R2          ; More to be copied?
59 025414 001402          BEQ      5$          ; Br if not
60 025416 000137 025216'  JMP      10$          ; Read and copy rest of mapped segment
61                                     ;
62                                     ; Finished loading the segment
63                                     ;
64 025422 012605          5$:    MOV      (SP)+, R5
65 025424 012604          MOV      (SP)+, R4
66 025426 012603          MOV      (SP)+, R3
67 025430 012602          MOV      (SP)+, R2
68 025432 012601          MOV      (SP)+, R1
69 025434 000207          RETURN
70                                     ;
71                                     ; Error occurred on read
72                                     ;
73 025436          22$:    .PRINT  #TSXHD      ; Print heading
74 025444          .PRINT  #RDERR      ; Read error
75 025452 000137 004216'  JMP      INISTP      ; Abort initialization
```

```

1          .SBTTL  GETSRT -- Load any shared run-time systems
2          ;-----
3          ; GETSRT is called to load into memory a shared run-time system.
4          ; Shared run-time systems are loaded into the top of memory.
5          ;
6          ; Inputs:
7          ; R1 = Pointer to shared run-time descriptor block.
8          ; R5 = 64-byte block number of top of free memory.
9          ;
10         ; Outputs:
11         ; R5 = New top of memory block number
12         ;
13 025456 010146 GETSRT: MOV     R1,-(SP)
14 025460 010246      MOV     R2,-(SP)
15 025462 010346      MOV     R3,-(SP)
16 025464 010446      MOV     R4,-(SP)
17         ;
18         ; See if this is a dummy run-time entry to allow for patching
19         ;
20 025466 021127 0000000 CMP     (R1),#DMYDEV ;Dummy run-time entry?
21 025472 001540      BEQ     7$ ;Br if yes
22         ;
23         ; Try to open a channel to run-time file
24         ;
25 025474      .LOOKUP #AREA,#1,R1 ;OPEN CHANNEL TO RUN-TIME FILE
26 025512 103010      BCC     8$ ;BR IF OPEN WAS SUCCESSFUL
27         ;
28         ; Cannot open shared run-time file.
29         ; See if he wants to abort or continue.
30         ;
31 025514 105737 0000000 TSTB    VINABT ;ABORT OR CONTINUE
32 025520 001132      BNE     9$ ;BR IF ABORT WANTED
33 025522 005061 0000000 CLR     RT$NAM(R1) ;Mark run-time as not-available
34 025526 005061 0000020 CLR     RT$NAM+2(R1)
35 025532 000520      BR      7$ ;GO LOAD NEXT RUN-TIME SYSTEM
36         ;
37         ; Set up information about position of run-time in physical memory
38         ;
39 025534 116102 0000000 B$:    MOVB    RT$SKP(R1),R2 ;GET # BLOCKS TO SKIP AT FRONT OF RUN-TIME
40 025540 042702 177400      BIC     #^C377,R2 ;CLEAR SIGN EXTENSION
41 025544 160200      SUB     R2,R0 ;GET # BLOCKS TO READ (LOOKUP SET ROW SIZE)
42 025546 010561 0000000 MOV     R5,RT$TOP(R1) ;SET 64-BYTE BLOCK # ABOVE TOP OF RUN-TIME
43 025552 010003      MOV     R0,R3 ;GET # 512-BYTE BLOCKS IN RUN-TIME
44 025554 072027 0000003 ASH     #3,R0 ;CONVERT TO # 64-BYTE BLOCKS
45 025560 160005      SUB     R0,R5 ;CALCULATE BASE 64-BYTE BLOCK # OF RUN-TIME
46 025562 020527 001600      CMP     R5,#1600 ;ARE WE ABOUT TO RUN OVER RT-11?
47 025566 103530      BLO     11$ ;BR IF YES
48 025570 010561 0000000 MOV     R5,RT$BAS(R1) ;SET BASE 64-BYTE BLOCK # OF RUN-TIME
49         ;
50         ; Read run-time system into memory and position in high-memory
51         ;
52 025574 010546      MOV     R5,-(SP) ;Save address of bottom of run-time
53 025576 013704 000152' 4$:    MOV     WRKBUF,R4 ;Point to work buffer
54 025602      .READW #AREA,#1,R4,#256.,R2 ;READ A BLOCK OF RUN-TIME FILE
55         ; Use memory management to access high-memory area.
56 025636 012701 0000000 MOV     #VPAR6,R1 ;GET VIRTUAL ADDRESS OF PAR6 ADDRESS REGION
57 025642 010537 0000000 MOV     R5,@#UPAR6 ;SET USER-MODE PAR6 MAP OFFSET VALUE

```

```

58 025646 012737 077406 000000G      MOV      #077406,@#UPDR6 ;SET PDR TO ALLOW FULL ACCESS TO PAGE
59 025654 052737 000000G 000000G      BIS      #UPMODE,@#PSW  ;SET PREVIOUS-MODE = USER FOR MTPD ACCESS
60 025662 012700 000400                MOV      #256.,R0      ;GET # WORDS TO MOVE
61 025666                DISABL                ;** Disable interrupts **
62 025674 052737 000000G 000000G      BIS      #MMENBL,@#SROMMR;enable memory management
63 025702 105737 000000G                TSTB     MEM256        ;DOES THIS MACHINE HAVE AT LEAST 256KB?
64 025706 001403                BEQ      3$          ;BR IF NOT
65 025710 052737 000000G 000000G      BIS      #EMMAP,@#SR3MMR ;enable extended memory addressing
66 025716 012446                3$:      MOV      (R4)+,-(SP)    ;TRANSFER DATA FROM BUFFER TO HIGH MEMORY
67 025720 106621                MTPD      (R1)+
68 025722 077003                SOB      R0,3$
69 025724 105737 000000G                TSTB     MEM256        ;DOES THIS MACHINE HAVE AT LEAST 256KB?
70 025730 001403                BEQ      31$          ;BR IF NOT
71 025732 042737 000000G 000000G      BIC      #EMMAP,@#SR3MMR ;DISABLE EXTENDED MEMORY MANAGEMENT
72 025740 042737 000000G 000000G 31$:  BIC      #MMENBL,@#SROMMR;DISABLE MEMORY MANAGEMENT
73 025746                ENABL                ;** Enable interrupts **
74 025754 062705 000010                ADD      #10,R5         ;ADVANCE 64-BYTE BLOCK # BY 512-BYTES
75 025760 005202                INC      R2              ;INC FILE BLOCK #
76 025762 077373                SOB      R3,4$          ;READ AND COPY REST OF FILE
77                                ;
78                                ; Finished loading the run-time system.
79                                ;
80 025764 012605                MOV      (SP)+,R5
81 025766                .CLOSE    #1
82                                ;
83                                ; Finished
84                                ;
85 025774 012604                7$:      MOV      (SP)+,R4
86 025776 012603                MOV      (SP)+,R3
87 026000 012602                MOV      (SP)+,R2
88 026002 012601                MOV      (SP)+,R1
89 026004 000207                RETURN
90                                ;
91                                ; Error -- Cannot find run-time system file
92                                ;
93 026006                9$:      .PRINT  #TSXHD          ;PRINT MESSAGE HEADING
94 026014                .PRINT  #COSRT          ;PRINT ERROR MESSAGE
95 026022 012702 000004                MOV      #4,R2          ;PRINT 4 RAD50 VALUES
96 026026 012100                10$:     MOV      (R1)+,R0      ;GET PART OF NAME
97 026030 004737 030012'            CALL     PRTR50        ;PRINT RAD50 VALUE
98 026034 077204                SOB      R2,10$
99 026036                .PRINT  #CRLF          ;END LINE
100 026044 000137 004216'            JMP      INISTP        ;ABORT INITIALIZATION
101                                ;
102                                ; Error -- Insufficient memory space to load run-time systems
103                                ;
104 026050                11$:      .PRINT  #TSXHD          ;PRINT HEADING
105 026056                .PRINT  #SRTOVF        ;PRINT ERROR MESSAGE
106 026064 000137 004216'            JMP      INISTP        ;ABORT INITIALIZATION

```

