Table of contents

Table of contents

```
                                .TITLE  TSKMN3 -- TSKMON Subroutines
                                .ENABL  LC
                                .DSABL  GBL
                                .CSECT  TSKMN3
                        TSKMN3:
                        ;
                        ;  Copyright 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988.
                        ;  S&H Computer Systems, Inc.
                        ;  Nashville, Tennessee
                        ;
                        ;  Macro calls
                        ;
                                .MCALL  .CSISPC,.TTOUTR,.SRESET
                                .MCALL  .READW,.TTYIN,.TTYOUT,.PURGE
                                .MCALL  .CSIGEN,.SAVEST,.REOPEN
                                .MCALL  .GTLIN,.GTIM,.DATE
                                .MCALL  .PRINT,.CLOSE,.LOOKUP
                                .MCALL  .WRITW,.ENTER,.EXIT
                                .MCALL  .SERR,.HERR,.FPROT,.GVAL,.PVAL
                                .MCALL  .ELRG,.CRRG
                        ;
                        ;  Global references and definitions
                        ;
                                .GLOBL  AFCF,$SCCA,AF$CCA,LWINDO,AF$NPW,$NOWIN,LSW11
                                .GLOBL  INSTBL,INGADR,INGEMT,IIBUF,II$NAM,II$$SZ,INSTBN
                                .GLOBL  ACRTXT,CHKCLU,CFSTRT,CFSTOP,R$CFST,CFACFL,CXTRMN
                                .GLOBL  PEKEMT,PEKADR,PEKSIZ,PRVOPT,CL$XLN,ACRSTR
                                .GLOBL  ABRTAD,ABRTCD,CINFLG,$VNOTT,LSTSPL,EM$IST
                                .GLOBL  CORUSR,LSW,$CTRLO,SERFLG,IOABFL,TSKMN3
                                .GLOBL  UTRPAD,JSWLOC,ERRLOC,MAXMEM,CHKALC
                                .GLOBL  USRSTK,$KINIT,CFSTK,MXJMEM,DFJMEM
                                .GLOBL  SPUBUF,SXBPNT,MXJADR,CKCLUS
                                .GLOBL  TMTOTH,TMTOTL,TMUSRH,TMIOWH,CFSQEZ,LDCLEN,DATTIM
                                .GLOBL  TMSWTH,TMIDLH,TMIOH,TMSWPH,PRGALL,LDMNT
                                .GLOBL  WILDFL,$NOIN,$NOWTT,$HITTY,LPARNT,WLDNAM
                                .GLOBL  P2$UP1,P2$UP2,P2$UP3,P2$UP4,P2$CXT,$SUCF
                                .GLOBL  PO$BYP,PO$NFR,PO$NFW,PO$SPF,RUNFLG
                                .GLOBL  P2$TRM,EM$OPR,EM$SPR,EM$TPR,PO$$NP,P2$$NP
                                .GLOBL  PO$DBG,PO$SPV,CKACOJ,P2$WRL,P2$GRP,P2$SAM,EM$CAJ
                                .GLOBL  PO$DET,PO$MEM,P2$MSG,PO$OPR,PO$LOK,PO$RT
                                .GLOBL  PO$SND,PO$NAM,PO$SPV,P2$RLK,P2$CGR,PO$SYS
                                .GLOBL  P2$VIR,PO$ALC,PRIVC2,$NOVLN,LSW2S
                                .GLOBL  CHKEQ,CLRPRV,RSTPRV,PRVLST,FIXPRV,CCSPRV
                                .GLOBL  PVNPW,PFSO,PFCO,EM$CSE,PRIVCO,PRIVSO,PRIVFO
                                .GLOBL  CDGET,CDPUT,CDBUF,CDGEMT,CDGADR,CDPEMT,CDPADR
                                .GLOBL  TECO,EDIT,KED,K52,$1STLG,$DIBOL,SETWRD,ACRPRV
                                .GLOBL  R.GSTS,RS.CRR,RS.GBL,RS.PVT,RS.EGR,OPTLST
                                .GLOBL  SH$VAL,SH$NAM,SH$$SZ,SH$RTN,SH$FLG,SFCBSZ
                                .GLOBL  TM$CLG,EM$ENM,EM$IOV,EM$ISV,KUSECK,SCNOPS
                                .GLOBL  ALCDEV,DLCEMT,SFID,RUNCHN,GETKCH
                                .GLOBL  SO$NVL,SO$OCT,SO$NO,HANENT,HANSIZ
                                .GLOBL  HAZEL,HAZLFL,HAZLNO,$MLOCK,MDT,LSW9
                                .GLOBL  LINBUF,LINNXT,LSTACT,PRGTOP,PRGSIZ,KMNHI
                                .GLOBL  KMNTOP,KMNPGS,KMNSTK,KMNSTR,CXTPAG
                                .GLOBL  LINPNT,LINCNT,LACTIV,LRDTIM,CS$RON
                                .GLOBL  LOTBUF,LOTNXT,LOTPNT,$VTESC
                                .GLOBL  LOTSIZ,LOTSPC,LCOL,TK1SEC,$NTGCC
```

```
                        .GLOBL  LAFSIZ,LFWLIM,LINCUR,NUMON,ILSW2
                        .GLOBL  $CARUP,DOASGN,LOGCHK,LOGDVU,LOGBAS
                        .GLOBL  LSUCF,$CCLRN,TALEMT,ALCDEV,EM$DAA,EM$DIU
                        .GLOBL  KL3CLR,$PRGLK,LSW5,PVON,S$SPND
                        .GLOBL  S$TWFN,S$TTFN,S$OTFN,S$IOFN,S$OTLO
                        .GLOBL  LSTDL,FSTDL,$DETCH,UMSYTP
                        .GLOBL  $DISCN,LPROJ,LPROG,LUNAME
                        .GLOBL  LCPUHI,LCPULO,LCONTM,$CTRLS,$SPLJB
                        .GLOBL  STPFLG,TOTON,USPLCH,SPLCHN,$CFKIL
                        .GLOBL  S$INWT,S$OTWT,S$TMWT,S$SFWT,$INDAB
                        .GLOBL  S$MSWT,CFBUF,CFEND,CCLSAV,KMNCHN
                        .GLOBL  MINTIM,LSECPT,MAXSEC,$EMTTR,SC$WRN
                        .GLOBL  OKFILE,OKFEND,OKFAND,OKFNND
                        .GLOBL  $CLTST,SKPSPC,SKPDLM,SC$SEV
                        .GLOBL  LJSW,CTRLTT,NEWJSW,JSTKND,VIMAGE
                        .GLOBL  USTART,GENTOP,BOTDEV,BOTUNI,BOTCSR
                        .GLOBL  $CTRLC,LSW2,$INKMN,CHAIN,UFORM
                        .GLOBL  MAXASN,$CFABT,INDSTA,INDERR
                        .GLOBL  AT$LOG,AT$SIZ,AT$DEV,AT$FIL,AT$EXT,AT$$SZ
                        .GLOBL  RUNDEV,LNBLKS,CXTBAS,CXTWDS,UHIMEM
                        .GLOBL  ASNTBL,$DILUP,CSHDEV,CSHDVN,LNSBLK
                        .GLOBL  ASNEND,LSW3,$DUPRN
                        .GLOBL  $FORM,$TAB,LSCCA,$CFSOT
                        .GLOBL  $PAGE,$SCOPE,$ECHO,$LC
                        .GLOBL  UCHAN,$FORMO,$CFALL,$CFDCC,$CFCCL
                        .GLOBL  LNPRIM,LNMAP,CW$50H,CONFIG
                        .GLOBL  $DOOFF,NUCHN,LRBFIL,CFIND
                        .GLOBL  C.CSW,C.DEVQ,C.SBLK,NLINES
                        .GLOBL  CD$NAM,CD$DVU,CD$BAS,CD$JOB,CD$$SZ,CD$$UB
                        .GLOBL  LTSCMD,LNSPAC,CFNEST,UCLNAM
                        .GLOBL  $CFOPN,CFSEND,PBFEND,CFSP,$TTGAG
                        .GLOBL  UFPTRP,SDSFCB,SD$DEL,CFLFL4,$UCLCF
                        .GLOBL  SDFLAG,SD$FLK,SD$WFM,SDFORM,$UCLRN
                        .GLOBL  SDBUF1,SDBLK,SPLND,LD$RON
                        .GLOBL  LDNAME,LDSIZE,LDFLAG,LDBASE,LDPDEV
                        .GLOBL  $DEFER,CFCHAN,SCHAIN,LDDEVX,CLDEVX
                        .GLOBL  CFPNT,CFBLK,$QUIET,DIABFL
                        .GLOBL  DIABNO,VT52NO,LA36NO,LA36FL
                        .GLOBL  LSW4,KL4CLR,SDSKIP,SDBU,SD$BAK
                        .GLOBL  $INCOR,$KED,TK5VAL
                        .GLOBL  SF$BSY,SFFORM,SD$SNG,SFNMBL,NFRESB
                        .GLOBL  SD$HLD,SF$HLD,CURPRM,PRMPNT,SF$1ST
                        .GLOBL  LSTPRM,PRMBUF,PRMEND,CFSPND
                        .GLOBL  SDFHD,SFFLAG,SFQLNK,CFHOLD
                        .GLOBL  LCOL,$QTSET,$TECO,CD$TOP,POPCF
                        .GLOBL  $WILD,ERRSEV,UERSEV,PASLIN
                        .GLOBL  LSTPL,SDCB,SDCBND
                        .GLOBL  VQUAN1,VQUN1A,VQUAN2,VHIPCT
                        .GLOBL  DCTRD,DCCRD,DCTWR,DCCWR
                        .GLOBL  VCORTM,KMPRMT,MXPRMT
                        .GLOBL  RDB,RDBEND,RT$NAM,RT$$SZ
                        .GLOBL  SDNAME,SDCBSZ,LSTSL,LSTATE
                        .GLOBL  TK1VAL,CINDAT,SYSDAT,SYTIMH,SYTIML
                        .GLOBL  BASMAP,LOMAP,HIMAP,JCXPGS
                        .GLOBL  TSXLN,TSXSIT,GRT1,TRGRET,LICTXT,SUPCOD,NAMTOP,SUMS,SUCS
                        .GLOBL  LPRG1,LPRG2,S$QUSR,S$IOWT,S$SFWT
                        .GLOBL  S$SPDB,S$SPCB,SFUSER,SFFILE,VT2007,VT2008
```

```
                .GLOBL  LCBIT,LA36,LA120,VT52,VT100,DIABLO,QUME
                .GLOBL  ADM3A,LTRMTP,LA12FL,LA12NO,VT52FL
                .GLOBL  VT10FL,VT10NO,QUMEFL,QUMENO,ADM3FL
                .GLOBL  ADM3NO,SYINDX,SYUNIT,NUMDEV,PNAME
                .GLOBL  OF$DEV,OF$UNT,OF$FIL,OF$FLG,SYNAME
                .GLOBL  OF$$SZ,OT$RON,RESDEV,$TAPE
                .GLOBL  KMNBAS
                .GLOBL  LSW6,$SNWTT,PF$SYS,PF$IOW
                .GLOBL  RSR,TSR,LMXNUM,LSTMX,MXDTR,ZCLR,MXCSR
                .GLOBL  $INDDF,$INDRN,IN$ACT,IN$CNT,IN$CMD,INDSAV
                .GLOBL  $PHONE,INVEC,LMXLN,MXVEC,$INIT,$DEAD
                .GLOBL  ITRMTP,LMXPRM,LSW7,CFSTS,CF$IND,CF$QUT
                .GLOBL  CFABLV,MONVEC,CVTUC,INDABT
                .GLOBL  LOGCHN,LOGFLG,LOGPTR,LOGBUF,LOGBLK
                .GLOBL  LF$OPN,LF$WRT,UCLBLK,UCLDAT
                .GLOBL  CSHHD,FC$CDX,FC$LNK,FD$NAM,UC$NDC,UC$MDC
                .GLOBL  CMDBUF,PAUMSG,RDCMD,DKSAV,SYSAV,CVTTAB,RUNHD,SEARCH
                .GLOBL  INVOPT,FKILL,ABRTCF,ACRFN,XAREA,FILNAM,NOPRG,FPRINT
                .GLOBL  PUSHCF,TRMSTR,FILNAM,R50DIR,R50SY,R50IND,PRTWRN
                .GLOBL  INDACT,R50DUP,R50PIP,R50KED,R50K52,R50KEX,WRNHED
                .GLOBL  BLKO,RDERM,R50VIR,NOSTRT,RUNEMT,OVRCOR
                .GLOBL  BADSAV,LDNAM,NOPRG,NOCIN,SIZVAL,ASKLNM,BADCMD,KCSIBF
                .GLOBL  ASDEX,GTRD50,R50BUF,R50LDO,MNTDEV,DMTARG
                .GLOBL  DEADEV,CHKMNT,CHKMTX,INFOMT,NOFLAG,MTOPHD,INVOPT,ILLCMD
                .GLOBL  R50LD,INVLDN,R50DSK,ACRFIL,BDFNAM,LOGASN,MNTFUL,R50LD7
                .GLOBL  TBLOVF,SETHD,CSIMS2,CKPRIV,R50NO,AMBOPT,ACRDEC,CKTERM
                .GLOBL  MAXAVL,PRTDEC,DEVUNT,PNAME,HANIDX,HNBUF,ACRSPD
                .GLOBL  ACROCT,HANBSY,CSIMS1,MISSEQ,NOIND,CKSYPV
                .GLOBL  BADPMT,BADPRI,TOTXT,CRLF,HIPRI,STLGHD,LOGCLS,R50LOG
                .GLOBL  BDLGOP,SPLHLA,NOCCL,LDOPHD,PRTFIX,PRTSPC
                .GLOBL  DLTXT,OCTFIX,PRTTTP,NATXT,NOTXT,YESTXT,NINTXT
                .GLOBL  PRTUNM,SYHD1,SYHD2,PRTLN,SPACE2,DETTXT,SPACE3,RNMS
                .GLOBL  SWPTX,LOCKTX,SPACE5,PRTDC3,KBMSG,DIVIDE,PRTDC2
                .GLOBL  COLOO,CPUAH,CPUAL,PRTTMV,NOFIL,CMDBUF,CALUCL
                .GLOBL  NOUDC,DEVHD1,ASNHD1,ASNHD2,SHMTH1,SHMTH2
                .GLOBL  CVDVNM,SPACE6,PRTBUF,PRTFNM,NONEMS,NODAT,NOLDMT
                .GLOBL  SUBARO,EDTFIL,RONTXT,NOTAVL,KBTX,PRTTMD
                .GLOBL  DELSPC,MONHD,MONAR1,NOPMGN,PMBUSY,MONAR2
                .GLOBL  NSWPMS,MAXMTX,CURMTX,SDNAME,CHKDLM,SPLHD,INVOPT
                .GLOBL  DEVIDL,COAL,ALDEX,COAD,SPACTV,SPWFM,DEVIDL,SPSNG
                .GLOBL  COAL,ALDEX,ALDBLK,COAD,SPACTV,SPWFM,DEVIDL
                .GLOBL  SPSNG,SPFUL,SPCF,SPFLK,NOFIL,SPGEMT,NOOPTT
                .GLOBL  BDLIN,MSGBUF,MSGEND,NOTON,GAGMSG,CHKTTD
                .GLOBL  LINFRE,DJABMS,DLMSG,INVTIM,DMTALL,H.CSR
                .GLOBL  SHTMSG,AUTHFN,SPLACT,DOSTOP,OFFEMT,KILEMT,UPTMMS
                .GLOBL  TMTOTH,DIVSOR,TMTOTL,PRTPCT,SUM1,SUM2,SUM3,SUM4
                .GLOBL  SUM5,SUM6,SUM7,OTHRON,SPLPND,STPASK,SRTSMS
                .GLOBL  SIZEMT,ASNOVF,INVLDM,CSIMS4,MNTARG,HUPARG,R50TT
                .GLOBL  KMNNAM,NOKMON,CCLNAM,OTRMNT,CHKDEV,DMTSUB,CMDCCL
                .GLOBL  SHOHD,SUBTXT,MNTTXT,SRTTXT,TOTMMS,UMSSMS
                .GLOBL  TSXSMS,USRMMS,JCXSMS,DZTXT,OCTPRT
                .GLOBL  PRTR50,PRTDAT,PRTTOD,PRTTIM,INVDEV,ALFN,R50DK
                .GLOBL  DETHD,DETARG,RUNMS,NOFRDL,R50MON,INVDAT,MUL32,COAF
                .GLOBL  BADBOT,START,BOTEMT,CF2DEP,LGOVER,R50CHR,REMNDR,PBUFND
                .GLOBL  PPNMSG,CTMSG,CPUMSG,MONTAB,KEYBUF,KEYEND,KMFTXT
                .GLOBL  KMSTK,ASNSRC,INSSRC,SJSPPN,FORCEO
                ;
```

```
                                ;  Assembly constants
                                ;
000012          LF      =       12              ;LINE FEED
000015          CR      =       15              ;CARRIAGE RETURN
000040          BLANK   =       40              ;ASCII SPACE
000007          BELL    =       07              ;ASCII BELL
000011          TAB     =       11              ;HORIZONTAL TAB
000014          FF      =       14              ;FORM FEED
000054          COMMA   =       54              ;COMMA
000400          BLKWDS  =       256.            ;# OF WORDS IN DISK BLOCK
000017          HANCHN  =       17
```

```
                        ;-------------------------------------------------
                        ;  Macro to cause a fatal error message to be printed.
                        ;
                              .MACRO  FERR    MSG
                              MOV     R5,-(SP)
                              MOV     MSG,R5
                              CALL    FPRINT
                              MOV     (SP)+,R5
                              .ENDM   FERR
                        ;
                        ;-------------------------------------------------
                        ;  Macro to print a fatal error message, clean up
                        ;  and then jump to RDCMD.
                        ;
                              .MACRO  FABORT  MSG
                              MOV     MSG,R5
                              JMP     FKILL
                              .ENDM   FABORT
                        ;
                        ;-----------------------------------------------------------------
                        ;  Macro to print a warning message.
                        ;
                              .MACRO  FWARN   MSG
                              MOV     R5,-(SP)
                              MOV     MSG,R5
                              CALL    PRTWRN
                              MOV     (SP)+,R5
                              .ENDM   FWARN
                        ;
                        ;---------------------------------------------------------------
                        ;  Macro to start a standard option table.
                        ;  Name = 1 to 4 character table name.
                        ;  NA = Number of arguments per table entry.
                        ;
                              .MACRO  TBLDEF  NAME,NA
                        NARGS =       NA
                              .CSECT  CMDV3
                        NAME'HD: .WORD  2*NA
                              .ENDM   TBLDEF
                        ;
                        ;---------------------------------------------------------------
                        ;  Macro to enter an option text name and a set of parameters
                        ;  into the currently open table.
                        ;  STRNG = Ascii name
                        ;  A,B,C = Set of option parameters to store in table with name.
                        ;
                              .MACRO  CMDDEF  STRNG,A,B,C,D
                              .CSECT  NAME3
                        L     =       .
                              .ASCIZ  /STRNG/
                              .CSECT  CMDV3
                              .WORD   L               ;POINTER TO NAME STRING
                              .WORD   A
                        .IIF  GE,<NARGS-2>    .WORD   B
                        .IIF  GE,<NARGS-3>    .WORD   C
                        .IIF  GE,<NARGS-4>    .WORD   D
                              .ENDM   CMDDEF
```

```
                        ;
                        ;-------------------------------------------------------------
                        ;  Macro to end a set of table entries.