```

1          .SBTTL  CSHBUF -- Allocate space for data cache tables
2          ;-----
3          ; Allocate space for data cache blocks and control tables.
4          ;
5          ; Inputs:
6          ;   R4 = 64-byte block number of base of free memory.
7          ;   R5 = 64-byte block number of top of free memory.
8          ;
9          ; Outputs:
10         ;   R5 = Updated 64-byte block number of top of free memory.
11         ;
12 026070 010246 CSHBUF: MOV     R2,-(SP)
13 026072 010346      MOV     R3,-(SP)
14         ;
15         ; See if data caching is wanted
16         ;
17 026074 013737 000000G 000000G      MOV     CSHALC,VCSHNB ;Set # blocks in use = # blocks allocated
18 026102 013702 000000G      MOV     CSHALC,R2      ;Did used request data caching?
19 026106 001464      BEQ     9$      ;Br if not
20         ;
21         ; Calculate number of 64-byte blocks needed for each cache control table
22         ;
23 026110 062702 000037      ADD     #31.,R2      ;Bound up to 32 word block
24 026114 072227 177773      ASH     #-5.,R2      ;Convert to # 64-byte blocks
25         ;
26         ; Compute total space that will be used by all cache data
27         ;
28 026120 013703 000000G      MOV     CSHALC,R3      ;Get # blocks in cache
29 026124 072327 000003      ASH     #3,R3      ;Get # 64-byte blks used by cache data buffers
30 026130 012700 000010      MOV     #8.,R0      ;Get number of cache control tables
31 026134 060203 1$: ADD     R2,R3      ;Accumulate total space needed
32 026136 077002      SDB     R0,1$
33 026140 010337 000000G      MOV     R3,CSHSIZ      ;Save total space used by cache data
34         ;
35         ; See if there is enough memory space available for the specified cache
36         ;
37 026144 010500      MOV     R5,R0      ;Get top of memory address
38 026146 160400      SUB     R4,R0      ;Compute # free 64-byte blocks
39 026150 020300      CMP     R3,R0      ;Is there enough total space?
40 026152 103045      BHIS    10$      ;Br if not
41         ;
42         ; Allocate space for cache data buffers
43         ;
44 026154 013700 000000G      MOV     CSHALC,R0      ;Get # blocks in data cache
45 026160 072027 000003      ASH     #3,R0      ;Get # 64-byte blocks needed for allocation
46 026164 160005      SUB     R0,R5      ;Allocate space for cache data buffers
47 026166 010537 000000G      MOV     R5,CSHBFP      ;Save pointer to base of buffer area
48         ;
49         ; Allocate space for each control table
50         ;
51 026172 160205      SUB     R2,R5      ;Allocate space for table
52 026174 010537 000000G      MOV     R5,CA$BLK      ;Block number associated with entry
53 026200 160205      SUB     R2,R5      ;Allocate space for table
54 026202 010537 000000G      MOV     R5,CA$DVU      ;Device and unit number
55 026206 160205      SUB     R2,R5      ;Allocate space for table
56 026210 010537 000000G      MOV     R5,CA$WCT      ;Number of words
57 026214 160205      SUB     R2,R5      ;Allocate space for table

```

```

58 026216 010537 0000000 MOV R5,CA$UFL ;LRU chain forward link
59 026222 160205 SUB R2,R5 ;Allocate space for table
60 026224 010537 0000000 MOV R5,CA$UBL ;LRU chain backward link
61 026230 160205 SUB R2,R5 ;Allocate space for table
62 026232 010537 0000000 MOV R5,CA$HFL ;Hash chain forward link
63 026236 160205 SUB R2,R5 ;Allocate space for table
64 026240 010537 0000000 MOV R5,CA$HBL ;Hash chain backward link
65 026244 160205 SUB R2,R5 ;Allocate space for table
66 026246 010537 0000000 MOV R5,CA$HSH ;Hash chain list heads
67 026252 020527 001600 CMP R5,#1600 ;Did we run over RT-11?
68 026256 101403 BLOS 10$ ;Br if yes
69 ;
70 ; Finished
71 ;
72 026260 012603 9$: MOV (SP)+,R3
73 026262 012602 MOV (SP)+,R2
74 026264 000207 RETURN
75 ;
76 ; Insufficient memory space available for cache data
77 ;
78 026266 10$: .PRINT #TSXHD ;Print heading
79 026274 .PRINT #CSHOVF ;Overflow message
80 026302 000137 004216' JMP INISTP ;Abort the initialization

```

CSHBUF -- Allocate space for data cache tables

```

1          . IF      EQ,PROCID      ;Don't allow ODT for production PRO version
2          . SBTTL   GETODT -- Load ODT
3          ;-----
4          ; GETODT is called to load ODT into memory above TSX and transfer control
5          ; to it. On return, ODT has been started.
6          ;
7          ; Inputs:
8          ; R5 = Address where ODT is to be loaded.
9          ;
10         ; Outputs:
11         ; R5 = Address above top of ODT.
12         ;
13 026306 010146 GETODT: MOV      R1,-(SP)
14 026310 010246      MOV      R2,-(SP)
15 026312 010346      MOV      R3,-(SP)
16 026314 010446      MOV      R4,-(SP)
17         ;
18         ; Try to lookup ODT rel file.
19         ;
20 026316          . LOOKUP #AREA,#1,#ODTBLK ;LOOKUP ODT REL FILE
21 026336 103010   BCC      1$          ;BR IF FOUND IT
22 026340          . PRINT #TSXHD          ;CAN'T FIND ODT
23 026346          . PRINT #NOODT
24 026354 000137 004216' JMP      INISTP          ;ABORT INITIALIZATION
25         ;
26         ; Read first block of ODT file and determine size of ODT.
27         ;
28 026360 062705 000310 1$:      ADD      #200.,R5          ;RESERVE SPACE FOR ODT STACK
29 026364 010500      MOV      R5,R0          ;CHECK MEMORY ADDRESS FOR OVERFLOW
30 026366 062700 001000      ADD      #512.,R0
31 026372 004737 027622'      CALL     CHKMEN
32 026376          . READW #AREA,#1,R5,#256.,#0
33 026432 103002      BCC      2$          ;Br if no read error
34 026434 000137 027016'      JMP      ODTRDX          ;Read error
35 026440 016502 000052      2$:      MOV      RSZ(R5),R2          ;GET SIZE OF ODT
36 026444 010203      MOV      R2,R3
37 026446 060503      ADD      R5,R3          ;GET ADDRESS ABOVE TOP OF ODT
38 026450 010300      MOV      R3,R0          ;CHECK MEMORY ADDRESS FOR OVERFLOW
39 026452 004737 027622'      CALL     CHKMEN
40 026456 000241      CLC
41 026460 006002      ROR      R2          ;GET # WORDS IN ODT
42         ; Get starting address of ODT
43 026462 016500 000040      MOV      STA(R5),R0          ;GET OFFSET TO START ADDRESS
44 026466 162700 001000      SUB      #ODTBAS,R0          ;CALCULATE ABSOLUTE STARTING ADDRESS
45 026472 060500      ADD      R5,R0
46 026474 010037 000300'      MOV      R0,ODTSTA          ;THIS IS REAL STARTING ADDRESS
47 026500 016501 000062      MOV      RBD(R5),R1          ;GET # OF BLOCK WITH RELOCATION INFO
48         ;
49         ; Read in ODT rel file image.
50         ;
51 026504          . READW #AREA,#1,R5,R2,#1
52 026540 103526      BCS      ODTRDX          ;BR IF READ ERROR
53         ;
54         ; Relocate addresses in ODT.
55         ; R5 = Address of base of ODT; R3 = Address above top of ODT.
56         ; R1 = Block number in rel file of start of relocation info.
57         ;

```

GETODT -- Load ODT

```

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

```

```

115 026742 012703 0000000 MOV    #TSEXEC,R3
116 026746 012704 0000000 MOV    #TSEMT,R4
117 026752 012746 0000000 MOV    #SYSMAP,-(SP) ; PASS ADDRESS OF SYSMAP CELL TO ODT
118 026756 012746 0000000 MOV    #MAPSYS,-(SP) ; PASS ADDRESS OF MAPSYS ROUTINE
119 026762 012705 026772' MOV    #10$,R5 ; ADDRESS FOR ODT TO RETURN TO
120
121 ; Enter ODT
122
123 026766 000177 151306 JMP     @ODTSTA ; JUMP TO START OF ODT
124
125 ; Return from ODT.
126 ; Continue initialization of TSX.
127
128 026772 013737 000014 0000000 10$: MOV    @#14,ODTTRP ; SAVE ODT BREAKPOINT ENTRY ADDRESS
129 027000 013705 000036' MOV    ODTTOP,R5 ; ADDRESS ABOVE TOP OF ODT
130 027004 012604 MOV    (SP)+,R4
131 027006 012603 MOV    (SP)+,R3
132 027010 012602 MOV    (SP)+,R2
133 027012 012601 MOV    (SP)+,R1
134 027014 000207 RETURN
135
136 ; Error while reading ODT rel file.
137
138 027016 ODTRDX: .PRINT #TSXHD ; PRINT ERROR MESSAGE
139 027024 .PRINT #ODTRDM
140 027032 000137 004216' JMP     INISTP ; ABORT INITIALIZATION
141
142 ; RTI instruction to disable interrupts
143
144 027036 000002 DORTI: RTI
145 .ENDC ; EQ,PROCID

```

OPNCHN -- Open a TSX-Plus channel

```

1          .SBTTL  OPNCHN -- Open a TSX-Plus channel
2          ;-----
3          ; OPNCHN is called to set up information in a TSX-Plus channel block
4          ; to make it look as if the channel has been opened to a specified
5          ; device with a .ENTER.
6          ;
7          ; Inputs:
8          ;   R0 = Address of channel block to be opened.
9          ;   R2 = Rad50 device name.
10         ;
11         ; Outputs:
12         ;   C-flag set ==> Cannot open the device.
13         ;
14 027040 010146 OPNCHN: MOV     R1, -(SP)
15 027042 010346      MOV     R3, -(SP)
16 027044 010003      MOV     R0, R3          ; Carry channel block address in R3
17         ;
18         ; Initialize the channel block
19         ;
20 027046 010301      MOV     R3, R1          ; Point to the channel block
21 027050 012700 000000C      MOV     #<CHNSIZ/2>, R0 ; Get # words to zero
22 027054 005021 2$:      CLR     (R1)+      ; Zero the channel block
23 027056 077002      SOB     R0, 2$
24 027060 012763 000000C 000000G      MOV     #<CS$OPN!CS$ENT>, C.CSW(R3) ; Initialize CSW to say chan open
25         ;
26         ; Convert the device name into device # and unit #
27         ;
28 027066 010200      MOV     R2, R0          ; Get the full device name
29 027070 004737 012632'      CALL    CVTDVU      ; Convert to dev # and unit #
30 027074 103411      BCS     9$              ; Br if we don't recognize the device name
31 027076 010001      MOV     R0, R1          ; Get index # and unit #
32 027100 000301      SWAB    R1              ; Get unit # to low byte
33 027102 110163 000000G      MOVB   R1, C.DEVQ(R3) ; Set unit # in channel block
34 027106 042700 000000C      BIC     #^C<CS$NMX>, R0 ; Clear all but device index number in R0
35 027112 050063 000000G      BIS     R0, C.CSW(R3) ; Store device index # into CSW
36         ;
37         ; Success
38         ;
39 027116 000241      CLC                     ; Signal success on return
40         ;
41         ; Finished
42         ;
43 027120 012603 9$:      MOV     (SP)+, R3
44 027122 012601      MOV     (SP)+, R1
45 027124 000207      RETURN

```

```

1          .SBTTL  SETCHN -- Copy RT-11 channel information into TSX system chan
2          ;-----
3          ; SETCHN is called to set up a TSX system channel block to access a file
4          ; that has been opened using RT-11.  The device index number is converted
5          ; from the RT-11 device number to the corresponding TSX device number.
6          ; Note: the channel must have been opened with a .lookup (not .enter)
7          ; to use this routine.
8          ;
9          ; Inputs:
10         ;   Channel # 1 = Open to file of interest.
11         ;   R0 = Address of TSX channel block which is to be set up.
12         ;   R2 = Rad-50 device name.
13         ;
14         ; Outputs:
15         ;   Channel block pointed to by R0 is set up for future TSX I/O.
16         ;   Channel # 1 is closed.
17         ;
18 027126 010146 SETCHN: MOV     R1, -(SP)
19 027130 010246         MOV     R2, -(SP)
20 027132 010346         MOV     R3, -(SP)
21 027134 010001         MOV     R0, R1          ; GET ADDRESS OF TSX CHANNEL BLOCK
22         ;
23         ; Do .SAVESTATUS to store channel information into TSX channel block.
24         ;
25 027136         .SAVEST #AREA, #1, R1        ; STORE CHANNEL STATUS INTO TSX CHANNEL BLOCK
26         ;
27         ; Now convert RT-11 device table index number into corresponding TSX
28         ; device table index number.
29         ;
30 027154 011103         MOV     (R1), R3        ; GET CSW FOR CHANNEL
31 027156 042703 177701  BIC     #^C76, R3      ; GET RT-11 DEVICE INDEX NUMBER
32 027162         .GVAL    #AREA, #404         ; GET RT-11 OFFSET TO PNAME TABLE
33 027202 060003         ADD     R0, R3         ; GET ADDRESS OF NAME OF DEVICE IN PNAME TABLE
34 027204         .GVAL    #AREA, R3          ; GET NAME OF DEVICE FROM RT-11
35 027222 013703 000000G MOV     NUMDEV, R3    ; GET INDEX # FOR LAST TSX DEVICE
36 027226 020063 000000G 1$: CMP     R0, PNAME(R3) ; LOOK FOR DEVICE IN OUR TABLES
37 027232 001404         BEQ     2$           ; BR IF FOUND
38 027234 162703 000002  SUB     #2, R3        ; CHECK NEXT ENTRY
39 027240 002372         BGE     1$           ; BR IF MORE TO CHECK
40 027242 000407         BR      MTSXDV       ; VERY STRANGE THAT WE DIDN'T FIND IT
41 027244 042711 000076  2$: BIC     #76, (R1)  ; CLEAR OUT RT-11 DEVICE #
42 027250 050311         BIS     R3, (R1)     ; STORE TSX DEVICE #
43         ;
44         ; Finished
45         ;
46 027252 012603         MOV     (SP)+, R3
47 027254 012602         MOV     (SP)+, R2
48 027256 012601         MOV     (SP)+, R1
49 027260 000207         RETURN
50         ;
51         ; Error: Could not locate Rt-11 device number in TSX device table.
52         ;
53 027262 MTSXDV: .PRINT  #TSXHD              ; PRINT ERROR MESSAGE
54 027270         .PRINT  #REQMIS             ; Missing a required device
55 027276 010200         MOV     R2, R0        ; GET RAD50 DEVICE NAME
56 027300 004737 030012' CALL     PRTR50      ; DISPLAY DEVICE NAME
57 027304         .PRINT  #CRLF

```

TSINIT -- TSX startup initializ MACRO V05.04 Monday 14-Dec-87 08:35 Page 77-1
SETCHN -- Copy RT-11 channel information into TSX system chan

58 027312 000137 004216'

JMP INISTP

;ABORT INITIALIZATION

SETSY -- Set up information about SY device