                        ;
                                .MACRO  TBLEND
                                .CSECT  CMDV3
                                .WORD   0
                                .CSECT  TSKMN3
                                .ENDM   TBLEND
```

```
                        .SBTTL  Privilege names and flags
                ;-----------------------------------------------------------------------
                ;  Table of process privilege names and flags.
                ;
                ;    Arg 1 = Privilege keyword.
                ;    Arg 2 = Name of routine to set or clear flag (PFLRTN).
                ;    Arg 3 = Flag mask.
                ;    Arg 4 = Offset to privilege word with flag.
                ;    Arg 5 = + ==> Set bit, - ==> Clear bit.
                ;
000000                  TBLDEF  PRV,4
000002                  CMDDEF  ALL,PFLALL,0,0,+2
000014                  CMDDEF  ALLO*CATE,PFLRTN,PO$ALC,0,+1
000026                  CMDDEF  NOALLO*CATE,PFLRTN,PO$ALC,0,-1
000040                  CMDDEF  BYP*ASS,PFLRTN,PO$BYP,0,+1
000052                  CMDDEF  NOBYP*ASS,PFLRTN,PO$BYP,0,-1
000064                  CMDDEF  DEB*UG,PFLRTN,PO$DBG,0,+1
000076                  CMDDEF  NODEB*UG,PFLRTN,PO$DBG,0,-1
000110                  CMDDEF  DET*ACH,PFLRTN,PO$DET,0,+1
000122                  CMDDEF  NODET*ACH,PFLRTN,PO$DET,0,-1
000134                  CMDDEF  GETC*XT,PFLRTN,P2$CXT,2,+1
000146                  CMDDEF  NOGETC*XT,PFLRTN,P2$CXT,2,-1
000160                  CMDDEF  MEML*OCK,PFLRTN,PO$LOK,0,+2
000172                  CMDDEF  NOMEML*OCK,PFLRTN,PO$LOK,0,-2
000204                  CMDDEF  MEMM*AP,PFLRTN,PO$MEM,0,+1
000216                  CMDDEF  NOMEMM*AP,PFLRTN,PO$MEM,0,-1
000230                  CMDDEF  MES*SAGE,PFLRTN,P2$MSG,2,+1
000242                  CMDDEF  NOMES*SAGE,PFLRTN,P2$MSG,2,-1
000254                  CMDDEF  NFSR*EAD,PFLRTN,PO$NFR,0,+1
000266                  CMDDEF  NONFSR*EAD,PFLRTN,PO$NFR,0,-1
000300                  CMDDEF  NFSW*RITE,PFLRTN,PO$NFW,0,+1
000312                  CMDDEF  NONFSW*RITE,PFLRTN,PO$NFW,0,-1
000324                  CMDDEF  OP*ER,PFLRTN,PO$OPR,0,+1
000336                  CMDDEF  NOOP*ER,PFLRTN,PO$OPR,0,-1
000350                  CMDDEF  PSWAP*M,PFLRTN,PO$LOK,0,+1
000362                  CMDDEF  NOPSWAP*M,PFLRTN,PO$LOK,0,-1
000374                  CMDDEF  REAL*TIME,PFLRTN,PO$RT,0,+1
000406                  CMDDEF  NOREAL*TIME,PFLRTN,PO$RT,0,-1
000420                  CMDDEF  RLO*CK,PFLRTN,P2$RLK,2,+1
000432                  CMDDEF  NORLO*CK,PFLRTN,P2$RLK,2,-1
000444                  CMDDEF  SEN*D,PFLRTN,PO$SND,0,+1
000456                  CMDDEF  NOSEN*D,PFLRTN,PO$SND,0,-1
000470                  CMDDEF  SETNA*ME,PFLRTN,PO$NAM,0,+1
000502                  CMDDEF  NOSETNA*ME,PFLRTN,PO$NAM,0,-1
000514                  CMDDEF  SETP*RV,PFLRTN,PO$SPV,0,+1
000526                  CMDDEF  NOSETP*RV,PFLRTN,PO$SPV,0,-1
000540                  CMDDEF  SPF*UN,PFLRTN,PO$SPF,0,+1
000552                  CMDDEF  NOSPF*UN,PFLRTN,PO$SPF,0,-1
000564                  CMDDEF  SYSG*BL,PFLRTN,P2$CGR,2,+1
000576                  CMDDEF  NOSYSG*BL,PFLRTN,P2$CGR,2,-1
000610                  CMDDEF  SYSP*RV,PFLRTN,PO$SYS,0,+1
000622                  CMDDEF  NOSYSP*RV,PFLRTN,PO$SYS,0,-1
000634                  CMDDEF  TER*MINAL,PFLRTN,P2$TRM,2,+1
000646                  CMDDEF  NOTER*MINAL,PFLRTN,P2$TRM,2,-1
000660                  CMDDEF  WOR*LD,PFLRTN,P2$WRL,2,+1
000672                  CMDDEF  NOWOR*LD,PFLRTN,P2$WRL,2,-1
000704                  CMDDEF  GRO*UP,PFLRTN,P2$GRP,2,+1
```

Privilege names and flags

```
     58 000716                          CMDDEF  NOGRO*UP, PFLRTN, P2$GRP, 2, -1
     59 000730                          CMDDEF  SAM*E, PFLRTN, P2$SAM, 2, +1
     60 000742                          CMDDEF  NOSAM*E, PFLRTN, P2$SAM, 2, -1
     61 000754                          CMDDEF  SUB*PROCESS, PFLRTN, P2$VIR, 2, +1
     62 000766                          CMDDEF  NOSUB*PROCESS, PFLRTN, P2$VIR, 2, -1
     63 001000                          CMDDEF  VIR*TUAL, PFLRTN, P2$VIR, 2, +2
     64 001012                          CMDDEF  NOVIR*TUAL, PFLRTN, P2$VIR, 2, -2
     65 001024                          CMDDEF  UP1, PFLRTN, P2$UP1, 2, +1
     66 001036                          CMDDEF  NOUP1, PFLRTN, P2$UP1, 2, -1
     67 001050                          CMDDEF  UP2, PFLRTN, P2$UP2, 2, +1
     68 001062                          CMDDEF  NOUP2, PFLRTN, P2$UP2, 2, -1
     69 001074                          CMDDEF  UP3, PFLRTN, P2$UP3, 2, +1
     70 001106                          CMDDEF  NOUP3, PFLRTN, P2$UP3, 2, -1
     71 001120                          CMDDEF  UP4, PFLRTN, P2$UP4, 2, +1
     72 001132                          CMDDEF  NOUP4, PFLRTN, P2$UP4, 2, -1
     73 001144                          CMDDEF  NONE, PFLNON, 0, 0, -2
     74 001156                          CMDDEF  STA*NDARD, PFLSTD, 0, 0, +2
     75 001170                          CMDDEF  STD, PFLSTD, 0, 0, +2
     76 001202                          TBLEND
```

```
                                        .SBTTL  Data areas
                                ;-------------------------------------------------------------------------
                                ;  Data areas
                                ;
                                ;  Table used to convert terminal speeds into speed code values
                                ;
000000  000062                  SPDVAL: .WORD   50.             ;0
000002  000113                          .WORD   75.             ;1
000004  000156                          .WORD   110.            ;2
000006  000206                          .WORD   134.            ;3 (134.5)
000010  000226                          .WORD   150.            ;4
000012  000454                          .WORD   300.            ;5
000014  001130                          .WORD   600.            ;6
000016  002260                          .WORD   1200.           ;7
000020  003410                          .WORD   1800.           ;10
000022  003720                          .WORD   2000.           ;11
000024  004540                          .WORD   2400.           ;12
000026  007020                          .WORD   3600.           ;13
000030  011300                          .WORD   4800.           ;14
000032  016040                          .WORD   7200.           ;15
000034  022600                          .WORD   9600.           ;16
000036  045400                          .WORD   19200.          ;17
                                ;
                                ;  EMT argument block used to move data from kernel to BLKO buffer
                                ;
000040     000      126         PEKEMT: .BYTE   0,126
000042  000010                          .WORD   10              ;Sub function code
000044  000000                  PEKADR: .WORD   0               ;Address of data within kernel
000046  000000                  PEKSIZ: .WORD   0               ;Number of bytes to move
000050  000000G                         .WORD   BLKO            ;Buffer where data is to be stored
                                ;
                                ;  Region Definition Block used to attach to IND PLAS region.
                                ;
000052  000000                  INDRDB: .WORD   0               ;Region ID
000054  000000                          .WORD   0               ;Region size
000056  000000                          .WORD   0               ;Status flags
000060  035164  000000                  .RAD50  /IND   /        ;Name of region
                                ;
                                ;  Words to hold privilege flags
                                ;
000064  000000  000000  000000  PFS0:   .WORD   0,0,0,0
000072  000000
000074  000000  000000  000000  PFC0:   .WORD   0,0,0,0
000102  000000
                                ;
                                ;  Misc data
                                ;
000104  100076                  R50TT0: .RAD50  /TT0/
000106  100105                  R50TT7: .RAD50  /TT7/
000110  012240                  R50CL:  .RAD50  /CL/
000112  012276                  R50CL0: .RAD50  /CL0/
000114  012305                  R50CL7: .RAD50  /CL7/
000116  013630                  R50C1:  .RAD50  /C1/
000120  013666                  R50C10: .RAD50  /C10/
000122  013675                  R50C17: .RAD50  /C17/
000124  075250  014644  000000  BOTHAN: .RAD50  /SY ddd   SYS/  ;SY and ddd replaced during DOSTOP
000132  075273
```

Data areas

```
     55 000134                          INSSPC:  .BLKW   5              ;Program spec beging checked for install
     56                                 ;
     57                                 ;  Byte data
     58                                 ;
     59 000146      000                 NEGFLG:  .BYTE   0              ;Flag to indicate a value should be negated
     60 000147      000                 SJSPPN:  .BYTE   0              ;Flag to indicate PPN display on SHOW JOBS
     61                                          .EVEN
```

```
                                        .SBTTL  ACRPRV -- Accrue a list of privileges
                                ;-----------------------------------------------------------------
                                ;  Accrue a list of privilege keywords of the form:
                                ;  privilege  or  (privilege[,...]).
                                ;
                                ;  Inputs:
                                ;   R3 = Points to start of privilege list.
                                ;
                                ;  Outputs:
                                ;   R3 = Points past end of privilege list.
                                ;   PFS0..PFSn = Privilege flags to be set.
                                ;   PFC0..PFCn = Privilege flags to be cleared.
                                ;
000150  010446                  ACRPRV: MOV     R4,-(SP)
000152  010546                          MOV     R5,-(SP)
                                ;
                                ;  Clear the words which will hold the privilege flags
                                ;
000154  004767  000074                  CALL    CLRPRV          ;Clear all privilege flags
                                ;
                                ;  Skip over leading spaces and see if privilege list is enclosed
                                ;  in parentheses.
                                ;
000160  004767  014636                  CALL    SKPSPC          ;Skip over spaces
000164  005005                          CLR     R5              ;Assume list not enclosed in parens
000166  121327  000050                  CMPB    (R3),#'(        ;Is list enclosed in parentheses?
000172  001002                          BNE     1$              ;Br if not
000174  005203                          INC     R3              ;Skip over parenthesis
000176  005205                          INC     R5              ;Remember parentheses enclose list
                                ;
                                ;  Process the next privilege keyword
                                ;
000200  012704  000000'         1$:     MOV     #PRVHD,R4       ;Point to table of privilege keywords
000204  004767  000476                  CALL    SETWRD          ;Process the next privilege keyword
                                ;
                                ;  See if we have reached the end of the list
                                ;
000210  004767  014606                  CALL    SKPSPC          ;Skip over spaces
000214  005705                          TST     R5              ;Is this a multi-item list?
000216  001413                          BEQ     9$              ;Br if not
000220  112300                          MOVB    (R3)+,R0        ;Get separator character
000222  120027  000051                  CMPB    R0,#')          ;End of the list?
000226  001407                          BEQ     9$              ;Br if yes
000230  120027  000054                  CMPB    R0,#',          ;Comma separator?
000234  001761                          BEQ     1$              ;Br if yes -- Keep going
000236                                  FABORT  #EM$CSE         ;Syntax error
                                ;
                                ;  Finished the privilege list
                                ;
000246  012605                  9$:     MOV     (SP)+,R5
000250  012604                          MOV     (SP)+,R4
000252  000207                          RETURN
```

```
                                        .SBTTL  CLRPRV -- Clear all parsing privilege flags
                                ;-----------------------------------------------------------------------
                                ;  Clear the words used to hold the privilege flags gotten during parsing.
                                ;
                                ;  Outputs:
                                ;   PFS0...PFSn and PFC0..PFCn are set to zero.
                                ;
000254  010446                  CLRPRV: MOV     R4,-(SP)
000256  010546                          MOV     R5,-(SP)
000260  012704  000064'                MOV     #PFS0,R4        ;Words of bits to set
000264  012705  000074'                MOV     #PFC0,R5        ;Words of bits to clear
000270  012700  000000G                MOV     #PVNPW,R0       ;Get # words to clear
000274  005024                  2$:     CLR     (R4)+
000276  005025                          CLR     (R5)+
000300  077003                          SOB     R0,2$
000302  012605                          MOV     (SP)+,R5
000304  012604                          MOV     (SP)+,R4
000306  000207                          RETURN
```

```
                                        .SBTTL  CCSPRV -- Copy current to set privileges
                                ;-------------------------------------------------------------------
                                ;  CCSPRV is used to copy current privileges to set privileges.
                                ;  This is done when running an installed program which is to be locked
                                ;  (RUN/LOCK or installed with the LOCK attribute) so that
                                ;  the privileges with which it was installed are not cleared when
                                ;  aborting possible command files.
                                ;
000310  010046                  CCSPRV: MOV     R0,-(SP)
000312  010446                          MOV     R4,-(SP)
000314  010546                          MOV     R5,-(SP)
000316  012704  000000G                 MOV     #PRIVC0,R4
000322  012705  000000G                 MOV     #PRIVS0,R5
000326  012700  000000G                 MOV     #PVNPW,R0
000332  012425                  1$:     MOV     (R4)+,(R5)+
000334  077002                          SOB     R0,1$
000336  012605                          MOV     (SP)+,R5
000340  012604                          MOV     (SP)+,R4
000342  012600                          MOV     (SP)+,R0
000344  000207                          RETURN
```

```
                        .SBTTL  RSTPRV -- Reset job privileges
                ;-----------------------------------------------------------------------
                ;  RSTPRV is called to reset the current job privileges to those
                ;  privileges for the current command file level.
                ;
000346  010146 RSTPRV: MOV     R1,-(SP)
000350  010446         MOV     R4,-(SP)
000352  010546         MOV     R5,-(SP)
                ;
                ;  If no command file is open now, restore command file privileges
                ;  to set privileges.
                ;
000354  116701 000000G        MOVB    CORUSR,R1       ;Get current job index number
000360  005767 000000G        TST     CFPNT           ;Is a command file open now?
000364  001022                BNE     3$              ;Br if yes
000366  032761 000000G 000000G BIT     #$INDRN,LSW5(R1);Is IND being started?
000374  001016                BNE     3$              ;Br if yes
000376  132767 000000C 000000G BITB    #<IN$ACT!IN$CNT>,INDSTA ;Is IND active?
000404  001012                BNE     3$              ;Br if yes
000406  012704 000000G        MOV     #PRIVS0,R4      ;Point to set privileges
000412  012705 000000G        MOV     #PRIVF0,R5      ;Point to command file privileges
000416  012700 000000G        MOV     #PVNPW,R0       ;Get # words to move
000422  012425         2$:    MOV     (R4)+,(R5)+     ;Copy set privileges to command file priv
000424  077002                SOB     R0,2$
000426  005067 000000G        CLR     AFCF            ;Clear all command file attribute flags
                ;
                ;  Now copy command file privileges to current privileges
                ;
000432  012704 000000G 3$:    MOV     #PRIVF0,R4      ;Point to cells with command file privileges
000436  012705 000000G        MOV     #PRIVC0,R5      ;Point to current priv cells
000442  012700 000000G        MOV     #PVNPW,R0       ;Get # words to move
000446  012425         1$:    MOV     (R4)+,(R5)+     ;Reset privilege flags
000450  077002                SOB     R0,1$
000452  004767 000066         CALL    FIXPRV          ;Transfer privileges to LSW tables
                ;
                ;  Reset program run attribute flags
                ;
000456  016767 000000G 000000G        MOV     AFCF,RUNFLG     ;Reset all program run options
000464  052761 000000G 000000G        BIS     #$SCCA,LSW5(R1) ;Assume control-C abort suppression wanted
000472  032767 000000G 000000G        BIT     #AF$CCA,RUNFLG  ;Does he want to suppress control-C abort?
000500  001003                BNE     4$              ;Br if yes
000502  042761 000000G 000000G        BIC     #$SCCA,LSW5(R1) ;Release SCCA
                ;
                ;  See if we need to reenable process windowing
                ;
000510  042761 000000G 000000G 4$:    BIC     #$NOWIN,LSW11(R1);Assume process windowing is to be enabled
000516  032767 000000G 000000G        BIT     #AF$NPW,RUNFLG  ;Are we suppressing process windowing?
000524  001403                BEQ     9$              ;Br if not
000526  052761 000000G 000000G        BIS     #$NOWIN,LSW11(R1);Suspend process windowing
                ;
                ;  Finished
                ;
000534  012605         9$:    MOV     (SP)+,R5
000536  012604                MOV     (SP)+,R4
000540  012601                MOV     (SP)+,R1
000542  000207                RETURN
```

```
                                        .SBTTL  FIXPRV -- Transfer privilege flags to LSW tables
                                ;-----------------------------------------------------------------
                                ;  FIXPRV is called to transfer privilege flags from the PRIVCO
                                ;  flag cell into the appropriate LSW cells.  It should be called
                                ;  any time a privilege change is made to PRIVCO.
                                ;
000544  010146                  FIXPRV: MOV     R1,-(SP)
000546  010246                          MOV     R2,-(SP)
000550  116701  000000G                MOVB    CORUSR,R1       ;Get current job index number
                                ;
                                ;  Initially reset all privilege flags in LSW tables
                                ;
000554  042761  000000G 000000G        BIC     #$NOVLN,LSW2(R1);Flag that disallows virtual line use
000562  042761  000000G 000000G        BIC     #$NOVLN,LSW2S(R1)
                                ;
                                ;  Now check privilege flags in PRIVC2
                                ;
000570  016702  000000G                MOV     PRIVC2,R2       ;Get current privilege flags
000574  032702  000000G                BIT     #P2$VIR,R2      ;Allow use of virtual lines?
000600  001006                         BNE     1$              ;Br if yes
000602  052761  000000G 000000G        BIS     #$NOVLN,LSW2(R1);Disallow virtual line use
000610  052761  000000G 000000G        BIS     #$NOVLN,LSW2S(R1)
                                ;
                                ;  Finished
                                ;
000616  012602                  1$:     MOV     (SP)+,R2
000620  012601                          MOV     (SP)+,R1
000622  000207                          RETURN
```

```
                                        .SBTTL  OPTLST -- Process list of command options
                                ;-----------------------------------------------------------------------
                                ;  Process a list of command options of the form:
                                ;  /option[=value]...
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of option command string.
                                ;   R4 = Pointer to option processing table.
                                ;
000624                          OPTLST:
                                ;
                                ;  See if there is another option
                                ;
000624  004767  014172          1$:     CALL    SKPSPC          ;Skip over any spaces
000630  121327  000057                  CMPB    (R3),#'/        ;Is there another option?