```

1          .SBTTL  SETSY  -- Set up information about SY device
2          ;-----
3          ; SETSY is called to set up information about the SY device.
4          ; It does this by determining what device RT-11 recognizes as SY.
5          ;
6          ; Inputs:
7          ;   R5 = Address of base of free memory area
8          ;
9          ; Outputs:
10         ;   SYNAME = RAD50 spec for physical system disk
11         ;   SYINDEX = TSX device table index for SY device
12         ;   SYUNIT = SY device unit number
13         ;
14 027316 010146 SETSY:  MOV     R1,-(SP)
15 027320 010246      MOV     R2,-(SP)
16         ;
17         ; Set up system device unit number
18         ;
19 027322      .GVAL    #AREA,#274      ;Get system unit # from RT-11 (high byte)
20 027342 010037 000000G      MOV     R0,SYUNIT      ;Set system unit number
21         ;
22         ; Set up system device index number
23         ;
24 027346      .GVAL    #AREA,#364      ;Get RT-11 system device index number
25 027366 010002      MOV     R0,R2      ;Save device index number
26 027370      .GVAL    #AREA,#404      ;Get offset within RMON of PNAME table
27 027410 060002      ADD     R0,R2      ;Get offset to name of SY device
28 027412      .GVAL    #AREA,R2      ;Get name of RT-11 system device
29 027430 013701 000000G      MOV     NUMDEV,R1      ;Get index to last TSX-Plus device entry
30 027434 020061 000000G 1$:  CMP     R0,PNAME(R1)    ;Search for device in TSX tables
31 027440 001405      BEQ     2$         ;Br if found it
32 027442 162701 000002      SUB     #2,R1      ;Keep looking if more
33 027446 002372      BGE     1$
34 027450 010002      MOV     R0,R2      ;Save name of system device
35 027452 000703      BR      MTSXDV      ;Missing device error
36 027454 010137 000000G 2$:  MOV     R1,SYINDEX    ;Store index # of TSX-Plus system device
37         ;
38         ; Set up RAD50 name of SY disk
39         ;
40 027460 113702 000001G      MOVB    SYUNIT+1,R2    ;GET SYSTEM UNIT NUMBER
41 027464 062702 000036      ADD     #36,R2      ;PUT IN "0" AS 3'RD CHARACTER OF NAME
42 027470 066102 000000G      ADD     PNAME(R1),R2    ;ADD DEVICE NAME
43 027474 010237 000000G      MOV     R2,SYNAME      ;THIS IS THE FULL SY DISK NAME
44         ;
45         ; Finished
46         ;
47 027500 012602      MOV     (SP)+,R2
48 027502 012601      MOV     (SP)+,R1
49 027504 000207      RETURN

```

RTFTCH -- Fetch a RT-11 device handler

```

1          .SBTTL  RTFTCH -- Fetch a RT-11 device handler
2          ;-----
3          ; RTFTCH is called to fetch an RT-11 device handler.
4          ; If the handler is already resident, nothing is done.
5          ; If the handler will fit in WRKBUF, it is fetched into there.
6          ; If the handler will not fit in WRKBUF, it is fetched into the top
7          ; of memory.
8          ;
9          ; Inputs:
10         ; R0 = RAD50 device name.
11         ; R5 = Address of start of free memory.
12         ;
13         ; Outputs:
14         ; C-flag cleared ==> Fetch was successful.
15         ; C-flag set      ==> Error on fetch.
16         ;
17 027506 010046 RTFTCH: MOV     R0,-(SP)
18 027510 010246      MOV     R2,-(SP)
19 027512 010546      MOV     R5,-(SP)
20         ;
21         ; Set the name of the device being fetched
22         ;
23 027514 010037 000130'      MOV     R0,FETDEV      ;Set name of device whose handler to fetch
24         ;
25         ; Do a .DSTAT to get information about the handler
26         ;
27 027520      .DSTAT #DSTBLK,#FETDEV ;Get information about the device handler
28 027532 103425      BCS     9$      ;Br if device not recognized
29         ;
30         ; Determine if the handler is currently resident
31         ;
32 027534 005737 000116'      TST     DSTBLK+4      ;Is the handler resident now?
33 027540 001021      BNE     8$      ;Br if yes
34         ;
35         ; The handler is not currently resident.
36         ; See if it will fit in WRKBUF.
37         ;
38 027542 013702 000152'      MOV     WRKBUF,R2      ;Set address where handler will be loaded
39 027546 013700 000114'      MOV     DSTBLK+2,R0    ;Get the size of the handler
40 027552 020037 000154'      CMP     R0,WRKSIZ      ;Will handler fit in WRKBUF?
41 027556 101405      BLOS    1$      ;Br if handler will fit in WRKBUF
42         ;
43         ; Handler will not fit in WRKBUF.
44         ; See if there is room to load it into the top of memory.
45         ;
46 027560 060500      ADD     R5,R0      ;Get address above top of area needed
47 027562 020037 000132'      CMP     R0,TOPMEM      ;Is there room for handler?
48 027566 101013      BHI     10$      ;Br if not
49 027570 010502      MOV     R5,R2      ;Set address where handler is to be loaded
50         ;
51         ; Fetch the handler
52         ;
53 027572      1$: .FETCH  R2,#FETDEV      ;Try to fetch the handler
54 027602 103401      BCS     9$      ;Br if error on fetch
55         ;
56         ; We successfully fetched the handler
57         ;

```

```
58 027604 000241      8$:      CLC                      ;Signal success on return
59                      ;
60                      ; Finished
61                      ;
62 027606 012605      9$:      MOV      (SP)+,R5
63 027610 012602              MOV      (SP)+,R2
64 027612 012600              MOV      (SP)+,R0
65 027614 000207              RETURN
66                      ;
67                      ; Insufficient memory available to load the handler
68                      ;
69 027616 004737 027642'    10$:     CALL      SIZERR          ;Generated system is too big -- abort
```

CHKMEM -- Check for memory space overflow

```

1          .SBTTL  CHKMEM -- Check for memory space overflow
2          ;-----
3          ; CHKMEM is called to make sure we have not overflowed the available memory
4          ; space while allocating space for TSX.
5          ; If a memory overflow occurs, an error message is printed and
6          ; the initialization is aborted.
7          ;
8          ; Inputs:
9          ;   RO = Address to be tested for validity.
10         ;
11 027622 020037 000302' CHKMEM: CMP      RO,MEMLIM      ; IS THE ADDRESS OK?
12 027626 103402          BLO      1$              ; BR IF OK
13 027630 004737 027642'          CALL     SIZERR      ; Generated system is too big -- abort
14 027634 004737 027662' 1$:      CALL     CCATST      ; CHECK FOR ^C ABORT REQUEST
15 027640 000207          RETURN
16
17         ;-----
18         ; Generated system is too big.  Abort the initialization.
19         ;
20 027642          SIZERR: .PRINT  #TSXHD          ; PRINT MESSAGE HEADING
21 027650          .PRINT  #TOOBIG         ; PRINT ERROR MESSAGE
22 027656 000137 004216'          JMP      INISTP      ; ABORT INITIALIZATION
23
24         ;-----
25         ; Check for control-C and abort initialization if requested.
26         ;
27 027662 005737 000040' CCATST: TST      CCAFLG      ; DID USER REQUEST ^C ABORT?
28 027666 001402          BEQ      1$              ; BRANCH IF NOT
29 027670 000137 004216'          JMP      INISTP      ; ELSE ABORT INITIALIZATION
30 027674 000207          1$:      RETURN

```

PRTDOCT -- Print octal value

```

1          .SBTTL  PRTDOCT -- Print octal value
2          ;-----
3          ; PRTDOCT is called to print an octal value without trailing Cr-Lf.
4          ;
5          ; Inputs:
6          ; R0 = value to be printed.
7          ;
8 027676 010146 PRTDOCT: MOV      R1,-(SP)
9 027700 010246      MOV      R2,-(SP)
10 027702 010001      MOV      R0,R1          ; GET VALUE TO PRINT
11 027704 012702 000006      MOV      #6,R2          ; PRINT 6 DIGITS
12 027710 005000      CLR      R0
13 027712 073027 000001      ASHC     #1,R0          ; GET 1ST OCTAL DIGIT (1 BIT)
14 027716 000403      BR       2$
15 027720 005000      1$: CLR      R0          ; INITIALIZE FOR SHIFT
16 027722 073027 000003      ASHC     #3,R0          ; SHIFT AN OCTAL DIGIT INTO R0
17 027726 062700 000060      2$: ADD     #'0,R0          ; CONVERT TO ASCII CHARACTER
18 027732          . TTYOUT          ; PRINT THE CHARACTER
19 027736 077210      SOB      R2,1$          ; LOOP AND PRINT MORE DIGITS
20 027740 012602      MOV      (SP)+,R2
21 027742 012601      MOV      (SP)+,R1
22 027744 000207      RETURN

```

```

1
2
3
4
5
6
7
8
9 027746 010146
10 027750 005046
11
12
13
14 027752 010001
15 027754 005000
16 027756 071027 000012
17 027762 062701 000060
18 027766 010146
19 027770 010001
20 027772 001370
21
22
23
24 027774 012600
25 027776 001403
26 030000
27 030004 000773
28
29
30
31 030006 012601
32 030010 000207

```

```

.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
;
PRTDEC: MOV     R1, -(SP)
        CLR     -(SP)          ; NULL ON STACK TO STOP US
;
; Convert value to ascii digit string and stack the digits.
;
        MOV     R0, R1          ; GET VALUE TO BE CONVERTED
1$:     CLR     R0              ; SET HIGH-ORDER PART OF VALUE TO 0
        DIV     #10., R0        ; DIVIDE R0-R1 BY 10.
        ADD     #'0, R1         ; CONVERT REMAINDER TO ASCII DIGIT
        MOV     R1, -(SP)       ; AND STACK THE DIGIT
        MOV     R0, R1          ; GET QUOTIENT
        BNE     1$              ; BR IF MORE DIGITS TO CONVERT
;
; Finished conversion. Print result.
;
2$:     MOV     (SP)+, R0        ; GET A DIGIT FROM THE STACK
        BEQ     3$              ; BR IF REACHED END
        .TTYOUT                    ; PRINT THE DIGIT
        BR      2$              ; PRINT MORE
;
; Finished
;
3$:     MOV     (SP)+, R1
        RETURN

```

PRTR50 -- Print Rad-50 value

```

1                                     .SBTTL  PRTR50 -- Print Rad-50 value
2                                     ;-----
3                                     ; PRTR50 is called to print a Rad-50 value.
4                                     ;
5                                     ; Inputs:
6                                     ; RO = value to be printed.
7                                     ;
8 030012 010146 PRTR50: MOV      R1,-(SP)
9 030014 010246      MOV      R2,-(SP)
10                                     ;
11                                     ; Convert value to ascii string and stack the characters.
12                                     ;
13 030016 012702 000003      MOV      #3,R2      ; GET # CHARS TO CVT
14 030022 010001      MOV      R0,R1      ; GET VALUE TO BE CONVERTED
15 030024 005000 1$: CLR      R0      ; CLEAR HIGH-ORDER VALUE
16 030026 071027 000050      DIV      #50,R0      ; DIVIDE R0-R1 BY 50
17 030032 116101 003464'      MOVB     R50CHR(R1),R1 ; CONVERT REMAINDER TO ASCII CHARACTER
18 030036 010146      MOV      R1,-(SP)      ; STACK THE CHARACTER
19 030040 010001      MOV      R0,R1      ; GET QUOTIENT
20 030042 077210      SOB      R2,1$      ; BR IF MORE CHARS TO CONVERT
21                                     ;
22                                     ; Finished conversion. Print the result.
23                                     ;
24 030044 012702 000003      MOV      #3,R2      ; GET # CHARS TO PRINT
25 030050 012600 2$: MOV      (SP)+,R0      ; GET NEXT CHARACTER
26 030052      . TTYOUT      ; PRINT THE CHARACTER
27 030056 077204      SOB      R2,2$      ; LOOP IF MORE CHARS TO PRINT
28                                     ;
29                                     ; Finished
30                                     ;
31 030060 012602      MOV      (SP)+,R2
32 030062 012601      MOV      (SP)+,R1
33 030064 000207      RETURN
34                                     ;-----
35                                     ; Define top of TSINIT
36                                     ;
37 030066 INITOP:
38                                     ;

```