000634  001004                          BNE     9$              ;Br if not
                                ;
                                ;  Process the next option
                                ;
000636  005203                          INC     R3              ;Skip past /
000640  004767  000042                  CALL    SETWRD          ;Process the option
000644  000767                          BR      1$              ;Go see if there are more options
                                ;
                                ;  Finished
                                ;
000646  000207                  9$:     RETURN
```

```
                        .SBTTL  SCNOPS -- Process a list of command options
                ;-----------------------------------------------------------------------
                ;  SCNOPS processes a list of command qualifiers which may be
                ;  separated by spaces, commas, or slashes.
                ;
                ;  Inputs:
                ;   R3 = Pointer to start of option command string.
                ;   R4 = Pointer to option processing table.
                ;
000650          SCNOPS:
                ;
                ;  Skip over any spaces
                ;
000650  004767  014146          11$:    CALL    SKPSPC          ;Skip over spaces
                ;
                ;  See if we have reached the end of the command
                ;
000654  105713                  TSTB    (R3)            ;Reached end of command?
000656  001412                  BEQ     12$             ;Br if yes
                ;
                ;  Skip over any option separators
                ;
000660  121327  000057          CMPB    (R3),#'/        ;Slash to separate options?
000664  001403                  BEQ     13$             ;Br if yes
000666  121327  000054          CMPB    (R3),#',        ;Comma to separate options?
000672  001001                  BNE     14$             ;Br if not
000674  005203          13$:    INC     R3              ;Skip over delimiter
                ;
                ;  Process an option
                ;
000676  004767  000004          14$:    CALL    SETWRD          ;Process the option
000702  000762                  BR      11$             ;Go back and check for more options
                ;
                ;  Finished
                ;
000704  000207          12$:    RETURN
```

```
                                        .SBTTL  SETWRD -- Process a SET command keyword
                                ;-----------------------------------------------------------------------
                                ;  SETWRD is called to process a keyword associated with a SET command.
                                ;  The appropriate processing subroutine is called for the keyword.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of command keyword.
                                ;   R4 = Pointer to keyword option list.
                                ;
                                ;  Outputs:
                                ;   R3 = Points beyond end of keyword.
                                ;
000706  010446                  SETWRD: MOV     R4,-(SP)
000710  010546                          MOV     R5,-(SP)
000712  010246                          MOV     R2,-(SP)
                                ;
                                ;  If keyword is preceded by "NO", append NO to keyword
                                ;
000714  004767  014102                  CALL    SKPSPC          ;Skip over leading spaces
000720  010302                          MOV     R3,R2           ;Save keyword pointer
000722  004767  004370                  CALL    GTRD50          ;Accrue the next word
000726  026767  000000G 000000G         CMP     R50BUF,R50NO    ;Is this word "NO"?
000734  001005                          BNE     1$              ;Br if not
000736  010305                          MOV     R3,R5           ;Get pointer into command past "NO"
000740  004767  014056                  CALL    SKPSPC          ;Skip up to next word
000744  112325                  4$:     MOVB    (R3)+,(R5)+     ;Concatenate keyword with NO
000746  001376                          BNE     4$              ;Move all of command
000750  010203                  1$:     MOV     R2,R3           ;Get back pointer to keyword
                                ;
                                ;  Look up the option keyword
                                ;
000752  004767  007364                  CALL    SEARCH          ;Look up keyword in table
000756  103405                          BCS     10$             ;Br if invalid keyword
                                ;
                                ;  Call routine to process the option
                                ;
000760  012602                          MOV     (SP)+,R2
000762  004734                          CALL    @(R4)+          ;Call routine to process the keyword
                                ;
                                ;  Finished
                                ;
000764  012605                          MOV     (SP)+,R5
000766  012604                          MOV     (SP)+,R4
000770  000207                          RETURN
                                ;
                                ;  Invalid keyword
                                ;
000772  005704                  10$:    TST     R4              ;Invalid or ambigious keyword?
000774  001404                          BEQ     11$             ;Br if invalid
000776                                  FABORT  #AMBOPT         ;Ambigious option
001006                          11$:    FABORT  #INVOPT         ;Invalid keyword
```

```
                                        .SBTTL  PRVOPT -- Process PRIVILEGE option
                                ;-----------------------------------------------------------------------
                                ;  Process the PRIVILEGE command option which may take the form:
                                ;  PRIVILEGE=privilege  or  PRIVILEGE=(list)
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer past the word "PRIVILEGE".
                                ;
                                ;  Outputs:
                                ;   R3 = Points past end of privilege list.
                                ;   PFS0..PFSn = Privilege flags to set.
                                ;   PFC0..PFCn = Privilege flags to clear.
                                ;
001016                          PRVOPT:
                                ;
                                ;  Equal sign should follow "PRIVILEGE"
                                ;
001016  004767  014142                  CALL    CHKEQ           ;Make sure equal sign follows
                                ;
                                ;  Now process the privilege list
                                ;
001022  004767  177122                  CALL    ACRPRV          ;Accrue the privilege list
                                ;
                                ;  Finished
                                ;
001026  000207                          RETURN
```

```
                                        .SBTTL  PFLRTN -- Set or clear privilege flags
                                ;-----------------------------------------------------------------------
                                ;  PFLRTN is called while parsing a PRIVILEGE list to set or clear
                                ;  privilege flag bits.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to parsing command entry with the following offsets
                                ;        having the values shown:
                                ;        0(R4) = Privilege flag mask word.
                                ;        2(R4) = Offset to privilege word with privilege bit.
                                ;        4(R4) = + ==> Enable privilege, - ==> Disable privilege.
                                ;
001030  010546                  PFLRTN: MOV     R5,-(SP)
                                ;
                                ;  First set the flag in the PFS0 or PFC0 vector
                                ;
001032  012705  000064'                 MOV     #PFS0,R5        ;Assume we are setting privilege
001036  005764  000004                  TST     4(R4)           ;Setting or clearing privilege?
001042  002002                          BGE     1$              ;Br if setting privilege
001044  012705  000074'                 MOV     #PFC0,R5        ;Point to clear-flag words
001050  066405  000002          1$:     ADD     2(R4),R5        ;Point to correct privilege word
001054  051415                          BIS     (R4),(R5)       ;Set correct flag bit
                                ;
                                ;  Now clear the bit in the complementary vector
                                ;
001056  012705  000074'                 MOV     #PFC0,R5        ;Assume we are granting privilege
001062  005764  000004                  TST     4(R4)           ;Are we setting or clearing privilege?
001066  002002                          BGE     2$              ;Br if granting privilege
001070  012705  000064'                 MOV     #PFS0,R5        ;Clearing privilege -- Clear bit in set vector
001074  066405  000002          2$:     ADD     2(R4),R5        ;Point to correct privilege word
001100  041415                          BIC     (R4),(R5)       ;Clear correct flag bit
                                ;
                                ;  Finished
                                ;
001102  012605                          MOV     (SP)+,R5
001104  000207                          RETURN

                                ;-----------------------------------------------------------------------
                                ;  Set standard privileges for a normal job.
                                ;
001106  012767  000000G 176750  PFLSTD: MOV     #P0$$NP,PFS0    ;Grant normal privileges
001114  012767  000000G 176744          MOV     #P2$$NP,PFS0+2
001122  012767  000000C 176744          MOV     #^C<P0$$NP>,PFC0;Remove all other privileges
001130  012767  000000C 176740          MOV     #^C<P2$$NP>,PFC0+2
001136  000207                          RETURN

                                ;-----------------------------------------------------------------------
                                ;  Set all privilege flags.
                                ;
001140  010246                  PFLALL: MOV     R2,-(SP)
001142  012702  000064'                 MOV     #PFS0,R2        ;Point to privilege word
001146  012700  000000G                 MOV     #PVNPW,R0       ;Get # privilege words
001152  012722  177777          1$:     MOV     #177777,(R2)+   ;Set all flags
001156  077003                          SOB     R0,1$
001160  012602                          MOV     (SP)+,R2
001162  000207                          RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Clear all privilege flags
                                ;
001164  010246                 PFLNON: MOV     R2,-(SP)
001166  012702  000074'                 MOV     #PFCO,R2        ;Point to clear-flag vector
001172  012700  000000G                 MOV     #PVNPW,R0       ;Get # privilege words
001176  012722  177777          1$:    MOV     #177777,(R2)+   ;Say all flags are to be cleared
001202  077003                          SOB     R0,1$
001204  012602                          MOV     (SP)+,R2
001206  000207                          RETURN
```

```
                                        .SBTTL  PRVLST -- List names of privileges
                                ;-----------------------------------------------------------------------
                                ;  PRVLST is called to list the names of keywords associated with a certain
                                ;  set of privilege flags.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to vector of privilege words.
                                ;   R3 = +1/-1 to select keyword(+1) or NOkeyword(-1)
                                ;   R4 = Starting column number.
                                ;        (+ ==> Insert leading comma, - ==> No leading comma)
                                ;   R0 = Column number to indent to if we need to wrap around the line.
                                ;
                                ;  Outputs:
                                ;   R4 = Updated column number (positive).
                                ;
001210  010067  000200          PRVLST: MOV     R0,INDCOL       ;Save col # to indent to
001214  010146                          MOV     R1,-(SP)
001216  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if there are any privileges to list
                                ;
001220  010201                          MOV     R2,R1           ;Point to privilege flag vector
001222  012705  000000G                 MOV     #PVNPW,R5       ;Get # words to check
001226  005721                  10$:    TST     (R1)+           ;Any privilege flags set?
001230  001002                          BNE     11$             ;Br if yes
001232  077503                          SOB     R5,10$          ;Check all priv words
001234  000464                          BR      9$              ;There are not privileges to list
                                ;
                                ;  Save column number information
                                ;
001236  010401                  11$:    MOV     R4,R1           ;Get starting column number info
001240  003002                          BGT     5$              ;Br if leading comma wanted
001242  005401                          NEG     R1              ;Get positive column number
001244  005004                          CLR     R4              ;No leading comma wanted
                                ;
                                ;  Initialize pointer to privilege keyword information table
                                ;
001246  012705  000002'         5$:     MOV     #PRVHD+2,R5     ;Point to first entry in table
                                ;
                                ;  See if this entry is selected
                                ;
001252  020365  000010          1$:     CMP     R3,10(R5)       ;Is this the right type of entry?
001256  001046                          BNE     2$              ;Br if not
001260  016500  000006                  MOV     6(R5),R0        ;Get offset to priv word in vector
001264  060200                          ADD     R2,R0           ;Point to correct privilege word
001266  036510  000004                  BIT     4(R5),(R0)      ;Is this privilege flag set?
001272  001440                          BEQ     2$              ;Br if not
                                ;
                                ;  This entry is selected, print its keyword
                                ;
001274  005704                          TST     R4              ;Need leading comma?
001276  001405                          BEQ     3$              ;Br if not
001300                                  .TTYOUT #COMMA          ;Print comma
001310  005201                          INC     R1              ;Count another column
001312  020127  000076          3$:     CMP     R1,#62.         ;Time for a new line?
001316  101414                          BLOS    6$              ;Br if not
001320                                  .PRINT  #CRLF           ;Start a new line
```

```
     58 001326  016704  000062                 MOV     INDCOL,R4       ;Get column to indent to
     59 001332  010401                         MOV     R4,R1           ;Reset column counter
     60 001334  005304                         DEC     R4              ;Get # spaces to print
     61 001336                          7$:    .TTYOUT #BLANK          ;Print a space
     62 001346  077405                         SOB     R4,7$           ;Indent to desired column
     63 001350  011504                  6$:    MOV     (R5),R4         ;Get pointer to keyword string
     64 001352  112400                  4$:    MOVB    (R4)+,R0        ;Get next character of keyword
     65 001354  001407                         BEQ     2$              ;Br if finished
     66 001356  120027  000052                 CMPB    R0,#'*          ;Don't print "*"
     67 001362  001773                         BEQ     4$
     68 001364                                 .TTYOUT                 ;Print a character
     69 001370  005201                         INC     R1              ;Count another column
     70 001372  000767                         BR      4$              ;Go print rest
     71                                 ;
     72                                 ;  Check next entry
     73                                 ;
     74 001374  062705  000012          2$:    ADD     #10.,R5         ;Point to next entry
     75 001400  005715                         TST     (R5)            ;Is there another entry?
     76 001402  001323                         BNE     1$              ;Loop if yes
     77                                 ;
     78                                 ;  Finished
     79                                 ;
     80 001404  010104                         MOV     R1,R4           ;Return updated column # in R4
     81 001406  012605                  9$:    MOV     (SP)+,R5
     82 001410  012601                         MOV     (SP)+,R1
     83 001412  000207                         RETURN
     84 001414  000000                  INDCOL: .WORD   0
```

```
                                        .SBTTL  CKACOJ -- Check if we are privileged to access another job
                                ;-----------------------------------------------------------------------
                                ;  Determine if the current job is privileged to affect the execution
                                ;  of another job.
                                ;
                                ;  Inputs:
                                ;   R2 = Line index number of job we want to affect.
                                ;
                                ;  Outputs:
                                ;   An error message is printed if access is not allowed.
                                ;   C-flag set on return ==> Not allowed to access the job.
                                ;
001416  010146                  CKACOJ: MOV     R1,-(SP)
                                ;
                                ;  Always allow access to our own job
                                ;
001420  120267  000000G                 CMPB    R2,CORUSR       ;Affecting our own job?
001424  001457                          BEQ     7$              ;Br if yes
                                ;
                                ;  Disallow access to detached jobs without DETACH privilege
                                ;
001426  020127  000000G                 CMP     R1,#LSTPL       ;Primary line?
001432  101407                          BLOS    1$              ;Br if so
001434  020127  000000G                 CMP     R1,#LSTDL       ;Detached line?
001440  101004                          BHI     1$              ;Br if secondary
001442  032767  000000G 000000G         BIT     #PO$DET,PRIVCO  ;Do we have DETACH privilege?
001450  001435                          BEQ     6$              ;Br to error return if not
                                ;
                                ;  If we have WORLD privilege we can access any job
                                ;
001452  032767  000000G 000000G 1$:      BIT     #P2$WRL,PRIVC2  ;Do we have WORLD privilege?
001460  001041                          BNE     7$              ;Br if yes
                                ;
                                ;  Always allow access to our virtual lines and children jobs.
                                ;
001462  116701  000000G                 MOVB    CORUSR,R1       ;Get our job index number
001466  126162  000000G 000000G         CMPB    LNPRIM(R1),LNPRIM(R2) ;Do we have the same primary line?
001474  001433                          BEQ     7$              ;Br if yes
001476  026201  000000G                 CMP     LPARNT(R2),R1   ;Are we parent to this job?
001502  001430                          BEQ     7$              ;Br if yes
                                ;
                                ;  See if project numbers of jobs match
                                ;
001504  026162  000000G 000000G 2$:      CMP     LPROJ(R1),LPROJ(R2) ;Do project numbers match?
001512  001014                          BNE     6$              ;Br if not -- We cannot change job
001514  032767  000000G 000000G         BIT     #P2$GRP,PRIVC2  ;Do we have GROUP privilege?
001522  001020                          BNE     7$              ;Br if yes
                                ;
                                ;  Project numbers match, check programmer numbers.
                                ;
001524  026162  000000G 000000G         CMP     LPROG(R1),LPROG(R2) ;Do programmer numbers match?
001532  001004                          BNE     6$              ;Br if not
001534  032767  000000G 000000G         BIT     #P2$SAM,PRIVC2  ;Do we have SAME privilege?
001542  001010                          BNE     7$              ;Br if yes
                                ;
                                ;  We cannot access the job
                                ;
```

```
     58 001544                          6$:     FERR    #EM$CAJ         ;Cannot access that job
     59 001560  000261                          SEC                     ;Signal error on return
     60 001562  000401                          BR      9$
     61                                 ;
     62                                 ;  We can access the job
     63                                 ;
     64 001564  000241                  7$:     CLC                     ;Signal success on return
     65                                 ;
     66                                 ;  Finished
     67                                 ;
     68 001566  012601                  9$:     MOV     (SP)+,R1
     69 001570  000207                          RETURN
```

```
                                        .SBTTL  SPLACT -- Check if spooler is active
                                ;-------------------------------------------------------------------
                                ;  SPLACT is called to determine if the spooling system is currently
                                ;  active or idle.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Spooler active
                                ;   C-flag clear ==> Spooler idle
                                ;
001572  010046                  SPLACT: MOV     R0,-(SP)
001574  012700  000000G                 MOV     #SDCB,R0        ;POINT TO CONTROL BLOCK FOR 1ST SPOOLED DEV
001600  020027  000000G         4$:     CMP     R0,#SDCBND      ;CHECKED ALL?
001604  103010                          BHIS    2$              ;BR IF YES
001606  005760  000000G                 TST     SDFHD(R0)       ;ANY PENDING PRINT FILES?
001612  001003                          BNE     3$              ;BR IF YES
001614  062700  000000G                 ADD     #SDCBSZ,R0      ;POINT TO NEXT DEV CONTROL BLOCK
001620  000767                          BR      4$              ;GO CHECK IT
001622  000261                  3$:     SEC                     ;SPOOLER IS ACTIVE
001624  000401                          BR      9$
001626  000241                  2$:     CLC                     ;SPOOLER IS IDLE
001630  012600                  9$:     MOV     (SP)+,R0
001632  000207                          RETURN
```

```
                                        .SBTTL  CHKTTD -- See if a device name is TT
                                ;-------------------------------------------------------------------------
                                ;  CHKTTD is called to determine if a device name is TT or TTn.
                                ;
                                ;  Inputs:
                                ;   R0 = Rad50 device name.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Device name is TT
                                ;
001634  020067  0000006         CHKTTD: CMP     R0,R50TT        ;Is device TT?
001640  001410                          BEQ     1$              ;Br if yes
001642  020067  176236                  CMP     R0,R50TT0       ;Is it in the range TT0 to TT7?
001646  103403                          BLO     2$              ;Br if not
001650  020067  176232                  CMP     R0,R50TT7
001654  101402                          BLOS    1$
001656  000241                  2$:     CLC                     ;Device is not TT
001660  000401                          BR      9$
001662  000261                  1$:     SEC                     ;Device is TT
001664  000207                  9$:     RETURN
```

```
                                        .SBTTL  DOSTOP -- Stop the system
                                ;-------------------------------------------------------------------------
                                ;  We want to stop the system.
                                ;  Read block 0 of boot device into memory.
                                ;
001666  016703  000000G        DOSTOP: MOV     BOTDEV,R3       ;GET RAD50 NAME OF BOOT DEVICE
001672  001421                        BEQ     2$              ;BR IF NO DEV NAME SPECIFIED -- BOOT FROM SY:
001674  005002                        CLR     R2              ;Clear high order
001676  071227  000050                DIV     #50,R2          ;Get the RAD50 unit number
001702  016702  000000G                MOV     BOTDEV,R2       ;Copy RAD50 name of boot device
001706  160302                        SUB     R3,R2           ;Remove unit number from boot device
001710  010267  176212                MOV     R2,BOTHAN+2     ;Put name in lookup
001714  005067  0000000                CLR     BOTUNI          ;Clear boot unit number
001720  162703  000036                SUB     #36,R3          ;Convert RAD50 unit to numeric
001724  002423                        BLT     4$              ;Br if less than 0
001726  020327  000007                CMP     R3,#7           ;Compare to valid device unit number
001732  003020                        BGT     4$              ;Br if greater than 7
001734  000415                        BR      3$
                                ;
                                ;  Booting from the system device.