```

1      .IF      NE,PROCID      ;Only assemble for protected Pro 350 version
2      ;
3      ; The following startup code is only included for the Pro version.
4      ; It is loaded here and executed very early during initialization
5      ; and subsequently overwritten by I/O buffers.
6      ;
7      .SBTTL  INSCHK  -- Installation validation subroutines for Pro-350
8      .MCALL  .PRINT
9      ;
10     ; Reserve an arg block area for encryption calls
11     ;
12     EDARGB: .WORD    -32.      ;# OF BYTES TO BE DECRYPTED (EDMTH3)
13     EDADDR: .WORD    DSKBUF    ;POINTER TO BUFFER TO BE DECRYPTED
14     ;
15     ; Recover license number and decrypt disk image of Pro ID to intermed. state
16     ;
17     INSCHK: MOV      R1,-(SP)    ;SAVE REGISTERS
18             MOV      R2,-(SP)
19             MOV      R3,-(SP)
20             MOV      R4,-(SP)
21             MOV      R5,-(SP)
22             MOV      (PC)+,R0    ;DECRYPT LICENSE NUMBER
23             .RAD50  /SCB/        ;WITH THIS CODE
24             XOR      R0,LICNUM   ;BY XORING IT
25             MOV      LICNUM,TSXSIT ;MOVE LICENCE NUMBER TO TSGEN CELL
26             MOV      #EDARGB,R0 ;POINT TO ENC/DEC ARG BLOCK (PRESET)
27             CALL     EDMTH3      ;DECRYPT TO INTERMED STATE
28     ;
29     ; Copy Pro ID ROM low bytes into memory, and encrypt 1 step
30     ;
31     IDADDR  = 173600            ;ADDRESS OF START OF PRO 350 ID ROM
32             MOV      #IDADDR,R1  ;GET POINTER TO PRO ID ROM
33             MOV      #ROMBUF,R2  ;POINTER TO COPY OF HARDWARE ID
34             MOV      R2,EDADDR   ;SAVE ADDRESS FOR ENCRYPTION
35             NEG      EDARGB      ;MAKE +32. FOR ENCRYPTION
36             MOV      EDARGB,R0   ;ALSO USE AS LOOP COUNTER
37     3$:      MOVB     (R1)+,(R2)+ ;GET NEXT LOW BYTE
38             INC      R1          ;SKIP ID ROM HIGH BYTES
39             SOB      R0,3$       ;REPEAT THROUGH 32 BYTE ROM
40             MOV      #EDARGB,R0 ;POINT TO ENCRYPTION ARG BLOCK
41             CALL     EDMTH2      ;PERFORM METHOD 2 ENCRYPTION
42     ;
43     ; Have intermediate state of both hardware and disk copies of Pro ID
44     ; in memory. Verify them against each other and correct memory image
45     ; of SCHED at the same time.
46     ;
47             MOV      #ROMBUF,R1  ;POINT TO HARDWARE COPY OF ID
48             MOV      #DSKBUF,R2 ;POINT TO DISK COPY OF ID
49             MOV      #SCHED,R4   ;POINT TO CODE TO BE CORRECTED
50             MOV      EDARGB,R3   ;INIT LOOP COUNTER
51             CALL     GETLIC      ;USE LIC # AS SEED FOR EDPRNW, IN R0
52     4$:      CMPB     (R1)+,(R2)+ ;VERIFY ID'S ARE THE SAME
53             BNE      5$          ;ABORT IF NO MATCH ON ANY BYTE
54             CALL     EDPRNW      ;RANDOMIZE R0 FOR XOR (LIC# INIT SEED)
55             MOV      @R4,R5      ;GET ENCRYPTED CODE
56             XOR      R0,R5       ;RESTORE FUNCTIONAL CODE
57             MOV      R5,(R4)+    ;PUT DECRYPTED CODE BACK IN MEMORY

```

```

58          SOB      R3, 4$          ; REPEAT THROUGH ID TESTS
59          MOV      (SP)+, R5        ; RESTORE REGISTERS
60          MOV      (SP)+, R4
61          MOV      (SP)+, R3
62          MOV      (SP)+, R2
63          MOV      (SP)+, R1
64          RETURN                    ; ID CHECKS AND CODE DECRYPTED
65          ;
66          5$:      .PRINT #TSXHD      ; ?TSX-F
67                  .PRINT #NOTLIC      ; NOT LICENSED FOR THIS MACHINE
68                  JMP      INISTP      ; ID'S DON'T MATCH, ABORT INIT.
69          ;
70          .NLIST   BEX
71          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          GETLIC:  MOV      LICNUM, R0      ; RETRIEVE DECRYPTED LIC # INTO R0
79                  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          DSKBUF:  .BLKB   32.              ; DISK IMAGE OF PRO ID
85          LICNUM:  .WORD   0                ; INCREMENTAL LICENSE NUMBER
86          ROMBUF:  .BLKB   32.              ; COPY OF ROM ID LOW BYTES

```

```

1          .SBTTL  EDEXPL -- Comments on encryption methods
2          ;
3          ; Encryption and decryption methods used here depend heavily on
4          ; pseudo-random numbers generated by the linear congrutential method.
5          ; See Hull and Dobell, SIAM Review, 4, 230, 1962.
6          ;
7          ; For the linear congruence relation:
8          ;
9          ;   X(I) == ( A * X(I-1) + C ) MOD M
10         ;
11         ;   X(I) is in the range 0 to M-1
12         ;
13         ; The sequence has full period M, provided that:
14         ;   1) C is relatively prime to M
15         ;   2) If p is a prime factor of M, A MOD p == 1
16         ;   3) If 4 is a factor of M, A MOD 4 == 1
17         ;
18         ; In the special case where M is a power of 2, these rules simplify to
19         ;   1) C must be odd
20         ;   2) A MOD 4 == 1
21         ;
22         .SBTTL  EDMTH2 -- Encryption method 2 (XOR with PRN high bytes)
23         ;
24         ; Using the license number as the initial seed, mask out the low 3 bits,
25         ; add 1 and call the PRN generator this many times to form the seed,
26         ; XOR each byte in the input buffer with the high byte of the next PRN
27         ; and replace the result in the input buffer. Decryption is accomplished
28         ; by a second application of the same process.
29         ;
30         ; Inputs:
31         ;   RO      Points to an arg block of the form:
32         ;           RO ---> buff_siz      ;word holding byte length of buffer
33         ;           buff_addr      ;address of buffer to be encrypted
34         ; Outputs:
35         ;   RO      Randomized
36         ;           input buffer encrypted
37         ;
38         EDMTH2:
39             MOV     R1, -(SP)      ; Save registers
40             MOV     R2, -(SP)
41             MOV     R3, -(SP)
42         ;
43         ; Fetch byte count, buffer pointer and initialize PRN seed
44         ;
45             MOV     (RO)+, R3      ; Fetch byte count of input buffer
46             MOV     (RO), R1       ; Fetch pointer to input buffer
47             CALL    GETLIC         ; Use license number as initial PRN seed
48             MOV     RO, R2         ; Copy license number to form repeat count
49             BIC     #^C7, R2      ; No more than 8 repeats
50             INC     R2             ; Make sure there is at least one
51         2$:      CALL    EDPRNW     ; Get a new PRN
52             SOB     R2, 2$        ; Advance the seed between 1 and 8 times
53         ;
54         ; Now sweep the buffer, XORing each byte with the high PRN byte
55         ;
56         1$:      CALL    EDPRNW     ; With seed in RO, get next random number
57             MOVB    (R1), R2      ; Get next input byte

```

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