                                ;
001736  016702  000000G        2$:     MOV     SYINDX,R2       ;GET DEVICE TABLE INDEX # FOR SY DEVICE
001742  016202  000000G                MOV     PNAME(R2),R2    ;GET RAD50 NAME OF SYSTEM DEVICE
001746  010267  176154                MOV     R2,BOTHAN+2     ;Put name in lookup
001752  116703  0000010                MOVB    SYUNIT+1,R3     ;GET UNIT # OF SYSTEM DISK
001756  060302                        ADD     R3,R2           ;ADD UNIT NUMBER TO BOOT DEVICE NAME
001760  062702  000036                ADD     #36,R2          ;CONVERT TO RAD50 UNIT NUMBER
001764  010267  000000G                MOV     R2,BOTDEV       ;BOOT FROM THIS DEVICE
001770  010367  000000G        3$:     MOV     R3,BOTUNI       ;SAVE BOOT DEVICE UNIT #
                                ;
                                ;  Find the device's CSR address.
                                ;
001774  016703  000000G        4$:     MOV     SYINDX,R3       ;Get device table index # for SY device
002000  116700  000001G                MOVB    SYUNIT+1,R0     ;Get system device unit #
002004  062700  000036                ADD     #36,R0          ;Convert unit # to RAD50
002010  066300  000000G                ADD     PNAME(R3),R0    ;Add in system device name
002014  010067  176104                MOV     R0,BOTHAN       ;Point to system device
002020                                .SERR                   ;Don't abort on lookup errors
002026                                .LOOKUP #XAREA,#1,#BOTHAN ;Try to open handler file
002046  103437                        BCS     99$             ;Error on lookup
                                ;
                                ;  Read block 0 of handler and save information about CSR address
                                ;
002050                                .READW  #XAREA,#1,#BLK0,#256.,#0 ;Read block 0 of handler
002106  103417                        BCS     99$             ;Error on read
002110  016767  000001C 000000G      MOV     BLK0+H.CSR,BOTCSR       ;Save CSR info
002116                                .CLOSE  #1              ;Close channel
                                ;
                                ;  Read the primary and secondary bootstrap.
                                ;
002124                                .LOOKUP #XAREA,#1,#BOTDEV;DO NON-FILE-STRUCTURED LOOKUP ON BOOT DEVICE
002144  103004                        BCC     1$              ;BR IF LOOKUP SUCCESSFUL
002146                          99$:  FABORT  #BADBOT         ;ERROR ON BOOT LOOKUP
                                ;  Read primary driver into memory.
002156                          1$:   .READW  #XAREA,#1,#START,#256.,#0;Read primary bootstrap
002214                                .READW  #XAREA,#1,#START+512.,#1024.,#2;READ SECONDARY BOOTSTRAP
                                ;
```

```
                                ;  Do special Kmon EMT to reboot.
                                ;  (This emt will copy the bootstrap to low memory and enter it)
                                ;
002254  012700  000000G         5$:     MOV     #BOTEMT,R0
002260  104375                         EMT     375             ;REBOOT
```

```
                                        .SBTTL  PUSHCF -- Push a command file
                                ;-------------------------------------------------------
                                ;  PUSHCF IS CALLED TO PUSH THE STATUS OF THE CURRENTLY OPEN
                                ;  COMMAND FILE ON A STACK SO A DEEPER LEVEL FILE CAN BE OPENED.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
002262  010146                  PUSHCF: MOV     R1,-(SP)
002264  116701  000000G                 MOVB    CORUSR,R1       ;GET USER INDEX #
002270  032761  000000G 000000G         BIT     #$CFOPN,LSW4(R1);IS A COMMAND FILE OPEN NOW?
002276  001030                          BNE     6$              ;BR IF YES
                                ;  THERE IS NO COMMAND FILE OPEN NOW
002300  132767  000000G 000000G         BITB    #IN$ACT,INDSTA  ;IS IND ACTIVE NOW?
002306  001403                          BEQ     11$             ;BR IF NOT
002310  116767  000000G 000000G         MOVB    INDSTA,CFIND    ;IF IND IS ACTIVE, SAVE ITS STATUS
002316  005767  000000G         11$:    TST     CFPNT           ;IS A COMMAND FILE IN USE NOW?
002322  001003                          BNE     8$              ;BR IF YES
002324  116767  000000G 000000G         MOVB    INDSTA,CFIND    ;SAVE IND STATUS FLAGS
002332  042761  000000G 000000G 8$:     BIC     #CFLFL4,LSW4(R1);CLEAR MISC COMMAND FILE FLAGS
002340  032761  000000G 000000G         BIT     #$QTSET,LSW2(R1);DOES HE WANT QUIET OR NOQUIET?
002346  001524                          BEQ     9$              ;BR IF NOQUIET WANTED
002350  052761  000000G 000000G         BIS     #$QUIET,LSW4(R1);SET QUIET
002356  000520                          BR      9$
                                ;  THERE IS AN OPEN COMMAND FILE WHICH NEEDS TO BE PUSHED
002360  010346                  6$:     MOV     R3,-(SP)
002362  010446                          MOV     R4,-(SP)
002364  010546                          MOV     R5,-(SP)
002366  016705  000000G                 MOV     CFSP,R5         ;GET SAVE STACK POINTER
002372  020527  000000C                 CMP     R5,#<CFSEND+20.+<2*PVNPW>>;ROOM ENOUGH TO START PUSH?
002376  103535                          BLO     CFOVFL          ;BR IF STACK OVERFLOW WOULD OCCUR
                                ;  DO .SAVESTATUS TO SAVE FILE NAME
002400  162705  000012                  SUB     #10.,R5         ;NEED 5 WORDS FOR .SAVEST
002404                                  .SAVEST #XAREA,#CFCHAN,R5
                                ;  NOW SAVE BUFFER POINTERS
002422  016745  000000G                 MOV     CFBLK,-(R5)     ;CURRENT FILE BLOCK #
002426  016745  000000G                 MOV     CFPNT,-(R5)     ;CURRENT POINTER INTO BUFFER
                                ;  Save command file privileges
002432  012703  000000C                 MOV     #PRIVFO+<2*PVNPW>,R3 ;Point past last privilege word
002436  014345                  12$:    MOV     -(R3),-(R5)     ;Push each privilege word
002440  020327  000000G                 CMP     R3,#PRIVFO      ;Pushed all yet?
002444  101374                          BHI     12$             ;Br if more to push
                                ;  Save command file attribute flags
002446  016745  000000G                 MOV     AFCF,-(R5)      ;Push attribute flags
                                ;  SAVE QUIET STATUS
002452  016145  000000G                 MOV     LSW4(R1),-(R5)  ;SAVE QUIET FLAG
                                ;  Save IND status
002456  016745  000000G                 MOV     CFIND,-(R5)     ;PUSH IND STATUS FLAGS
002462  116767  000000G 000000G         MOVB    INDSTA,CFIND    ;SET NEW IND STATUS FOR COMMAND FILE
                                ;  NOW SAVE INFO ABOUT PARAMETERS
002470  016745  000000G                 MOV     CURPRM,-(R5)    ;CURRENT PARAMETER POINTER
002474  016704  000000G                 MOV     PBFEND,R4       ;ADDR OF END OF PARAMETER STRING
002500  005204                          INC     R4              ;ROUND UP TO NEXT WORD
002502  042704  000001                  BIC     #1,R4
002506  012703  000000G                 MOV     #PRMBUF,R3      ;POINT TO START OF PARAM STRING
002512  020304                  2$:     CMP     R3,R4           ;PUSHED ALL ON STACK?
002514  103005                          BHIS    1$              ;BR IF YES
002516  020527  000002G                 CMP     R5,#<CFSEND+2.> ;STACK OVERFLOW?
002522  101463                          BLOS    CFOVFL          ;BR IF OVERFLOW
```

```
                                        MOV     (R3)+,-(R5)     ;PUSH PARAMETER STRING
                                        BR      2$
                                1$:     MOV     R4,-(R5)        ;PUSH POINTER TO END OF STRING
                                ;  PUSH PARAMETER POINTERS
                                        CLR     -(R5)           ;PUSH ZERO TO MARK END
                                        MOV     #LSTPRM,R3      ;POINT TO LAST PARAM PTR CELL
                                4$:     TST     -(R3)           ;IS POINTER IN USE?
                                        BNE     3$              ;BR IF YES
                                        CMP     R3,#PRMPNT      ;CHECKED ALL?
                                        BHI     4$              ;BR IF NOT
                                        BR      5$              ;BR IF NO PARAMETERS DEFINED
                                3$:     CMP     R5,#CFSEND      ;ROOM TO PUSH INFO?
                                        BLOS    CFOVFL          ;BR IF STACK OVERFLOW
                                        MOV     (R3),-(R5)      ;PUSH PARAMETER POINTER
                                        TST     -(R3)           ;POINT TO NEXT PARAM POINTER CELL
                                        CMP     R3,#PRMPNT      ;MORE TO PUSH?
                                        BHIS    3$              ;BR IF YES
                                ;  WE HAVE SAVED ALL INFORMATION NEEDED TO RESTART INDIRECT FILE
                                5$:     MOV     R5,CFSP         ;SAVE STACK POINTER
                                        BIC     #$CFOPN,LSW4(R1);SAY NO FILE IS OPEN NOW
                                        INCB    CFNEST          ;SAY WE ARE NESTED DEEPTER
                                        MOV     (SP)+,R5
                                        MOV     (SP)+,R4
                                        MOV     (SP)+,R3
                                ;  SET UP POINTERS FOR NEW COMMAND FILE
                                9$:     MOV     #CFBUF,R0       ;SET POINTER TO COMMAND BUFFER
                                        CALL    CFSTRT          ;INIT POINTER INTO BUFFER
                                        CLR     CFBLK           ;RESET FILE BLOCK # TO ZERO
                                        CLR     CURPRM          ;CLEAR PARAM STRING POINTER
                                        MOV     #PRMPNT,R1      ;CLEAR ALL PARAM POINTERS
                                7$:     CLR     (R1)+
                                        CMP     R1,#LSTPRM
                                        BLO     7$
                                        MOV     #PRMBUF,PBFEND  ;SAY NO PARAM STRING YET
                                ;  Say that IND is not active now
                                        CLRB    INDSTA          ;SAY IND IS NOT ACTIVE NOW
                                        MOV     (SP)+,R1
                                        RETURN
                                ;
                                ;  ERROR -- OVERFLOW OF FILE SAVE STACK
                                CFOVFL: FABORT  #CF2DEP
```

```
                                        .SBTTL  POPCF  -- Pop a command file
                                ;-------------------------------------------------------------
                                ;  POPCF IS CALLED TO CLOSE THE CURRENTLY OPEN INDIRECT COMMAND
                                ;  FILE AND REOPEN THE ONE WHICH IS NEXT HIGHER IN THE STACK.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
002702  010146                  POPCF:  MOV     R1,-(SP)
002704  116701  000000G                 MOVB    CORUSR,R1       ;GET USER INDEX NUMBER
002710  042761  000000G 000000G         BIC     #$CFCCL,LSW4(R1);SAY WE HAVE FINISHED ANY CCL COMMAND
002716  105767  000000G                 TSTB    CFNEST          ;Any nested command files?
002722  001003                          BNE     10$             ;Br if yes
002724  005767  000000G                 TST     CFPNT           ;ANY INDIRECT FILES IN USE?
002730  001563                          BEQ     9$              ;BR IF NOT
002732  116767  000000G 000000G 10$:    MOVB    CFIND,INDSTA    ;RESTORE IND STATUS FLAGS
002740  105067  000000G                 CLRB    CFHOLD          ;CLEAR ANY COMMAND FILE HOLDING CHAR
002744  032761  000000G 000000G         BIT     #$CFOPN,LSW4(R1);IS @FILE CHANNEL OPEN NOW?
002752  001403                          BEQ     1$              ;BR IF NOT
                                ;  CLOSE CURRENTLY OPEN FILE
002754                                  .CLOSE  #CFCHAN         ;CLOSE THE FILE
                                ;  SEE IF THERE IS A HIGHER LEVEL FILE TO RESTORE
002762  105767  000000G          1$:     TSTB    CFNEST          ;ANY FILES ON STACK NOW?
002766  001016                          BNE     2$              ;BR IF YES
                                ;  THERE ARE NO HIGHER LEVEL FILES
002770  042761  000000C 000000G         BIC     #<$CFOPN!$CFALL!$CFSOT>,LSW4(R1);CLEAR COMMAND FILE FLAGS
002776  042761  000000G 000000G         BIC     #$NOIN,LSW3(R1) ;Allow input to be accepted for line
003004  042761  000000G 000000G         BIC     #$SUCF,LSW9(R1) ;Say start-up command file finished
003012  004767  000570                  CALL    CFSTOP          ;Suspend command file input
003016  004767  175324                  CALL    RSTPRV          ;Reset command file privileges
003022  000526                          BR      9$
                                ;  REOPEN NEXT HIGHER LEVEL FILE
                                ;  RESTORE PARAMETER POINTERS
003024  010346                  2$:     MOV     R3,-(SP)
003026  010446                          MOV     R4,-(SP)
003030  010546                          MOV     R5,-(SP)
003032  016705  000000G                 MOV     CFSP,R5         ;GET STACK POINTER
003036  012703  000000G                 MOV     #PRMPNT,R3      ;POINT TO PARAM POINTER CELLS
003042  012504                  4$:     MOV     (R5)+,R4        ;GET A PARAMETER POINTER
003044  001402                          BEQ     3$              ;BR IF END OF LIST HIT
003046  010423                          MOV     R4,(R3)+        ;RESTORE POINTER
003050  000774                          BR      4$
003052  020327  000000G         3$:     CMP     R3,#LSTPRM      ;ZERO ALL OTHER PARAM POINTERS
003056  103002                          BHIS    5$
003060  005023                          CLR     (R3)+
003062  000773                          BR      3$
                                ;  RESTORE PARAMETER STRING
003064  012504                  5$:     MOV     (R5)+,R4        ;GET ADDRESS OF END OF STRING
003066  010467  000000G                 MOV     R4,PBFEND
003072  020427  000000G         7$:     CMP     R4,#PRMBUF      ;RESTORED ALL OF PARAM STRING?
003076  101402                          BLOS    6$              ;BR IF YES
003100  012544                          MOV     (R5)+,-(R4)     ;POP STRING OFF STACK
003102  000773                          BR      7$
003104  012567  000000G         6$:     MOV     (R5)+,CURPRM    ;POP POINTER INTO STRING
                                ;  Restore IND status flags
003110  012567  000000G                 MOV     (R5)+,CFIND     ;RESTORE IND STATUS FLAGS
                                ;  RESET COMMAND FILE CONTROL FLAGS
003114  012704  000000G                 MOV     #CFLFL4,R4      ;GET MIST CONTROL FLAGS
003120  040461  000000G                 BIC     R4,LSW4(R1)     ;CLEAR THOSE FLAGS
```

```
                        COM     R4                      ;MASK ALL BUT THOSE FLAGS
                        BIC     R4,(R5)
                        BIS     (R5)+,LSW4(R1)  ;SET DESIRED COMBINATION
                ;  Restore command file attribute flags
                        MOV     (R5)+,AFCF      ;Restore attribute flags
                ;  Restore command file privilege flags
                        MOV     #PRIVFO,R0      ;Point to privilege vector
                11$:    MOV     (R5)+,(R0)+     ;Pop a privilege flag
                        CMP     R0,#PRIVFO+<2*PVNPW>;Popped all?
                        BLO     11$             ;Loop if not
                        CALL    RSTPRV          ;Reset some flags
                ;  RESTORE BUFFER POINTER INFORMATION
                8$:     MOV     (R5)+,R0        ;POINTER INTO BUFFER
                        CALL    CFSTRT          ;Set command file buffer pointer
                        MOV     (R5)+,CFBLK     ;CURRENT FILE BLOCK NUMBER
                ;  NOW REOPEN THE INDIRECT COMMAND FILE
                        .REOPEN #XAREA,#CFCHAN,R5
                        ADD     #10.,R5         ;POP FILE NAME INFO OFF STACK
                ;  REREAD CURRENT BUFFER
                        .READW  #XAREA,#CFCHAN,#CFBUF,#256.,CFBLK
                ;  FINISHED RESTORING FILE
                        MOV     R5,CFSP         ;SAVE UPDATED STACK POINTER
                        DECB    CFNEST          ;SAY ONE LESS FILE ON STACK
                        BIS     #$CFOPN,LSW4(R1);SAY CHANNEL IS OPEN
                        MOV     (SP)+,R5
                        MOV     (SP)+,R4
                        MOV     (SP)+,R3
                9$:     MOV     (SP)+,R1
                        RETURN
```

```
                        .SBTTL  ABRTCF -- Abort all command files
                ;-----------------------------------------------------------------------
                ;  ABRTCF is called to close all open indirect command files
                ;  including those on the stack.
                ;  If IND is active, only command files under IND are aborted.
                ;  All registers are preserved.
                ;
003304  010146          ABRTCF: MOV     R1,-(SP)
003306  116701  000000G          MOVB    CORUSR,R1       ;GET USER INDEX #
003312  016700  000000G          MOV     CFSPND,R0       ;Is there a suspended command file?
003316  001404                  BEQ     2$              ;Br if not
003320  004767  000306          CALL    CFSTRT          ;Restart command file input
003324  005067  000000G          CLR     CFSPND          ;No more suspended command file
                ;
                ;  Abort all nested command files
                ;
003330  105767  000000G  2$:     TSTB    CFNEST          ;Any nested command files?
003334  001003                  BNE     3$              ;Br if yes
003336  005767  000000G          TST     CFPNT           ;IS AN INDIRECT FILE OPEN?
003342  001407                  BEQ     4$              ;BR IF NOT
003344  132767  000000G 000000G 3$:     BITB    #IN$ACT,INDSTA  ;Have we reached the level of IND?
003352  001003                  BNE     4$              ;Br if yes -- Leave IND in control
003354  004767  177322          CALL    POPCF           ;CLOSE IT
003360  000763                  BR      2$
                ;
                ;  Reset misc command file related values
                ;
003362  042761  000000G 000000G 4$:     BIC     #$CFABT,LSW6(R1);SAY ALL COMMAND FILES HAVE BEEN ABORTED
003370  042761  000000G 000000G        BIC     #$NTGCC,LSW9(R1);Clear saved ctrl-C flag
003376  042761  000000G 000000G        BIC     #$CFDCC,LSW4(R1)
003404  105067  000000G          CLRB    UERSEV          ;CLEAR USER ERROR FLAG
                ;
                ;  Reset command file privileges from set privileges
                ;
003410  004767  174732          CALL    RSTPRV          ;Reset current privileges
                ;
                ;  Finished
                ;
003414  012601                  MOV     (SP)+,R1
003416  000207                  RETURN
```

```
                                        .SBTTL  INDABT -- Abort execution of IND and nested command files
                                ;-----------------------------------------------------------------------
                                ;  INDABT is called to abort the execution of IND and of any nested
                                ;  command files.
                                ;
003420  010146                  INDABT: MOV     R1,-(SP)
003422  116701  000000G               MOVB    CORUSR,R1       ;Get user index #
                                ;
                                ;  Close command files
                                ;
003426  016700  000000G               MOV     CFSPND,R0       ;Is there a suspended command file?
003432  001404                        BEQ     2$              ;Br if not
003434  004767  000172                CALL    CFSTRT          ;Restart suspended command file
003440  005067  000000G               CLR     CFSPND          ;No more suspended command file
003444  005767  000000G         2$:   TST     CFPNT           ;Is an indirect file open?
003450  001403                        BEQ     1$              ;Br if not
003452  004767  177224                CALL    POPCF           ;Close it
003456  000772                        BR      2$
003460  042761  000000C 000000G 1$:   BIC     #$CFKIL!$CFABT,LSW6(R1);Say command files have been aborted
003466  105067  000000G               CLRB    UERSEV          ;Clear user error flag
                                ;
                                ;  Stop IND
                                ;
003472  105067  000000G               CLRB    INDSTA          ;Say IND is finished
003476  042761  000000G 000000G       BIC     #$INDRN,LSW5(R1);Say IND is not running now
003504  042761  000000G 000000G       BIC     #$NTGCC,LSW9(R1);Clear saved ctrl-C flag
003512  042761  000000G 000000G       BIC     #$CFDCC,LSW4(R1)
                                ;
                                ;  Reset privileges
                                ;
003520  004767  174622                CALL    RSTPRV          ;Reset privileges
                                ;
                                ;  Eliminate IND global PLAS region
                                ;
003524  012767  000000C 000001C       MOV     #<RS.GBL!RS.PVT>,INDRDB+R.GSTS ;Set status flags in RDB
003532                                .CRRG   #XAREA,#INDRDB  ;Try to attach to IND region
003552  103413                        BCS     3$              ;Br if cannot attach
003554  012767  000000G 000001C       MOV     #RS.EGR,INDRDB+R.GSTS ;Set eliminate-global-region flag
003562                                .ELRG   #XAREA,#INDRDB  ;Eliminate the region
                                ;
                                ;  Finished
                                ;
003602  012601                  3$:   MOV     (SP)+,R1
003604  000207                        RETURN
```

```
                        .SBTTL  CFSTOP -- Stop input from a command file
                ;-----------------------------------------------------------------------
                ;  Zero CFPNT to say input is not coming from a command file.
                ;
003606  010046  CFSTOP: MOV     R0,-(SP)
                ;
                ;  Say input not coming from a command file
                ;
003610  005067  000000G         CLR     CFPNT           ;Clear command file buffer pointer
                ;
                ;  Reset status flags in RMON cell
                ;
003614  016700  000000G         MOV     CXTRMN,R0       ;Get virtual address of RMON
003620  042760  000000G 000000G BIC     #CFACFL,R$CFST(R0) ;Say input not from CF
                ;
                ;  Finished
                ;
003626  012600                  MOV     (SP)+,R0
003630  000207                  RETURN

                        .SBTTL  CFSTRT -- Start input from a command file
                ;-----------------------------------------------------------------------
                ;  Store a buffer pointer into CFPNT to start command file input.
                ;
                ;  Inputs:
                ;   R0 = Value to be stored into CFPNT
                ;
003632  010046  CFSTRT: MOV     R0,-(SP)
                ;
                ;  Store buffer pointer into CFPNT
                ;
003634  010067  000000G         MOV     R0,CFPNT        ;Set CF buffer pointer
003640  001405                  BEQ     9$              ;Br if not starting command file
                ;
                ;  Set status flags in RMON cell
                ;
003642  016700  000000G         MOV     CXTRMN,R0       ;Get RMON address
003646  052760  000000G 000000G BIS     #CFACFL,R$CFST(R0);Set command-file-active flags
                ;
                ;  Finished
                ;
003654  012600  9$:     MOV     (SP)+,R0
003656  000207                  RETURN
```

```
                        .SBTTL  CFSQEZ -- Squeeze space in command file buffer
                ;-----------------------------------------------------------------------
                ;  CFSQEZ is called to squeeze all remaining commands in the current
                ;  command file buffer to the top of the buffer.
                ;  Nulls are removed and all pending commands are moved up against
                ;  the top of the buffer.
                ;
                ;  Inputs:
                ;   CFPNT:  0==>Command file not active.
                ;           Non-zero==>Points to start of pending commands in buffer.
                ;
                ;  Outputs:
                ;   CFPNT: Points to start of pending commands in buffer.
                ;   R0 = Top of free area in buffer.
                ;
003660  010246  CFSQEZ: MOV     R2,-(SP)
003662  010346          MOV     R3,-(SP)
003664  010446          MOV     R4,-(SP)
                ;
                ;  Determine if there are any pending commands in the command file buffer
                ;
003666  012703  001000G         MOV     #CFBUF+512.,R3  ;POINT PAST TOP OF BUFFER
003672  016704  000000G         MOV     CFPNT,R4        ;GET POINTER TO PENDING COMMANDS
003676  001411                  BEQ     4$              ;BR IF THERE ARE NO PENDING COMMANDS
                ;
                ;  There are pending commands in the buffer.
                ;  Move them to the top of the buffer.
                ;
003700  010302                  MOV     R3,R2           ;GET POINTER TO TOP OF BUFFER
003702  020204  1$:     CMP     R2,R4           ;HAVE WE MOVED ALL PENDING COMMANDS?
003704  101404          BLOS    6$              ;BR IF YES
003706  114200          MOVB    -(R2),R0        ;GET NEXT CHAR
003710  001774          BEQ     1$              ;AND SKIP NULLS
003712  110043          MOVB    R0,-(R3)        ;PACK INTO TOP OF BUFFER
003714  000772          BR      1$
003716  010367  000000G 6$:     MOV     R3,CFPNT        ;SAVE NEW COMMAND FILE POINTER
                ;
                ;  Null fill buffer from base up to start of pending commands
                ;
003722  010300  4$:     MOV     R3,R0           ;SAVE POINTER PAST TOP OF FREE AREA
003724  012704  000000G         MOV     #CFBUF,R4       ;POINT TO BASE OF BUFFER
003730  020304  3$:     CMP     R3,R4           ;HAVE WE FILLED TO BASE?
003732  101402          BLOS    5$              ;BR IF YES
003734  105043          CLRB    -(R3)           ;NULL FILL BUFFER
003736  000774          BR      3$
                ;
                ;  Finished
                ;
003740  012604  5$:     MOV     (SP)+,R4
003742  012603          MOV     (SP)+,R3
003744  012602          MOV     (SP)+,R2
003746  000207          RETURN
```

```
                        .SBTTL  LOGCHK -- Check to see if log file is on specified dev
                ;-----------------------------------------------------------------------
                ;  LOGCHK is called to determine if the log file is open to a specified
                ;  device or to a subdevice contained within the specified device.
                ;  If the log file is on the specified device, print a warning message
                ;  and close the log file.
                ;
                ;  Inputs:
                ;   R5 = Rad50 name of device.
                ;
003750  010246  LOGCHK: MOV     R2,-(SP)
003752  010346         MOV     R3,-(SP)
003754  010446         MOV     R4,-(SP)
003756  010546         MOV     R5,-(SP)
                ;
                ;  See if the log file is currently open
                ;
003760  032767  000000G 000000G      BIT     #LF$OPN,LOGFLG  ;Is the log file currently open?
003766  001445         BEQ     9$              ;Br if not
                ;
                ;  The log file is open.
                ;  Convert device name into device # and unit #
                ;
003770  004767  005634         CALL    CHKDEV          ;Cvt name to dev # and unit #
003774  103442         BCS     9$              ;Br if don't recognize name
                ;
                ;  At this point, R0 = unit number, R4 = device index number.
                ;  Determine if this is a physical or logical disk.
                ;
003776  020467  000000G      CMP     R4,LDDEVX       ;Is this a logical disk?
004002  001406         BEQ     1$              ;Br if yes
                ;
                ;  This is a physical disk
                ;
004004  000300         SWAB    R0              ;Get unit # to high-order byte
004006  050004         BIS     R0,R4           ;Combine device and unit #
004010  020467  000000G      CMP     R4,LOGDVU       ;Same device as log file?
004014  001032         BNE     9$              ;Br if not
004016  000421         BR      8$              ;Br if yes
                ;
                ;  This is a logical disk
                ;
004020  006300         1$:     ASL     R0              ;Convert unit # to word table index
004022  016004  000000G      MOV     LDPDEV(R0),R4   ;Get physical dev # and unit #
004026  016002  000000G      MOV     LDBASE(R0),R2   ;Get base block # of logical disk
004032  010203         MOV     R2,R3
004034  066003  000000G      ADD     LDSIZE(R0),R3   ;Get top block # of logical disk
004040  020467  000000G      CMP     R4,LOGDVU       ;Is log file on same physical disk?
004044  001016         BNE     9$              ;Br if not
004046  026702  000000G      CMP     LOGBAS,R2       ;Is log file on log disk within this disk?
004052  103413         BLO     9$              ;Br if not
004054  026703  000000G      CMP     LOGBAS,R3
004060  103010         BHIS    9$
                ;
                ;  Log file is on the specified device
                ;  Pring a warning message and close the log file.
                ;
```

```
     58 004062                                  8$:      FWARN   #TM$CLG          ;Say we are closing the log file
     59 004076  004767  000012                          CALL    LOGCLS           ;Close the log file
     60                                         ;
     61                                         ;  Finished
     62                                         ;
     63 004102  012605                          9$:      MOV     (SP)+,R5
     64 004104  012604                                   MOV     (SP)+,R4
     65 004106  012603                                   MOV     (SP)+,R3
     66 004110  012602                                   MOV     (SP)+,R2
     67 004112  000207                                   RETURN
```

```
                                        .SBTTL  LOGCLS -- Close the log file
                                ;-----------------------------------------------------------------
                                ;  LOGCLS is called to close the log file for the current job.
                                ;
004114  010246                  LOGCLS: MOV     R2,-(SP)
004116  032767  000000G 000000G        BIT     #LF$OPN,LOGFLG  ;Is the log file open?
004124  001451                        BEQ     9$              ;Br if not
                                ;
                                ;  Log file is open, write last buffer to file
                                ;
004126  016702  000000G               MOV     LOGPTR,R2       ;Get buffer pointer
004132  001436                        BEQ     2$              ;Br if file overflow occurred
004134  032702  000001                BIT     #1,R2           ;Do we need to put in a null at end?
004140  001401                        BEQ     1$              ;Br if not
004142  105022                        CLRB    (R2)+           ;Put null to fill out last word
004144  162702  000000G         1$:   SUB     #LOGBUF,R2      ;Get # bytes in log buffer
004150  001427                        BEQ     2$              ;Br if buffer is empty
004152  006202                        ASR     R2              ;Get # words in buffer
004154                                .WRITW  #XAREA,#LOGCHN,#LOGBUF,R2,LOGBLK ;Write block to log file
004212  103006                        BCC     2$              ;Br if write ok
004214                                FERR    #LGOVER         ;Log file overflow
                                ;
                                ;  Close the log file
                                ;
004230  005067  000000G         2$:   CLR     LOGPTR          ;Say we are no longer logging
004234  042767  000000G 000000G        BIC     #LF$OPN,LOGFLG  ;Say log file is closed
004242                                .CLOSE  #LOGCHN         ;Close log file channel
                                ;
                                ;  Finished
                                ;
004250  012602                  9$:   MOV     (SP)+,R2
004252  000207                        RETURN
```

```
                                        .SBTTL  ACRDEC -- Accrue a decimal value
                                ;-----------------------------------------------------------------
                                ;  ACRDEC is called to accrue a decimal number.
                                ;  Leading spaces and an optional leading equal sign are skipped.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of number.
                                ;
                                ;  Outputs:
                                ;   R0 = Delimiter hit at end of number.
                                ;   R1 = Accrued value.
                                ;   R3 = Pointer to delimiter at end of number.
                                ;
004254  010446                  ACRDEC: MOV     R4,-(SP)
                                ;
                                ;  Skip leading spaces and an optional leading equal sign
                                ;
004256  004767  010540                  CALL    SKPSPC          ;Skip over leading spaces
004262  121327  000075                  CMPB    (R3),#'=        ;Is there an equal sign?
004266  001003                          BNE     3$              ;Br if not
004270  005203                          INC     R3              ;Skip past the equal sign
004272  004767  010524                  CALL    SKPSPC          ;Skip spaces following equal sign
                                ;
                                ;  See if number is preceded by a minus sign
                                ;
004276  105067  173644          3$:     CLRB    NEGFLG          ;Assume number should not be negated
004302  121327  000055                  CMPB    (R3),#'-        ;Is there a leading minus sign?
004306  001005                          BNE     5$              ;Br if not
004310  105267  173632                  INCB    NEGFLG          ;Remember to negate value
004314  005203                          INC     R3              ;Skip past minus sign
004316  004767  010500                  CALL    SKPSPC          ;Skip over spaces
                                ;
                                ;  Make sure first character of number is a decimal digit
                                ;
004322  111300                  5$:     MOVB    (R3),R0         ;Get first character of number
004324  120027  000060                  CMPB    R0,#'0          ;Is 1st character a digit?
004330  103432                          BLO     4$              ;Br if not
004332  120027  000071                  CMPB    R0,#'9
004336  101027                          BHI     4$              ;Br if not
                                ;
                                ;  Accrue the number
                                ;
004340  005001                          CLR     R1              ;ACCRUE VALUE IN R1
004342  112300                  2$:     MOVB    (R3)+,R0        ;GET NEXT CHARACTER
004344  120027  000056                  CMPB    R0,#'.          ;Decimal pointer delimiter?
004350  001414                          BEQ     6$              ;Br if yes
004352  010004                          MOV     R0,R4
004354  162704  000060                  SUB     #'0,R4          ;CONVERT DIGIT TO BINARY VALUE
004360  100407                          BMI     1$              ;BRANCH IF CHAR NOT A DIGIT
004362  020427  000011                  CMP     R4,#9.          ;IS IT A DIGIT?
004366  003004                          BGT     1$              ;BRANCH IF NOT
                                ;
                                ;  Multiply previous value by 10 and add new digit.
                                ;
004370  070127  000012                  MUL     #10.,R1         ;MULTIPLY BY 10.
004374  060401                          ADD     R4,R1           ;ADD IN NEW DIGIT VALUE
004376  000761                          BR      2$
```

```
                                        ;
                                        ;  Hit delimiter.
                                        ;
004400  005303                  1$:     DEC     R3              ;POINT TO DELIMITER
004402  105767  173540          6$:     TSTB    NEGFLG          ;Should we negate the value
004406  001401                          BEQ     9$              ;Br if not
004410  005401                          NEG     R1              ;Negate the value
004412  012604                  9$:     MOV     (SP)+,R4
004414  000207                          RETURN
                                        ;
                                        ;  Error -- Expected number does not start with a digit
                                        ;
004416                          4$:     FABORT  #EM$ENM         ;Expected number is missing
```

```
                                        .SBTTL  ACROCT -- Accrue an octal value
                                ;-------------------------------------------------
                                ;  ACROCT is called to accrue an octal value.
                                ;  Leading spaces and an optional leading equal sign are skipped.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of number.
                                ;
                                ;  Outputs:
                                ;   R0 = Delimiter hit at end of number.
                                ;   R1 = Accrued value.
                                ;   R3 = Pointer to delimiter at end of number.
                                ;
004426  010446                  ACROCT: MOV     R4,-(SP)
                                ;
                                ;  Skip leading spaces and an optional leading equal sign
                                ;
004430  004767  010366                  CALL    SKPSPC          ;Skip over leading spaces
004434  121327  000075                  CMPB    (R3),#'=        ;Is there an equal sign?
004440  001003                          BNE     3$              ;Br if not
004442  005203                          INC     R3              ;Skip past the equal sign
004444  004767  010352                  CALL    SKPSPC          ;Skip spaces following equal sign
                                ;
                                ;  See if number is preceded by a minus sign
                                ;
004450  105067  173472          3$:     CLRB    NEGFLG          ;Assume number should not be negated
004454  121327  000055                  CMPB    (R3),#'-        ;Is there a leading minus sign?
004460  001005                          BNE     5$              ;Br if not
004462  105267  173460                  INCB    NEGFLG          ;Remember to negate value
004466  005203                          INC     R3              ;Skip past minus sign
004470  004767  010326                  CALL    SKPSPC          ;Skip over spaces
                                ;
                                ;  Make sure first character of number is an octal digit
                                ;
004474  111300                  5$:     MOVB    (R3),R0         ;Get first character of number
004476  120027  000060                  CMPB    R0,#'0          ;Is 1st character a digit?
004502  103432                          BLO     4$              ;Br if not
004504  120027  000067                  CMPB    R0,#'7
004510  101027                          BHI     4$              ;Br if not
                                ;
                                ;  Accrue the number
                                ;
004512  005001                          CLR     R1              ;ACCRUE VALUE IN R1
004514  112300                  2$:     MOVB    (R3)+,R0        ;GET NEXT CHAR
004516  010004                          MOV     R0,R4
004520  162704  000060                  SUB     #'0,R4          ;CONVERT ASCII CHAR TO VALUE
004524  100407                          BMI     1$              ;BRANCH IF NOT DIGIT
004526  020427  000007                  CMP     R4,#7           ;LEGAL OCTAL DIGIT?
004532  003004                          BGT     1$              ;BRANCH IF NOT
004534  072127  000003                  ASH     #3,R1           ;MULTIPLY PREVIOUS VALUE BY 8
004540  060401                          ADD     R4,R1           ;ADD IN NEW DIGIT
004542  000764                          BR      2$
                                ;  HIT DELIMITER
004544  020427  000011          1$:     CMP     R4,#9.          ;Is this a decimal digit?
004550  101407                          BLOS    4$              ;Error -- He specified a decimal value
004552  005303                          DEC     R3              ;MAKE R3 POINT TO DELIMITER
004554  105767  173366                  TSTB    NEGFLG          ;Should we negate the value?
```

```
     58 004560  001401                        BEQ     9$              ;Br if not
     59 004562  005401                        NEG     R1              ;Negate the value
     60 004564  012604                9$:     MOV     (SP)+,R4
     61 004566  000207                        RETURN
     62                               ;
     63                               ;  Error -- Invalid octal value
     64                               ;
     65 004570                        4$:     FABORT  #EM$IOV         ;Invalid octal value
```

```
                                        .SBTTL  ACRSPD -- Accrue a line speed value
                                ;-----------------------------------------------------------------------
                                ;  Accrue a line speed value and return the corresponding speed code.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of speed parameter
                                ;
                                ;  Outputs:
                                ;   R3 = Pointer to delimiter past end of speed value.
                                ;   R5 = Speed code.
                                ;
004600  010146                  ACRSPD: MOV     R1,-(SP)
                                ;
                                ;  Accrue decimal speed value
                                ;
004602  004767  177446                  CALL    ACRDEC          ;Accrue decimal speed value
                                ;
                                ;  If speed was 134.5, skip over ".5"
                                ;
004606  020127  000206                  CMP     R1,#134.        ;Decimal value for speed = 134?
004612  001006                          BNE     2$              ;Br if not
004614  122327  000056                  CMPB    (R3)+,#'.       ;Skip period
004620  001013                          BNE     8$
004622  122327  000065                  CMPB    (R3)+,#'5       ;Skip 5
004626  001010                          BNE     8$
                                ;
                                ;  Convert speed to speed code
                                ;
004630  012705  000036          2$:     MOV     #2*15.,R5       ;Get index to highest speed value
004634  020165  000000'          3$:     CMP     R1,SPDVAL(R5)   ;Search for specified speed
004640  001407                          BEQ     4$              ;Br if found it
004642  162705  000002                  SUB     #2,R5           ;Try next table entry
004646  002372                          BGE     3$              ;Loop if more to check
                                ;
                                ;  Invalid speed value
                                ;
004650                          8$:     FABORT  #EM$ISV         ;Invalid speed value
                                ;
                                ;  Finished
                                ;
004660  006205                  4$:     ASR     R5              ;Get 1x speed code
004662  012601                          MOV     (SP)+,R1
004664  000207                          RETURN
```

```
                                        .SBTTL  OCTPRT -- Print an octal value
                                ;-------------------------------------------------
                                ;  OCTPRT IS A SUBROUTINE WHICH PRINTS OUT AN OCTAL VALUE.