```

1      .SBTTL  EDMTH3 -- Encryption/decryption meth 3 (swap bytes&shift bits)
2
3      ; Using a prn of repeat length same as input string length, select prn
4      ; numbered bytes from the input string, combine them into a word,
5      ; shift the combined bytes a random number of bits, recombine the shifted
6      ; bits and set the confused bytes back into the prn selected string bytes.
7      ; Sign of the byte count indicates: + = encryption; - = decryption.
8      ; If the byte count is 0 or 1, no encryption occurs. If the byte count is
9      ; odd, then one random selected byte will not be encrypted.
10
11     ; Inputs:
12     ;      R0      Points to an arg block of the form:
13     ;      R0 ---> buff_siz      ; Word holding byte length of buffer.
14     ;                               ; Note that buffer must be 512 or less
15     ;                               ; in length. Flag encryption by using
16     ;                               ; positive byte count ( 2 to 512. ).
17     ;                               ; Flag decryption by using negative
18     ;                               ; byte count (-2 to -512. ).
19     ;
20     ;                               buff_addr      ; Address of buffer to be encrypted
21     ; Outputs:
22     ;      R0      Randomized
23     ;      Input buffer encrypted
24
25     EDMTH3: MOV     R1, -(SP)      ; Save registers
26             MOV     R2, -(SP)
27             MOV     R3, -(SP)
28             MOV     R4, -(SP)
29             MOV     R5, -(SP)
30             MOV     (R0)+, R3      ; Save the string length
31             MOV     @R0, -(SP)    ; And save the input buffer pointer
32     ; Initialize prn generator of desired length
33             MOV     R3, R0
34             BGE     1$
35             NEG     R0
36             INC     R3
37             CALL    INPRNM        ; Set up for desired repeat length
38     ; Start encryption loop through string
39             ASR     R3
40             CALL    GETLIC        ; Repeat for 1/2 the string length
41             CALL    EDPRNM        ; Get lic. num. for initial seed in R0
42             TST     R3            ; Seed PRN generator (-adjacent pairs)
43             BEQ     9$            ; Less than 2 bytes left?
44             CLR     R4            ; Quit if so (odd len -> 1 byte unch.)
45             CLR     R5            ; Clean out shifting registers
46             MOV     @SP, R1
47             MOV     R1, R2        ; Retrieve buffer pointer
48             ; Select first random byte
49             CALL    EDPRNM        ; Randomize in range 0 - (strlen-1)
50             ADD     R0, R1        ; Point to first random byte of pair
51     ; Select second random byte
52             CALL    EDPRNM        ; Randomize again
53             ADD     R0, R2        ; Point to next random byte
54     ; Use part of PRNM as semi-random shift amount
55             MOV     R0, -(SP)
56             BIC     #^C6, R0      ; Save EDPRNM seed for later
57             ; Get a semi-random shift amount
58             ; Select encryption or decryption

```

```

58          TST      R3          ; Positive for encryption
59          BMI      3$          ; Branch if decrypting
60          ; Do this part for encryption
61          BISB     @R1,R4      ; Get first byte without sign extend
62          SWAB     R4          ; And put it in the high byte
63          BISB     @R2,R4      ; Combine it with first byte
64          CLC          ; Always do at least one shift
65          ROR      R4          ; Shift once
66          ROR      R5          ; Get low bit into r5
67          NEG      R0          ; Right shifts for encryption
68          DEC      R3          ; Reduce count of pairs remaining
69          BR       4$          ; Skip decryption stuff
70          ; Do this part for decryption
71          3$: BISB     @R1,R5      ; Get first byte without sign extend
72          SWAB     R5          ; And put it in the high byte
73          BISB     @R2,R5      ; Combine it with the first byte
74          CLC          ; Always do at least one shift
75          ROL      R5          ; Shift once
76          ROL      R4          ; Get high bit into R4
77          INC      R3          ; Reduce count of pairs remaining
78          ; Shift and recombine the (en)ide)rypted bytes
79          4$: ASHC     R0,R4      ; Shift combined bytes 0,2,4 or 6 more
80          BIS      R5,R4      ; Recombine bytes
81          MOV      (SP)+,R0      ; Recover EDPRNM seed
82          ; Now put encrypted bytes back into input string
83          MOVB     R4,@R2      ; Store low byte at second byte place
84          SWAB     R4          ; Get high byte
85          MOVB     R4,@R1      ; Store high byte at first byte place
86          BR       2$          ; Repeat through string
87          ; Done, restore registers and return
88          9$: MOV      (SP)+,R0      ; Just pop saved buffer address
89          MOV      (SP)+,R5      ; Restore registers
90          MOV      (SP)+,R4
91          MOV      (SP)+,R3
92          MOV      (SP)+,R2
93          MOV      (SP)+,R1
94          RETURN

```

```
1          .SBTTL  EDPRNW -- Pseudo random number generator with MOD 2^16
2          ;
3          ; Linear congruential pseudo-random number generator with maximum repeat
4          ; length of 65536 (2^16). cf. Hull and Dobell and Knuth, vol 2.
5          ;
6          ; Inputs:
7          ;      R0      Seed value
8          ;
9          ; Outputs:
10         ;      R0      New PRN, should be used for next seed
11         ;
12         EDPRNW:
13             MOV     R4, -(SP)      ; Save registers
14             MOV     R5, -(SP)
15             MOV     R0, R4        ; Get seed to be multiplied
16             MOV     (PC)+, R0     ; Fetch multiplier
17         EDPRNA: . WORD 104375     ; Multiplier, can be replaced
18             MUL     R0, R4        ; Multiply by A
19             ADD     (PC)+, R5     ; Add C
20         EDPRNC: . WORD 012705    ; Additive, can be replaced
21             MOV     R5, R0        ; Return result mod 65536. as PRN
22             MOV     (SP)+, R5     ; Restore registers
23             MOV     (SP)+, R4
24             RETURN
```

```
1          .SBTTL  INPRNM -- Initialize PRN generator with repeat range M
2          ;
3          ; Using the Hull and Dobell rules, determine acceptable values for
4          ; A and C to get a repeat range of M.
5          ;
6          ; Outputs:
7          ;     EDMULA Set with first acceptable multiplier
8          ;     EDADDC Set with first acceptable additive factor
9          ;     EDMODM Set with desired repeat length
10         ;
11         INPRNM:
12             MOV     #32,EDMODM      ;Get repeat length to cover Pro ID
13             MOV     #5,EDMULA       ;Use first valid A
14             MOV     #3,EDADDC       ;And first valid C
15             RETURN
```

PRTR50 -- Print Rad-50 value

```

1          .SBTTL  EDPRNM -- Generate pseudo-random number in specified range M
2          ;
3          ; *****
4          ; * INPRNM MUST BE CALLED BEFORE FIRST SEED IS PASSED TO THIS ROUTINE!!!! *
5          ; *****
6          ;
7          ; Using linear congruential method (cf. Hull and Dobell), generate
8          ; pseudo-random number using seed passed in R0. Return new PRN in R0.
9          ;
10         ; Inputs:
11         ;   R0      Seed value, must be in range 0 to M (EDMODM)
12         ;
13         ; Outputs:
14         ;   R0      New pseudo-random number, should be used for next seed
15         ;
16         EDPRNM:
17             MOV     R4, -(SP)      ; Save R4 and R5
18             MOV     R5, -(SP)
19             CMP     R0, EDMODM     ; Is seed in range 0 to EDMODM?
20             BLO     1$             ; Branch and proceed if so
21             MOV     R0, R5        ; Set up to divide it by EDMODM
22             CLR     R4            ; Set up for divide
23             DIV     EDMODM, R4    ; Divide it
24             MOV     R5, R0        ; And use remainder as seed
25         1$:      MOV     R0, R4    ; Get current seed ready to be multiplied
26             MUL     (PC)+, R4    ; Multiply by chosen A
27         EDMULA:  .WORD 25173.    ; Replace at run-time with 5
28             ADD     (PC)+, R5    ; Add in C
29         EDADDC:  .WORD 13849.    ; Replace at run-time with 3
30             CLR     R4            ; Clear high word for division
31             DIV     (PC)+, R4    ; Perform mod M
32         EDMODM:  .WORD 256.      ; Replace at run-time with 32.
33             MOV     R5, R0        ; Return remainder
34             MOV     (SP)+, R5    ; Restore R4 and R5
35             MOV     (SP)+, R4
36             RETURN
37         ;
38         . IFF      ; NE, PROCID      ; Assemble if protection code not included
39         DSKBUF:    ; Define dummy DSKBUF global symbol
40         . ENDC     ; NE, PROCID
41         ;
42         ; Address of real top of TSINIT, including PRO init code
43         ;
44         030066
45         000000
46         000001
47         PROITP:    .CSECT TSXEND
48                   .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: 00:03:19.50

TSINIT -- TSX startup initializ MACRO V05.04 Monday 14-Dec-87 08:35 Page 89-1
PRTR50 -- Print Rad-50 value

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

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

CL\$DRE	1-124	39-46*						
CL\$DRG	1-124	39-38*						
CL\$DRP	1-124	39-37*						
CL\$DRS	1-126	39-44*						
CL\$STA	1-126	39-83*						
CLDEVX	1-142	31-29	39-94*					
CLEOFS	1-125	39-53	39-54					
CLHEAD	1-142	39-99	39-117					
CLINCP	1-154	22-21						
CLINIT	24-73	39-11#						
CLK100	1-187#	5-165	23-14*					
CLKRTI	1-106	5-71						
CLKVEC	1-157	5-71*	5-72*	5-165*	23-14	24-78*		
CLORSZ	1-111	39-39						
CLOTIR	1-154	22-20						
CLSIZE	1-142	39-100						
CLSTS	1-117	39-96	39-114					
CLTOTL	1-112	24-71	31-18	39-18	39-101	39-108	69-149	
CLVEND	3-68#	39-132						
CLVERS	1-110	39-123	39-137*					
CLVTBL	3-60#	39-128						
CD\$BBT	1-129	39-74						
CD\$DEF	1-125	39-62						
CD\$FF	1-125	39-71						
CD\$TAB	1-125	39-68						
CONFQ2	1-104	23-165*	23-166*	23-169*	24-79			
CONFIG	1-103	23-163*	24-5					
CONSPC	4-13#	28-108						
COSRT	4-34#	73-94						
CRLF	4-9#	27-96	27-101	28-111	28-119	58-10	73-99	77-57
CS\$ENT	1-144	76-24						
CS\$NMX	1-88	76-34						
CS\$OPN	1-144	76-24						
CSHALC	1-121	35-71	69-161	74-17	74-18	74-28	74-44	
CSHBAS	1-104	7-39	9-37*					
CSHBFP	1-147	74-47*						
CSHBUF	24-206	74-12#						
CSHDEV	1-86	35-63*						
CSHDVN	1-86	35-67*						
CSHOVF	4-39#	74-79						
CSHSIZ	1-147	74-33*						
CSHVEC	1-105	7-42						
CURDEV	1-204#	59-54*	60-40	62-45				
CURNAM	1-205#	51-25*	56-20*	58-8				
CVTDVU	30-37	33-14#	76-29					
CW\$50H	1-103	24-5						
CW\$BTH	1-102	23-161						
CW\$ESP	1-112	23-166						
CW\$FB	1-102	23-162						
CW\$FGJ	1-102	23-162						
CW\$GDH	1-102	23-161						
CW\$LGS	1-102	23-161						
CW\$PRO	1-106	23-116	24-79					
CW\$QBS	1-114	23-169						
CW\$USR	1-103	23-162						
CW\$XM	1-103	23-162						

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

[illegible]

[illegible]

O. PAR	1-135	6-34*	72-25*																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
--------	-------	-------	--------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

PISRT	1-57	1-249#										
PMPAR	1-131	37-93*										
PMSIZE	1-131	37-91										
PNAME	1-89	24-47*	24-62*	33-32	39-95*	39-113*	51-45	53-29*	56-19	56-20	77-36	78-30
	78-42											
PPTERM	1-57	1-254#										
PROASM	1-20	1-32	1-57	1-63	5-29	5-143	10-1	13-1	14-1	24-38	24-187	27-1
	44-45	44-55	48-1	64-1								
PROBUF	1-58	1-206#	23-119*									
PROCID	1-24	23-95	23-146	75-1	75-60	84-1						
PROFLG	1-98	5-53	23-118*									
PROITP	1-57	89-44#										
PRTDEC	23-35	27-100	28-110	82-9#								
PRTDCT	27-95	81-8#										
PRTR50	28-118	30-63	58-9	73-97	77-56	83-8#						
PSW	1-94	44-71*	44-88*	56-71*	56-85*	61-18*	61-65*	72-41*	72-55*	73-59*	73-61*	73-73*
PTBYT	1-100	57-24										
PTWRD	1-100	57-23										
PVSPBL	1-114	30-115										
QBUS	1-137	23-167	44-117									
R50C1	1-222#	32-30	39-113									
R50C10	1-223#	32-32	32-36									
R50C17	1-224#	32-34										
R50CHR	4-42#	83-17										
R50CL	1-219#	32-17	39-95									
R50CLO	1-220#	32-19	32-23									
R50CL7	1-221#	32-21										
R50CSH	1-213#	9-35										
R5OLD	1-217#	24-62	51-30									
R5OLOK	1-211#	9-31										
R5OLS	1-226#											
R5OMSG	1-209#	9-19										
R50ODT	1-227#	23-87										
R50PI	1-218#											
R50SY	1-216#											
R50TID	1-214#	9-39										
R50TT	1-215#	24-47										
R50USR	1-212#	9-27										
R50VM	1-225#											
R50WIN	1-210#	9-23										
RBD	1-177#	75-47										
RBR	1-116	14-102										
RC#SZ	1-121	35-28										
RDB	1-120	24-178										
RDBEND	1-120	24-182										
RDERR	4-37#	6-46	66-52	68-53	72-74							
RDINT	1-116	14-106										
REDUCE	4-31#	23-33										
RELFIL	75-58#											
RELOC	1-101	57-27										
REQMIS	4-10#	77-54										
RID	1-176#											
RLBF	1-233#	75-59*	75-64*	75-65	75-70	75-75						
RLBFND	1-234#	75-65*	75-66*	75-87								
RMNPDR	1-112	45-56*										
RMON	1-104	5-17*	5-167*	23-13	23-115	23-159	52-91	52-92*	52-95*	61-22*	61-61*	

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

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

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

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

[illegible]