                                ;  WHEN CALLED THE VALUE TO BE PRINTED MUST BE IN R2.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
004666  010046                  OCTPRT: MOV     R0,-(SP)
004670  010246                          MOV     R2,-(SP)
004672  012700  000030                  MOV     #30,R0
004676  000261                          SEC                     ;SET STOPPER BIT
004700  006102                  1$:     ROL     R2              ;MOVE 1ST BIT TO R0
004702  106100                          ROLB    R0
004704                                  .TTYOUT                 ;PRINT THE ASCII CHAR
004710  012700  000206                  MOV     #206,R0         ;SHIFTED 60 + REPEAT BIT
004714  006302                  2$:     ASL     R2              ;SHIFT OFF 1ST BIT
004716  001403                          BEQ     3$              ;BRANCH IF FINISHED
004720  106100                          ROLB    R0              ;MOVE BIT TO R0
004722  103774                          BCS     2$              ;DO LOOP 2 TIMES
004724  000765                          BR      1$              ;NOW GO GET 3RD BIT
004726  012602                  3$:     MOV     (SP)+,R2
004730  012600                          MOV     (SP)+,R0
004732  000207                          RETURN
```

```
                                        .SBTTL  OCTFIX -- Print octal value with fixed # spaces
                                ;-----------------------------------------------------------------------
                                ;  Print an octal value with a specified number of digits.
                                ;
                                ;  Inputs:
                                ;   R3 = Number of digits to print (1 - 6).
                                ;   R5 = Value to be printed.
                                ;
004734  010146          OCTFIX: MOV     R1,-(SP)
004736  010346                  MOV     R3,-(SP)
004740  010501                  MOV     R5,R1           ;Get value to be printed
004742  020327  000006          CMP     R3,#6.          ;Are we printing a full 6 digits?
004746  002404                  BLT     3$              ;Br if not
004750  005000                  CLR     R0              ;Shift 1st bit into R0
004752  073027  000001          ASHC    #1,R0
004756  000411                  BR      2$              ;Enter conversion loop
004760  012700  000020  3$:     MOV     #16.,R0         ;Determine # bits to shift to left
004764  160300                  SUB     R3,R0           ;  justify the data value in R1
004766  160300                  SUB     R3,R0
004770  160300                  SUB     R3,R0
004772  072100                  ASH     R0,R1           ;Left justify data value in R1
004774  005000          1$:     CLR     R0              ;Shift an octal digit into R0
004776  073027  000003          ASHC    #3,R0
005002  062700  000060  2$:     ADD     #'0,R0          ;Convert to ASCII character
005006                          .TTYOUT                 ;Print the digit
005012  077310                  SOB     R3,1$           ;Loop if more digits to print
005014  012603                  MOV     (SP)+,R3
005016  012601                  MOV     (SP)+,R1
005020  000207                  RETURN
```

```
                        .SBTTL  ACRTXT -- Accrue a character string
                ;-----------------------------------------------------------------------
                ;  Accrue a character string specified in the form
                ;  [=]string   or   [=]'string'   or   [=]"string"
                ;
                ;  Inputs:
                ;   R3 = Pointer to start of string
                ;
                ;  Outputs:
                ;   Accrued string is stored in asciz form in BLK0.
                ;   R0 = Number of characters in string (not counting null at end)
                ;   R3 = Points past end of string
                ;
005022  010146  ACRTXT: MOV     R1,-(SP)
005024  010546          MOV     R5,-(SP)
                ;
                ;  Skip up to start of string
                ;
005026  004767  007770          CALL    SKPSPC          ;Skip over any spaces
005032  121327  000075          CMPB    (R3),#'=        ;Was equal sign specified before string?
005036  001003                  BNE     1$              ;Br if not
005040  005203                  INC     R3              ;Skip past equal sign
005042  004767  007754          CALL    SKPSPC
                ;
                ;  See what the string delimiter is
                ;
005046  111305  1$:     MOVB    (R3),R5         ;Get string delimiter
005050  120527  000047          CMPB    R5,#47          ;Single quote?
005054  001431                  BEQ     6$              ;Br if yes
005056  120527  000042          CMPB    R5,#42          ;Double quote?
005062  001426                  BEQ     6$              ;Br if yes
                ;
                ;  String is not quoted.
                ;  Begin loop to get characters from the string.
                ;
005064  012701  000000G 2$:     MOV     #BLK0,R1        ;Point to buffer where we store result
                ;
                ;  Get next char from input string
                ;
005070  112300  4$:     MOVB    (R3)+,R0        ;Get next char from string
005072  001414                  BEQ     8$              ;Br if reached end of string
                ;
                ;  See if this is a control character sequence of the form ^char
                ;
005074  120027  000136          CMPB    R0,#'^          ;Start of control char sequence?
005100  001007                  BNE     3$              ;Br if not
005102  112300                  MOVB    (R3)+,R0        ;Get next char from string
005104  001422                  BEQ     10$             ;Br if delimiter missing
005106  120027  000136          CMPB    R0,#'^          ;Make "^^" = "^"
005112  001402                  BEQ     3$
005114  042700  177740          BIC     #^C<37>,R0      ;Convert char to control character
                ;
                ;  Store character into result buffer
                ;
005120  110021  3$:     MOVB    R0,(R1)+        ;Store char into result buffer
005122  000762                  BR      4$              ;Go get next char
                ;
```

```
                                ;  Reached end of string
                                ;
005124  005303                  8$:     DEC     R3              ;Point back to null at end of input string
005126  105011                          CLRB    (R1)            ;Store null at end of string
005130  162701  0000000G                SUB     #BLK0,R1        ;Determine length of string
005134  010100                          MOV     R1,R0           ;Return in R0
005136  000402                          BR      9$
                                ;
                                ;  Accrue a quoted string
                                ;
005140  004767  000016          6$:     CALL    ACRSTR          ;Accrue quoted string
                                ;
                                ;  Finished
                                ;
005144  012605                  9$:     MOV     (SP)+,R5
005146  012601                          MOV     (SP)+,R1
005150  000207                          RETURN
                                ;
                                ;  Invalid string
                                ;
005152                          10$:    FABORT  #EM$IST
```

```
                                        .SBTTL  ACRSTR -- Accrue a quoted character string
                                ;-----------------------------------------------------------------------
                                ;  Accrue a character string specified in the form
                                ;  [=]'string'   or   [=]"string"
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of string
                                ;
                                ;  Outputs:
                                ;   Accrued string is stored in asciz form in BLK0.
                                ;   R0 = Number of characters in string (not counting null at end)
                                ;   R3 = Points past end of string
                                ;
005162  010146                  ACRSTR: MOV     R1,-(SP)
005164  010546                          MOV     R5,-(SP)
                                ;
                                ;  Skip up to start of string
                                ;
005166  004767  007630                  CALL    SKPSPC          ;Skip over any spaces
005172  121327  000075                  CMPB    (R3),#'=        ;Was equal sign specified before string?
005176  001003                          BNE     1$              ;Br if not
005200  005203                          INC     R3              ;Skip past equal sign
005202  004767  007614                  CALL    SKPSPC
                                ;
                                ;  See what the string delimiter is
                                ;
005206  112305                  1$:     MOVB    (R3)+,R5        ;Get string delimiter
005210  120527  000047                  CMPB    R5,#47          ;Single quote?
005214  001403                          BEQ     2$              ;Br if yes
005216  120527  000042                  CMPB    R5,#42          ;Double quote?
005222  001031                          BNE     10$             ;Br if invalid delimiter
                                ;
                                ;  Begin loop to get characters from the string
                                ;
005224  012701  0000000G        2$:     MOV     #BLK0,R1        ;Point to buffer where we store result
                                ;
                                ;  Get next char from input string
                                ;
005230  112300                  4$:     MOVB    (R3)+,R0        ;Get next char from string
005232  001425                          BEQ     10$             ;Br if missing delimiter
005234  120005                          CMPB    R0,R5           ;Is this the end delimiter?
005236  001414                          BEQ     9$              ;Br if yes
                                ;
                                ;  See if this is a control character sequence of the form ^char
                                ;
005240  120027  000136                  CMPB    R0,#'^          ;Start of control char sequence?
005244  001007                          BNE     3$              ;Br if not
005246  112300                          MOVB    (R3)+,R0        ;Get next char from string
005250  001416                          BEQ     10$             ;Br if delimiter missing
005252  120027  000136                  CMPB    R0,#'^          ;Make "^^" = "^"
005256  001402                          BEQ     3$
005260  042700  177740                  BIC     #^C<37>,R0      ;Convert char to control character
                                ;
                                ;  Store character into result buffer
                                ;
005264  110021                  3$:     MOVB    R0,(R1)+        ;Store char into result buffer
005266  000760                          BR      4$              ;Go get next char
```

```
                                        ;
                                        ;  Finished
                                        ;
005270  105011                  9$:     CLRB    (R1)            ;Store null at end of string
005272  162701  0000006                 SUB     #BLK0,R1        ;Determine length of string
005276  010100                          MOV     R1,R0           ;Return in R0
005300  012605                          MOV     (SP)+,R5
005302  012601                          MOV     (SP)+,R1
005304  000207                          RETURN
                                        ;
                                        ;  Invalid string
                                        ;
005306                          10$:    FABORT  #EM$IST
```

```
                                        .SBTTL  GTRD50 -- Accrue a RAD50 value
                                ;-------------------------------------------------------------------------
                                ;  GTRD50 IS CALLED TO ACCRUE A RAD50 NAME.
                                ;  WHEN CALLED R3 MUST POINT TO THE BEGINNING OF THE NAME
                                ;  ANY LEADING SPACES ARE SKIPPED.
                                ;  ON RETURN THE ACCRUED VALUE IS STORED IN R50BUF AND
                                ;  R50BUF+2.  ALL REGISTERS ARE PRESERVED EXCEPT R3
                                ;  WHICH POINTS TO THE END OF THE NAME ON RETURN.
                                ;
005316  010046                  GTRD50: MOV     R0,-(SP)
005320  010146                          MOV     R1,-(SP)
                                ;  SKIP LEADING BLANKS.
005322  121327  000040          2$:     CMPB    (R3),#' ;IS THIS CHAR ABLANK?
005326  001002                          BNE     1$      ;BRANCH IF NOT
005330  005203                          INC     R3      ;KEEP SKIPPING
005332  000773                          BR      2$
005334  012746  022000          1$:     MOV     #22000,-(SP)    ;TELL US WHEN TO STOP
005340  005001                  8$:     CLR     R1      ;FORM VALUE IN R1
005342  111300                  7$:     MOVB    (R3),R0 ;GET NEXT CHAR
                                ;  CHECK FOR WILDCARD CHARACTER ('*')
005344  120027  000052                  CMPB    R0,#'*  ;WILDCARD?
005350  001003                          BNE     10$     ;BR IF NOT
005352  012700  000035                  MOV     #35,R0  ;RETURN CODE FOR *
005356  000434                          BR      5$
                                ;  CHECK FOR DIGIT
005360  120027  000060          10$:    CMPB    R0,#'0
005364  103406                          BLO     3$      ;BR IF DELIMITER
005366  120027  000071                  CMPB    R0,#'9
005372  101005                          BHI     4$      ;BR IF NOT DIGIT
005374  062700  177756                  ADD     #<36-'0>,R0     ;CONVERT DIGIT TO RAD50 VAL
005400  000423                          BR      5$
                                ;  HIT DELIMITER
005402  005000                  3$:     CLR     R0      ;CONVERT TO SPACE
005404  000422                          BR      6$
                                ;  CHECK FOR ALPHA CHARACTER
005406  120027  000141          4$:     CMPB    R0,#141 ;IS THIS A LOWER CASE LETTER?
005412  103406                          BLO     12$     ;BR IF NOT
005414  120027  000172                  CMPB    R0,#172 ;LOWER CASE Z
005420  101370                          BHI     3$      ;BR IF MUST BE DELIMITER
005422  162700  000040                  SUB     #40,R0  ;CONVERT LOWER CASE TO UPPER CASE
005426  000406                          BR      13$
005430  120027  000101          12$:    CMPB    R0,#'A
005434  103762                          BLO     3$      ;BRANCH IF DELIMITER
005436  120027  000132                  CMPB    R0,#'Z
005442  101357                          BHI     3$      ;BRANCH IF DELIMITER
005444  062700  177700          13$:    ADD     #<1-'A>,R0      ;CONVERT TO RAD50 VAL
005450  005203                  5$:     INC     R3      ;POINT TO NEXT CHAR
                                ;  MULTIPLY PREVIOUS VALUE BY 50 AND ADD NEW VALUE
005452  070127  000050          6$:     MUL     #50,R1  ;MULTIPLY PREVIOUS VALUE BY 50
005456  060001                          ADD     R0,R1   ;ADD NEW VALUE
005460  006316                          ASL     @SP     ;TEST FOR END OF 3 CHARS
005462  103327                          BCC     7$      ;BRANCH IF NOT END OF 3
005464  005716                          TST     @SP     ;FINISHED 6 CHARACTERS?
005466  001403                          BEQ     9$      ;BRANCH IF YES
005470  010167  000000G               MOV     R1,R50BUF       ;SAVE 1ST 3 CHARS
005474  000721                          BR      8$
005476  010167  000002G         9$:    MOV     R1,<R50BUF+2>   ;SAVE 2ND 3 CHARS
```

```
     58 005502  005726                          TST     (SP)+           ;CLEAN OFF STACK
     59                                ;  SKIP ANY CHARACTERS IN NAME AFTER SIXTH
     60 005504  112300                 11$:     MOVB    (R3)+,R0        ;GET NEXT CHAR
     61 005506  004767  007116                   CALL    CHKDLM          ;SEE IF IT IS A DELIMITER
     62 005512  103374                           BCC     11$             ;LOOP IF NOT
     63 005514  005303                           DEC     R3              ;POINT TO DELIMITER
     64 005516  012601                           MOV     (SP)+,R1
     65 005520  012600                           MOV     (SP)+,R0
     66 005522  000207                           RETURN
```

```
                                        .SBTTL  PRTPCT -- Print percentage value
                                ;-------------------------------------------------------------------------
                                ;  PRTPCT is called to convert a 32-bit value to a percentage and print the
                                ;  resulting value.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of 32-bit value to be converted (stored high-order word first)
                                ;   DIVSOR = 32-bit divisor to use for computing percentage.
                                ;
005524  010146                  PRTPCT: MOV     R1,-(SP)
005526  010446                          MOV     R4,-(SP)
005530  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get dividend and multiply by 100.
                                ;
005532  012104                          MOV     (R1)+,R4        ;GET HIGH-ORDER VALUE
005534  012105                          MOV     (R1)+,R5        ;GET LOW-ORDER VALUE
005536  012700  000144                  MOV     #100.,R0        ;SET MULTIPLIER
005542  004767  000334                  CALL    MUL32           ;MULTIPLY BY 100.
                                ;
                                ;  Divide to compute percentage
                                ;
005546  004767  000232                  CALL    DIV32           ;DIVIDE TO COMPUTE PERCENTAGE
                                ;
                                ;  16-bit quotient is now in R5.
                                ;  Print the value.
                                ;
005552  004767  000376                  CALL    PRTDEC          ;PRINT THE VALUE
                                ;
                                ;  Print percent sign
                                ;
005556  012700  000045                  MOV     #'%,R0          ;GET PERCENT SIGN
005562                                  .TTYOUT                 ;PRINT IT
                                ;
                                ;  Finished
                                ;
005566  012605                          MOV     (SP)+,R5
005570  012604                          MOV     (SP)+,R4
005572  012601                          MOV     (SP)+,R1
005574  000207                          RETURN
```

```
                                        .SBTTL  PRTR50 -- Print a RAD50 value
                                ;-------------------------------------------------------------------------
                                ;  PRTR50 is called to print a Rad-50 value.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to be printed.
                                ;
005576  010146                  PRTR50: MOV     R1,-(SP)
005600  010246                          MOV     R2,-(SP)
                                ;
                                ;  Convert value to ascii string and stack the characters.
                                ;
005602  012702  000003                  MOV     #3.,R2          ;CONVERT 3 CHARS
005606  005046                          CLR     -(SP)           ;PUT NULL ON STACK TO SIGNAL END
005610  010001                          MOV     R0,R1           ;GET VALUE TO BE CONVERTED
005612  005000                  1$:     CLR     R0              ;CLEAR HIGH-ORDER VALUE
005614  071027  000050                  DIV     #50,R0          ;DIVIDE R0-R1 BY 50
005620  116146  000000G                 MOVB    R50CHR(R1),-(SP);CONVERT REMAINDER TO ASCII CHARACTER & STACK
005624  010001                          MOV     R0,R1           ;GET QUOTIENT
005626  077207                          SOB     R2,1$           ;LOOP IF MORE CHARS TO CONVERT
                                ;
                                ;  Finished conversion.  Print the result.
                                ;
005630  012600                  2$:     MOV     (SP)+,R0        ;GET CHAR TO PRINT
005632  001403                          BEQ     3$              ;BR IF HIT END
005634                                  .TTYOUT                 ;PRINT THE ASCII CHARACTER
005640  000773                          BR      2$              ;KEEP GOING
                                ;
                                ;  Finished
                                ;
005642  012602                  3$:     MOV     (SP)+,R2
005644  012601                          MOV     (SP)+,R1
005646  000207                          RETURN
```

```
                        .SBTTL  PRTFNM -- Print a file name
                ;-----------------------------------------------------------------
                ;  Convert a 1 to 6 character file name that is stored in rad50 form
                ;  into an ascii string and store the string into a specified buffer.
                ;  Trailing spaces are not printed.
                ;
                ;  Inputs:
                ;   R0 = Pointer to 2 words containing file name in RAD50 form.
                ;   R3 = Pointer to buffer where ascii string is to be stored.
                ;
                ;  Outputs:
                ;   R3 = Pointer past end of name in buffer.
                ;
005650  010146  PRTFNM: MOV     R1,-(SP)
005652  010246          MOV     R2,-(SP)
005654  010446          MOV     R4,-(SP)
005656  010546          MOV     R5,-(SP)
005660  010005          MOV     R0,R5           ;Get pointer to file name
005662  012704  000002  MOV     #2,R4           ;Convert and print 2 words
                ;
                ;  Convert RAD50 name to ascii string and stack the characters
                ;
005666  005046  4$:     CLR     -(SP)           ;Put null on stack to mark the end
005670  012702  000003  MOV     #3,R2           ;Convert 3 characters from this word
005674  012501          MOV     (R5)+,R1        ;Get RAD50 value to be converted
005676  005000  1$:     CLR     R0              ;Clear high-order word for divide
005700  071027  000050  DIV     #50,R0          ;Divide R0-R1 by 50
005704  116146  000000G MOVB    R50CHR(R1),-(SP);Stack next char of file name
005710  010001          MOV     R0,R1           ;Get quotient
005712  077207          SOB     R2,1$           ;Loop if more chars to convert
                ;
                ;  Finished converting a word.  Move characters to buffer.
                ;
005714  012600  2$:     MOV     (SP)+,R0        ;Get next character of name
005716  001405          BEQ     5$              ;Br if hit end of word
005720  120027  000040  CMPB    R0,#40          ;Is this character a space?
005724  001773          BEQ     2$              ;Don't print spaces
005726  110023          MOVB    R0,(R3)+        ;Move character to buffer
005730  000771          BR      2$
005732  077423  5$:     SOB     R4,4$           ;Loop if 2nd word to convert
                ;
                ;  Finished
                ;
005734  012605  3$:     MOV     (SP)+,R5
005736  012604          MOV     (SP)+,R4
005740  012602          MOV     (SP)+,R2
005742  012601          MOV     (SP)+,R1
005744  000207          RETURN
```

```
                                        .SBTTL  DIVIDE -- Divide 32-bit qty by 16-bit
                                ;-----------------------------------------------------------------
                                ;  SUBROUTINE DIVIDE IS CALLED TO DIVIDE THE 32-BIT QUANTITY
                                ;  IN R4 (HIGH ORDER) AND R5 (LOW ORDER) BY THE 16-BIT
                                ;  QUANTITY IN R3.  ON RETURN THE QUOTIENT IS IN R4-R5
                                ;  AND THE REMAINDER IS IN R0.  ALL OTHER REGISTERS ARE PRESERVED.
                                ;
005746  010246                  DIVIDE: MOV     R2,-(SP)
005750  005000                          CLR     R0              ;INITIALIZE REMAINDER
005752  012702  000037                  MOV     #31.,R2         ;GET SHIFT COUNT
005756  006305                  1$:     ASL     R5              ;SHIFT BIT OUT OF LOW ORDER
005760  006104                          ROL     R4              ;SHIFT THROUGH HIGH ORDER
005762  006100                          ROL     R0              ;INTO R0
005764  020003                          CMP     R0,R3           ;GOT  ENOUGH TO SUBTRACT YET?
005766  103402                          BLO     2$              ;BR IF NOT
005770  160300                          SUB     R3,R0           ;SUBTRACT DIVISOR
005772  005205                          INC     R5              ;INCREASE QUOTIENT
005774  005302                  2$:     DEC     R2              ;COUNT # BITS SHIFTED
005776  100367                          BPL     1$              ;BR IF MORE TO DO
006000  012602                          MOV     (SP)+,R2
006002  000207                          RETURN
```

```
                                        .SBTTL  DIV32  -- Divide 32-bit qty by 32-bit qty
                                ;-----------------------------------------------------------------------
                                ;  DIV32 divides one 32-bit value by another 32-bit value producing
                                ;  a 32-bit quotient and a 32-bit remainder.
                                ;
                                ;  Inputs:
                                ;   R4-R5 = Dividend (R4 = high-order, R5 = low-order)
                                ;   DIVSOR = Divisor (High-order in 1st word)
                                ;
                                ;  Outputs:
                                ;   R4-R5 = Quotient
                                ;   REMNDR = 32-bit remainder
                                ;
006004  010246          DIV32:  MOV     R2,-(SP)
006006  010346                  MOV     R3,-(SP)
006010  005002                  CLR     R2              ;INITIALIZE REMAINDER (R2-R3)
006012  005003                  CLR     R3
006014  012700  000040          MOV     #32.,R0         ;GET SHIFT COUNT
006020  073427  000001  1$:     ASHC    #1,R4           ;SHIFT BIT OUT OF DIVIDEND
006024  006103                  ROL     R3              ;AND INTO REMAINDER
006026  006102                  ROL     R2
006030  020267  000000G         CMP     R2,DIVSOR       ;GOT ENOUGH TO SUBTRACT YET?
006034  103412                  BLO     2$              ;BR IF NOT
006036  101003                  BHI     3$              ;BR IF YES
006040  020367  000002G         CMP     R3,DIVSOR+2     ;CHECK LOW-ORDER PART
006044  103406                  BLO     2$              ;BR IF NOT ENOUGH YET
006046  166703  000002G  3$:     SUB     DIVSOR+2,R3     ;SUBTRACT LOW-ORDER DIVISOR
006052  005602                  SBC     R2              ;PROPOGATE BORROW
006054  166702  000000G         SUB     DIVSOR,R2       ;SUBTRACT HIGH-ORDER DIVISOR
006060  005205                  INC     R5              ;INCREASE QUOTIENT
006062  077022          2$:     SOB     R0,1$           ;DO FULL DIVIDE
006064  010267  000000G         MOV     R2,REMNDR       ;STORE HIGH-ORDER REMAINDER
006070  010367  000002G         MOV     R3,REMNDR+2     ;STORE LOW-ORDER REMAINDER
006074  012603                  MOV     (SP)+,R3
006076  012602                  MOV     (SP)+,R2
006100  000207                  RETURN
```

```
                                        .SBTTL  MUL32  -- Multiply 32-bit qty by 16-bit qty
                                ;-----------------------------------------------------------------------
                                ;  MUL32 is called to multiply a 32-bit value by a 16-bit value producing
                                ;  a 32-bit product.
                                ;
                                ;  Inputs:
                                ;   R4-R5 = 32-bit value to be multiplied (R4=high-order, R5=low-order)
                                ;   R0 = 16-bit multiplier
                                ;
                                ;  Outputs:
                                ;   R4-R5 = 32-bit product.
                                ;   R0 is preserved.
                                ;
006102  010046                  MUL32:  MOV     R0,-(SP)
006104  010246                          MOV     R2,-(SP)
006106  010346                          MOV     R3,-(SP)
006110  010402                          MOV     R4,R2           ;COPY VALUE TO BE MULTIPLIED
006112  010503                          MOV     R5,R3
006114  005004                          CLR     R4              ;FORM PRODUCT IN R4-R5
006116  005005                          CLR     R5
006120  000241                  1$:     CLC
006122  006000                          ROR     R0              ;SHOULD WE ADD IN THIS TIME?
006124  103003                          BCC     2$              ;BR IF NOT
006126  060305                          ADD     R3,R5           ;ADD LOW-ORDER PART
006130  005504                          ADC     R4              ;PROPOGATE CARRY
006132  060204                          ADD     R2,R4           ;ADD HIGH-ORDER PART
006134  073227  000001          2$:     ASHC    #1,R2           ;SHIFT MULTIPLICAN VALUE
006140  005700                          TST     R0              ;MORE TO MULTIPLY?
006142  001366                          BNE     1$              ;LOOP IF YES
006144  012603                          MOV     (SP)+,R3
006146  012602                          MOV     (SP)+,R2
006150  012600                          MOV     (SP)+,R0
006152  000207                          RETURN
```

```
                                        .SBTTL  PRTDEC -- Print a decimal value
                                ;-----------------------------------------------------------
                                ;  PRTDEC IS CALLED TO PRINT THE DECIMAL VALUE IN R5.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
006154  010046                  PRTDEC: MOV     R0,-(SP)
006156  010246                          MOV     R2,-(SP)
006160  010346                          MOV     R3,-(SP)
006162  010446                          MOV     R4,-(SP)
006164  010546                          MOV     R5,-(SP)
006166  012702  0000000G                MOV     #PBUFND,R2      ;POINT TO END OF PRINT BUFFER
006172  005004                  1$:     CLR     R4              ;CLEAR HIGH-ORDER FOR DIVIDE
006174  071427  000012                  DIV     #10.,R4         ;DIVIDE R4-R5 BY 10.
006200  062705  000060                  ADD     #'0,R5          ;CONVERT REMAINDER TO ASCII DIGIT
006204  110542                          MOVB    R5,-(R2)        ;STORE THE CHARACTER
006206  010405                          MOV     R4,R5           ;ANYTHING LEFT TO CONVERT
006210  001370                          BNE     1$              ;BR IF YES
                                ;  VALUE IS CONVERTED, PRINT IT.
006212                                  .PRINT  R2
006216  012605                          MOV     (SP)+,R5
006220  012604                          MOV     (SP)+,R4
006222  012603                          MOV     (SP)+,R3
006224  012602                          MOV     (SP)+,R2
006226  012600                          MOV     (SP)+,R0
006230  000207                          RETURN
```

```
                                        .SBTTL  PRTLN  -- Print a job number
                                ;-----------------------------------------------------------------
                                ;  Print a job number.
                                ;  A job index number is divided by 2 to produce a job number and that
                                ;  job number is printed in a field with 2 positions.
                                ;
                                ;  Inputs:
                                ;   R5 = Job index number of job whose number is to be printed.
                                ;
006232  010346                  PRTLN:  MOV     R3,-(SP)
006234  010546                          MOV     R5,-(SP)
006236  006205                          ASR     R5              ;Convert job index # to job #
006240  012703  000002                  MOV     #2,R3           ;Print value in 2 character field
006244  004767  000006                  CALL    PRTFIX          ;Print it
006250  012605                          MOV     (SP)+,R5
006252  012603                          MOV     (SP)+,R3
006254  000207                          RETURN
```

```
                                        .SBTTL  PRTFIX -- Print value with fixed field width
                                ;-----------------------------------------------------------------------
                                ;  PRTFIX is called to print a decimal value using a specified number
                                ;  of columns.
                                ;  Leading spaces are inserted if necessary to pad the value to the specified
                                ;  size.
                                ;
                                ;  Inputs:
                                ;   R3 = Number of columns to print.
                                ;   R5 = Value to print.
                                ;
006256  010046                  PRTFIX: MOV     R0,-(SP)
006260  010246                          MOV     R2,-(SP)
006262  010346                          MOV     R3,-(SP)
006264  010446                          MOV     R4,-(SP)
006266  010546                          MOV     R5,-(SP)
006270  012702  0000000G                MOV     #PBUFND,R2      ;POINT TO END OF CONVERSION BUFFER
006274  005004                  1$:     CLR     R4              ;CLEAR HIGH-ORDER FOR DIVIDE
006276  071427  000012                  DIV     #10.,R4         ;DIVIDE R4-R5 BY 10.
006302  062705  000060                  ADD     #'0,R5          ;CONVERT REMAINDER TO ASCII DIGIT
006306  110542                          MOVB    R5,-(R2)        ;MOVE TO PRINT BUFFER
006310  005303                          DEC     R3              ;COUNT DOWN # OF COLUMNS USED
006312  010405                          MOV     R4,R5           ;GET REMAINING QUOTIENT
006314  001367                          BNE     1$              ;BR IF MORE TO CONVERT
006316  005703                          TST     R3              ;DO WE NEED TO PUT IN PADDING SPACES?
006320  003403                          BLE     2$              ;BR IF NOT
006322  112742  000040          3$:     MOVB    #' ,-(R2)       ;INSERT LEADING SPACES
006326  077303                          SOB     R3,3$
006330                          2$:     .PRINT  R2              ;PRINT THE FINAL RESULT
006334  012605                          MOV     (SP)+,R5
006336  012604                          MOV     (SP)+,R4
006340  012603                          MOV     (SP)+,R3
006342  012602                          MOV     (SP)+,R2
006344  012600                          MOV     (SP)+,R0
006346  000207                          RETURN
```

```
                                        .SBTTL  PRTDC2 -- Print decimal value with 2 digits
                                ;-------------------------------------------------------------
                                ;  PRTDC2 PRINTS A DECIMAL VALUE CONTAINED IN R5 LIKE PRTDEC.
                                ;  HOWEVER, PRTDC2 ALWAYS PRINTS AT LEAST TWO DIGITS IN THE
                                ;  NUMBER.   ALL REGISTERS ARE PRESERVED.
                                ;
006350  020527  000011          PRTDC2: CMP     R5,#9.          ;DOES VALUE USE 1 OR MORE DIGITS?
006354  101006                          BHI     1$              ;BR IF USES MORE THAN 1 DIGIT
006356  010046                          MOV     R0,-(SP)
006360                                  .TTYOUT #'0             ;PRINT LEADING ZERO
006370  012600                          MOV     (SP)+,R0
006372  004767  177556          1$:     CALL    PRTDEC          ;NOW PRINT VALUE
006376  000207                          RETURN

                                        .SBTTL  PRTDC3 -- Print decimal value with 3 digits
                                ;-------------------------------------------------------------
                                ;  PRTDC3 prints a decimal value and uses at lease 3 columns to
                                ;  display the value.  Leading spaces are printed if necessary to
                                ;  fill out the three columns.
                                ;
                                ;  Inputs:
                                ;   R5 = Value to be printed.
                                ;
006400  020527  000143          PRTDC3: CMP     R5,#99.         ;Will value print with 3 digits?
006404  101004                          BHI     1$              ;Br if yes
006406                                  .TTYOUT #40             ;Print a blank if not
006416  020527  000011          1$:     CMP     R5,#9.          ;Will value print with 2 digits?
006422  101004                          BHI     2$              ;Br if yes
006424                                  .TTYOUT #40             ;Print second blank if not
006434  004767  177514          2$:     CALL    PRTDEC          ;Print actual value
006440  000207                          RETURN

                                        .SBTTL  PRTSPC -- Print specified number of spaces
                                ;-------------------------------------------------------------
                                ;  Print the specified number of spaces.
                                ;
                                ;  Inputs:
                                ;   R3 = Number of spaces to print
                                ;
006442  010346                  PRTSPC: MOV     R3,-(SP)
006444  001405                          BEQ     9$
006446                          1$:     .TTYOUT #40             ;Print a space
006456  077305                          SOB     R3,1$           ;Loop if more to print
006460  012603                  9$:     MOV     (SP)+,R3
006462  000207                          RETURN
```

```
                                        .SBTTL  PRTTTP -- Print terminal type name
                                ;-----------------------------------------------------------------------
                                ;  PRTTTP prints the terminal type being used by a specified line.
                                ;  The terminal type name that is printed is 9 characters long.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
006464  010246                  PRTTTP: MOV     R2,-(SP)
006466  010446                          MOV     R4,-(SP)
006470  016104  000000G                 MOV     LTRMTP(R1),R4   ;Get terminal type flags
006474  032761  000000G 000000G         BIT     #$KINIT,LSW(R1) ;Has line initializion been done yet?
006502  001002                          BNE     1$              ;Br if yes
006504  016104  000000G                 MOV     ITRMTP(R1),R4   ;Get sysgen terminal type code
006510  012702  000022          1$:     MOV     #NTRMTP,R2      ;Get # terminal types
006514  020462  006556'        2$:     CMP     R4,TTFTBL(R2)   ;Is this the terminal type?
006520  001407                          BEQ     3$              ;Br if yes
006522  162702  000002                  SUB     #2,R2           ;More to check?
006526  002372                          BGE     2$              ;Br if yes
006530                          4$:     .PRINT  #TTNXXX         ;Unknown terminal type
006536  000404                          BR      9$
006540  016202  006602'        3$:     MOV     TTNTBL(R2),R2   ;Get pointer to terminal name string
006544                                  .PRINT  R2              ;Print terminal name
                                ;
                                ;  Finished
                                ;
006550  012604                  9$:     MOV     (SP)+,R4
006552  012602                          MOV     (SP)+,R2
006554  000207                          RETURN

                                ;
                                ;  Terminal type flag table
                                ;
006556  000000G                 TTFTBL: .WORD   VT52             ;VT52
006560  000000G                         .WORD   VT100           ;VT100
006562  000000G                         .WORD   HAZEL           ;HAZELTINE
006564  000000G                         .WORD   ADM3A           ;ADM3A
006566  000000G                         .WORD   LA36            ;LA36
006570  000000G                         .WORD   LA120           ;LA120
006572  000000G                         .WORD   DIABLO          ;DIABLO
006574  000000G                         .WORD   QUME            ;QUME
006576  000000G                         .WORD   VT2007          ;VT200 -- 7 bit control codes
006600  000000G                         .WORD   VT2008          ;VT200 -- 8 bit control codes
        000022                  NTRMTP  =       <.-TTFTBL>-2    ;Highest terminal type index
                                ;
                                ;  Terminal type name pointer table
                                ;
006602  006637'                 TTNTBL: .WORD   TTNV52           ;VT52
006604  006650'                         .WORD   TTNV10          ;VT100
006606  006672'                         .WORD   TTNHZL          ;HAZELTINE
006610  006703'                         .WORD   TTNADM          ;ADM3A
006612  006714'                         .WORD   TTNL36          ;LA36
006614  006725'                         .WORD   TTNL12          ;LA120
006616  006736'                         .WORD   TTNDIA          ;DIABLO
006620  006747'                         .WORD   TTNQUM          ;QUME
006622  006661'                         .WORD   TTNV20          ;VT200
006624  006661'                         .WORD   TTNV20          ;VT200
```

```
                                        ;
                                        ;  Terminal name strings
                                        ;
                                                .NLIST  BEX
006626      165     156     153     TTNXXX: .ASCII  /unknown /<200>
006637      126     124     065     TTNV52: .ASCII  /VT52    /<200>
006650      126     124     061     TTNV10: .ASCII  /VT100   /<200>
006661      126     124     062     TTNV20: .ASCII  /VT200   /<200>
006672      110     141     172     TTNHZL: .ASCII  /Hazeltne/<200>
006703      101     104     115     TTNADM: .ASCII  /ADM3A   /<200>
006714      114     101     063     TTNL36: .ASCII  /LA36    /<200>
006725      114     101     061     TTNL12: .ASCII  /LA120   /<200>
006736      104     151     141     TTNDIA: .ASCII  /Diablo  /<200>
006747      121     165     155     TTNQUM: .ASCII  /Qume    /<200>
                                                .EVEN
                                                .LIST   BEX
```

```
                                        .SBTTL  EDTFIL -- Edit file spec
                                ;-----------------------------------------------------------------------
                                ;  EDTFIL is called to convert a file specification stored in RAD50
                                ;  form into an asciz string of the form dev:name.ext
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of area where result is to be stored.
                                ;   R4 = Pointer to 4-word block with file spec in RAD50 form.
                                ;
                                ;  Outputs:
                                ;   Asciz file spec is in result buffer.
                                ;   R3 = Points to null at end of string.
                                ;
006760  010446                  EDTFIL: MOV     R4,-(SP)
                                ;
                                ;  Edit device name
                                ;
006762  012400                          MOV     (R4)+,R0        ;GET DEVICE NAME
006764  001404                          BEQ     1$              ;BR IF NO DEVICE SPECIFIED
006766  004767  000046                  CALL    EDTR50          ;EDIT INTO BUFFER
006772  112723  000072                  MOVB    #':,(R3)+       ;TERMINATE DEV NAME WITH COLON
                                ;
                                ;  Get file name
                                ;
006776  012400                  1$:     MOV     (R4)+,R0        ;GET 1ST 3 CHARS OF FILE NAME
007000  001414                          BEQ     3$              ;BR IF NO FILE NAME
007002  004767  000032                  CALL    EDTR50          ;EDIT IN 1ST 3 CHARS
007006  012400                          MOV     (R4)+,R0        ;GET 2ND 3 CHARS
007010  001402                          BEQ     2$              ;BR IF ALL BLANK
007012  004767  000022                  CALL    EDTR50          ;EDIT INTO BUFFER
                                ;
                                ;  Put in extension
                                ;
007016  012400                  2$:     MOV     (R4)+,R0        ;GET EXTENSION
007020  001404                          BEQ     3$              ;BR IF NO EXTENSION
007022  112723  000056                  MOVB    #'.,(R3)+       ;PUT PERIOD AT END OF FILE NAME
007026  004767  000006                  CALL    EDTR50          ;EDIT IN EXTENSION
                                ;
                                ;  Finished
                                ;
007032  105013                  3$:     CLRB    (R3)            ;PUT IN NULL AT END OF STRING
007034  012604                          MOV     (SP)+,R4
007036  000207                          RETURN
```

```
                                        .SBTTL  EDTR50 -- Convert RAD50 value to ascii
                                ;-----------------------------------------------------------------
                                ;  EDTR50 is called to convert a RAD50 value to an ascii
                                ;  character string and store the string into a specified buffer.
                                ;
                                ;  Inputs:
                                ;   R0 = RAD50 value to convert
                                ;   R3 = Pointer to buffer where result is to be stored.
                                ;
                                ;  Outputs:
                                ;   Ascii string is stored into result buffer.
                                ;   R3 = Pointer past end of last character stored into buffer.
                                ;
007040  010146                  EDTR50: MOV     R1,-(SP)
007042  010246                          MOV     R2,-(SP)
                                ;
                                ;  See if value is wildcard ("*")
                                ;
007044  020027  000000G                 CMP     R0,#WLDNAM      ;Wildcard?
007050  001003                          BNE     5$              ;Br if not
007052  112723  000052                  MOVB    #'*,(R3)+       ;Store wildcard character
007056  000420                          BR      9$
                                ;
                                ;  Convert value to ascii characters and stack the characters
                                ;
007060  005046                  5$:     CLR     -(SP)           ;PUT NULL ON STACK TO SIGNAL END OF CHARS
007062  012702  000003                  MOV     #3,R2           ;GET # OF CHARS TO CONVERT
007066  010001                          MOV     R0,R1           ;GET VALUE TO CONVERT
007070  005000                  1$:     CLR     R0              ;CLEAR HIGH-ORDER FOR DIVIDE
007072  071027  000050                  DIV     #50,R0          ;DIVIDE R0-R1 BY 50
007076  005701                          TST     R1              ;IS THE CHARACTER A SPACE?
007100  001402                          BEQ     4$              ;BR IF YES
007102  116146  000000G                 MOVB    R50CHR(R1),-(SP);STACK THE CHARACTER
007106  010001                  4$:     MOV     R0,R1           ;GET QUOTIENT
007110  077211                          SOB     R2,1$           ;LOOP IF MORE TO CONVERT
                                ;
                                ;  Move characters from the stack to the result buffer
                                ;
007112  112623                  2$:     MOVB    (SP)+,(R3)+     ;MOVE CHAR TO RESULT
007114  001376                          BNE     2$              ;LOOP IF MORE TO MOVE
007116  005303                          DEC     R3              ;POINT PAST LAST CHAR
                                ;
                                ;  Finished
                                ;
007120  012602                  9$:     MOV     (SP)+,R2
007122  012601                          MOV     (SP)+,R1
007124  000207                          RETURN
```

```
                                        .SBTTL  PRTUNM -- Print user name or PPN
                                ;-----------------------------------------------------------------
                                ;  PRTUNM is called to print the name of the user (or the PPN
                                ;  if no user name is present) for the user of a time-sharing line.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number
                                ;
007126  010446                  PRTUNM: MOV     R4,-(SP)
007130  010546                          MOV     R5,-(SP)
                                ;
                                ;  If line is not logged on, there is nothing to print
                                ;
007132  032761  000000G 000000G         BIT     #$KINIT,LSW(R1) ;Has line been initialized?
007140  001442                          BEQ     10$             ;Br if not
                                ;
                                ;  Did user request PPN display?
                                ;
007142  105767  171001                  TSTB    SJSPPN          ;SHOW JOBS/PPN?
007146  001017                          BNE     15$             ;BRANCH IF SO
                                ;
                                ;  See if user name is known
                                ;
007150  010105                          MOV     R1,R5           ;GET JOB #
007152  012704  000014                  MOV     #12.,R4         ;EACH JOB HAS A 12. CHAR USER NAME
007156  070527  000006                  MUL     #6,R5           ;GET POINTER TO USER NAME FOR THIS JOB
007162  062705  000000G                 ADD     #LUNAME,R5
007166  121527  000040                  CMPB    (R5),#'         ;IS USER NAME BLANK?
007172  001405                          BEQ     15$             ;BR IF YES -- PRINT PPN INSTEAD
007174                          16$:    .TTYOUT (R5)+           ;PRINT USER NAME
007202  077404                          SOB     R4,16$
007204  000420                          BR      10$
                                ;
                                ;  Don't know user name, print PPN
                                ;
007206  016105  000000G         15$:    MOV     LPROJ(R1),R5    ;GET PROJECT #
007212  001415                          BEQ     10$             ;BR IF NOT LOGGED ON
007214                                  .PRINT  #PPNMSG
007222  004767  176726          9$:     CALL    PRTDEC          ;PRINT PROJECT NUMBER
007226  112700  000054                  MOVB    #',,R0
007232                                  .TTYOUT
007236  016105  000000G                 MOV     LPROG(R1),R5    ;PRINT PROGRAMMER NUMBER
007242  004767  176706                  CALL    PRTDEC
                                ;
                                ;  Finished
                                ;
007246  012605                  10$:    MOV     (SP)+,R5
007250  012604                          MOV     (SP)+,R4
007252  000207                          RETURN
```

```
                                        .SBTTL  PRTTIM -- Print job statistics
                                ;-----------------------------------------------------------
                                ;  PRTTIM IS CALLED TO PRINT THE JOB ACCOUNTING STATISTICS FOR
                                ;  THE JOB WHOSE LINE INDEX NUMBER IS IN R1.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
007254  010046                  PRTTIM: MOV     R0,-(SP)
007256  010346                          MOV     R3,-(SP)
007260  010446                          MOV     R4,-(SP)
007262  010546                          MOV     R5,-(SP)
                                ;
                                ;  PRINT CONNECT TIME
                                ;
007264                                  .PRINT  #CTMSG          ;PRINT CONNECT TIME HEADER
007272  016705  000000G                 MOV     MINTIM,R5       ;GET CURRENT MINUTE TIMER VALUE
007276  166105  000000G                 SUB     LCONTM(R1),R5   ;CALCULATE CONNECT TIME FOR LINE
007302  005205                          INC     R5              ;CHARGE A MINIMUM OF 1 MINUTE
007304  005004                  1$:     CLR     R4              ;CLEAR HIGH-ORDER FOR DIVIDE
007306  012703  000074                  MOV     #60.,R3         ;SET TO DIVIDE BY 60.
007312  004767  176430                  CALL    DIVIDE          ;DIVIDE BY 60
007316  004767  177026                  CALL    PRTDC2          ;PRINT # HOURS CONNECTED
007322  010005                          MOV     R0,R5           ;GET # MINUTES CONNECTED
007324                                  .TTYOUT #':             ;PRINT COLON AFTER HOURS
007334  004767  177010                  CALL    PRTDC2          ;PRINT # MINUTES CONNECTED
007340                                  .PRINT  #COL00          ;PRINT ':00' SECONDS
                                ;
                                ;  PRINT CPU TIME
                                ;
007346                                  .PRINT  #CPUMSG         ;PRINT CPU HEADER MESSAGE
007354  016104  0000000G                MOV     LCPUHI(R1),R4   ;GET HIGH ORDER CPU TIME (CLOCK TICKS)
007360  016105  0000000G                MOV     LCPULO(R1),R5   ;GET LOW ORDER CPU TIME (CLOCK TICKS)
007364  004767  000020                  CALL    PRTTMV          ;PRINT TIME VALUE
007370                                  .PRINT  #CRLF           ;END PRINT LINE
007376  012605                          MOV     (SP)+,R5
007400  012604                          MOV     (SP)+,R4
007402  012603                          MOV     (SP)+,R3
007404  012600                          MOV     (SP)+,R0
007406  000207                          RETURN
```

```
                        .SBTTL  PRTTMV -- Print a time value
                ;-----------------------------------------------------------------------
                ;  PRTTMV is called to display a time value in the format HH:MM:SS
                ;
                ;  Inputs:
                ;   R4 = High-order time value (clock tick units)
                ;   R5 = Low-order time value
                ;
                ;  Outputs:
                ;   Time value is printed without a trailing CR/LF.
                ;   CPUAL = Low-order CPU time in 0.1 second units.
                ;   CPUAH = High-order CPU time in 0.1 second units.
                ;
007410  010346  PRTTMV: MOV     R3,-(SP)
007412  010446          MOV     R4,-(SP)
007414  010546          MOV     R5,-(SP)
                ;
                ;  Convert time to seconds and fractions thereof
                ;
007416  066705  000000G         ADD     TK5VAL,R5       ;Round to nearest second
007422  005504                  ADC     R4              ;Propogate round carry
007424  016703  000000G         MOV     TK1SEC,R3       ;Get # clock ticks per second
007430  004767  176312          CALL    DIVIDE          ;Convert to seconds and fractions
                ;
                ;  We now have # seconds in R4-R5.
                ;  Convert to minutes and seconds.
                ;
007434  012703  000074          MOV     #60.,R3         ;60 seconds per minute
007440  004767  176302          CALL    DIVIDE          ;SPLIT INTO MINUTES AND SECONDS
007444  010046                  MOV     R0,-(SP)        ;SAVE # SECONDS
                ;
                ;  Split time into hours and minutes
                ;
007446  004767  176274          CALL    DIVIDE          ;SPLIT INTO # HOURS AND MINUTES
                ;
                ;  Print the complete time value
                ;
007452  004767  176672          CALL    PRTDC2          ;PRINT  # HOURS
007456  010005                  MOV     R0,R5           ;GET # MINUTES
007460                          .TTYOUT #':
007470  004767  176654          CALL    PRTDC2          ;PRINT # MINUTES
007474                          .TTYOUT #':
007504  012605                  MOV     (SP)+,R5        ;GET # SECONDS
007506  004767  176636          CALL    PRTDC2          ;PRINT # SECONDS
                ;
                ;  Now convert original clock-tick time into tenths of seconds
                ;
007512  011605                  MOV     (SP),R5         ;Recover original low-order value
007514  016604  000002          MOV     2(SP),R4        ;Recover original high-order value
                ;
                ;  Convert time value to seconds and fractions thereof
                ;
007520  016703  000000G         MOV     TK1SEC,R3       ;Get # clock ticks per second
007524  004767  176216          CALL    DIVIDE          ;Convert to seconds and fractions thereof
                ;
                ;  Now R4-R5 have # of seconds of time.
                ;  R0 has remainder in units of 1/50, 1/60, or 1/64 second units.
```

```
                                ;  Convert whole seconds value to 1/10 second units.
                                ;
007530  010046                        MOV     R0,-(SP)        ;Save fractional remainder
007532  012700  000012                MOV     #10.,R0         ;Multiply whole seconds value by 10.
007536  004767  176340                CALL    MUL32           ;Get approximate # 1/10
007542  010467  000000G               MOV     R4,CPUAH        ;Save high-order part
007546  010567  000000G               MOV     R5,CPUAL        ;Save low-order part
                                ;
                                ;  Now convert fractional reminder into 1/10 second units
                                ;
007552  012605                        MOV     (SP)+,R5        ;Get fractional remainder
007554  005004                        CLR     R4              ;Clear high-order for divide
007556  071467  000000G               DIV     TK1VAL,R4       ;Convert remainder to 1/10 sec units
007562  060467  000000G               ADD     R4,CPUAL        ;Add to low-order part
007566  005567  000000G               ADC     CPUAH           ;Propogate carry to high-order part
                                ;
                                ;  Finished
                                ;
007572  012605                        MOV     (SP)+,R5
007574  012604                        MOV     (SP)+,R4
007576  012603                        MOV     (SP)+,R3
007600  000207                        RETURN
```

```
                                .SBTTL  PRTTMD -- Print a time value with days
                        ;-----------------------------------------------------------------------
                        ;  PRTTMD is called to display a time value in the format DD HH:MM:SS
                        ;
                        ;  Inputs:
                        ;   R4 = High-order time value (0.1 second units)
                        ;   R5 = Low-order time value
                        ;
007602  010346          PRTTMD: MOV     R3,-(SP)
007604  010446                  MOV     R4,-(SP)
007606  010546                  MOV     R5,-(SP)
007610  062705  000005          ADD     #5.,R5          ;ROUND TO NEAREST SECOND
007614  005504                  ADC     R4
007616  012703  000012          MOV     #10.,R3         ;GET DIVISOR
007622  004767  176120          CALL    DIVIDE          ;SPLIT INTO SECONDS AND TENTHS
007626  012703  000074          MOV     #60.,R3
007632  004767  176110          CALL    DIVIDE          ;SPLIT INTO MINUTES AND SECONDS
007636  010046                  MOV     R0,-(SP)        ;SAVE # SECONDS
007640  004767  176102          CALL    DIVIDE          ;SPLIT INTO # HOURS AND MINUTES
007644  020527  000030          CMP     R5,#24.         ;MORE THAN 1 DAY?
007650  103415                  BLO     1$              ;BR IF NOT
007652  071427  000030          DIV     #24.,R4         ;GET DAYS AND HOURS WITHIN A DAY
007656  010546                  MOV     R5,-(SP)        ;SAVE HOURS (REMAINDER)
007660  010405                  MOV     R4,R5           ;GET NUMBER OF DAYS
007662  004767  176266          CALL    PRTDEC          ;PRINT NUMBER OF DAYS
007666  010046                  MOV     R0,-(SP)        ;SAVE MINUTES
007670                          .TTYOUT #40             ;PRINT A SPACE FOLLOWING THE DAY VALUE
007700  012600                  MOV     (SP)+,R0        ;RECOVER MINUTES
007702  012605                  MOV     (SP)+,R5        ;GET NUMBER OF HOURS WITHIN THE DAY
007704  004767  176440  1$:     CALL    PRTDC2          ;PRINT  # HOURS
007710  010005                  MOV     R0,R5           ;GET # MINUTES
007712                          .TTYOUT #':
007722  004767  176422          CALL    PRTDC2          ;PRINT # MINUTES
007726                          .TTYOUT #':
007736  012605                  MOV     (SP)+,R5        ;GET # SECONDS
007740  004767  176404          CALL    PRTDC2          ;PRINT # SECONDS
                        ;
                        ;  Finished
                        ;
007744  012605                  MOV     (SP)+,R5
007746  012604                  MOV     (SP)+,R4
007750  012603                  MOV     (SP)+,R3
007752  000207                  RETURN
```

```
                                        .SBTTL  PRTDAT -- Print the current date
                                ;-----------------------------------------------------------
                                ;  PRTDAT IS CALLED TO PRINT THE CURRENT DATE.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
007754  010046                  PRTDAT: MOV     R0,-(SP)
007756  010246                          MOV     R2,-(SP)
007760  010546                          MOV     R5,-(SP)
007762                                  .DATE                   ;GET CURRENT DATE
007770  010046                          MOV     R0,-(SP)        ;SAVE DATE
007772  006300                          ASL     R0              ;LEFT JUSTIFY VALUE
007774  012702  000005                  MOV     #5,R2           ;SHIFT OFF 5 BITS
010000  005005                          CLR     R5              ;INTO R5
010002  006100                  1$:     ROL     R0              ;SHIFT MONTH VALUE INTO R5
010004  006105                          ROL     R5
010006  077203                          SOB     R2,1$           ;LOOP IF MORE BITS TO SHIFT
010010  005305                          DEC     R5              ;GET MONTH VALUE IN RANGE 0-11
010012  010546                          MOV     R5,-(SP)        ;SAVE MONTH VALUE
                                ;  PRINT DAY NUMBER
010014  012702  000005                  MOV     #5,R2           ;GET 5 BITS OF DAY VALUE
010020  005005                          CLR     R5
010022  006100                  2$:     ROL     R0              ;SHIFT BITS INTO R5
010024  006105                          ROL     R5
010026  077203                          SOB     R2,2$
010030  004767  176120                  CALL    PRTDEC          ;PRINT DAY-OF-MONTH
010034                                  .TTYOUT #'-             ;PUT IN SEPARATOR
                                ;  PRINT MONTH NAME
010044  011605                          MOV     (SP),R5         ;GET MONTH INDEX
010046  006305                          ASL     R5              ;*2
010050  062605                          ADD     (SP)+,R5        ;*3
010052  062705  000000G                 ADD     #MONTAB,R5      ;POINT INTO ASCII MONTH TABLE
010056                                  .TTYOUT (R5)+           ;PRINT NAME OF MONTH
010064                                  .TTYOUT (R5)+
010072                                  .TTYOUT (R5)
010100                                  .TTYOUT #'-             ;PUT IN SEPARATOR
                                ;  PRINT YEAR
010110  012605                          MOV     (SP)+,R5        ;GET BACK ORIGINAL DATE VALUE
010112  042705  177740                  BIC     #<^C37>,R5      ;CLEAR ALL BUT YEAR FIELD
010116  062705  000110                  ADD     #72.,R5         ;YEAR # IS RELATIVE TO 1972
010122  004767  176222                  CALL    PRTDC2          ;PRINT YEAR VALUE
010126  012605                          MOV     (SP)+,R5
010130  012602                          MOV     (SP)+,R2
010132  012600                          MOV     (SP)+,R0
010134  000207                          RETURN
```

```
                                        .SBTTL  PRTTOD -- Print the time of day
                                ;-------------------------------------------------------------
                                ;  PRTTOD IS CALLED TO PRINT THE TIME OF DAY.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
010136  010046                  PRTTOD: MOV     R0,-(SP)
010140  010346                          MOV     R3,-(SP)
010142  010446                          MOV     R4,-(SP)
010144  010546                          MOV     R5,-(SP)
010146                                  .GTIM   #XAREA,#CPUAH   ;GET # CLOCK TICKS SINCE 00:00
010166  016704  000000G               MOV     CPUAH,R4        ;GET HIGH-ORDER VALUE
010172  016705  000000G               MOV     CPUAL,R5        ;GET LOW-ORDER VALUE
010176  016703  000000G               MOV     TK1SEC,R3       ;Get # clock ticks per second
010202  004767  175540                CALL    DIVIDE          ;Get # seconds past midnight
010206  012703  000074                MOV     #60.,R3         ;DIVIDE BY 60 TO SPLIT INTO SEC & MIN
010212  004767  175530                CALL    DIVIDE
010216  010046                       MOV     R0,-(SP)        ;SAVE # SECONDS INTO MINUTE
010220  004767  175522                CALL    DIVIDE          ;GET # MINUTES & # HOURS
010224  004767  176120                CALL    PRTDC2          ;PRINT HOUR VALUE
010230  010005                       MOV     R0,R5           ;GET # MINUTE VALUE
010232                                  .TTYOUT #':             ;PUT IN COLON SEPARATOR
010242  004767  176102                CALL    PRTDC2          ;PRINT # MINUTES
010246                                  .TTYOUT #':             ;ANOTHER COLON
010256  012605                       MOV     (SP)+,R5        ;GET # SECONDS
010260  004767  176064                CALL    PRTDC2          ;PRINT # SECONDS
010264  012605                       MOV     (SP)+,R5
010266  012604                       MOV     (SP)+,R4
010270  012603                       MOV     (SP)+,R3
010272  012600                       MOV     (SP)+,R0
010274  000207                       RETURN

                                        .SBTTL  DATIM  -- Print date and time
                                ;-------------------------------------------------------------
                                ;  DATTIM IS CALLED TO PRINT THE CURRENT DATE AND TIME.
                                ;  IF NO DATE HAS BEEN ENTERED DATTIM RETURNS WITHOUT DOING
                                ;  ANYTHING.  ALL REGISTERS ARE PRESERVED.
                                ;
010276  010046                  DATTIM: MOV     R0,-(SP)
010300                                  .DATE                   ;GET CURRENT DATE
010306  005700                       TST     R0              ;WAS DATE ENTERED BY OPERATOR?
010310  001412                       BEQ     1$              ;BR IF NOT
010312  004767  177436                CALL    PRTDAT          ;PRINT CURRENT DATE
010316                                  .PRINT  #SPACE2         ;PRINT 2 SPACES
010324  004767  177606                CALL    PRTTOD          ;PRINT TIME OF DAY
010330                                  .PRINT  #CRLF           ;PRINT CR-LF
010336  012600                       1$:     MOV     (SP)+,R0
010340  000207                       RETURN
```

```
                                        .SBTTL  SEARCH -- Search keyword list
                                ;-----------------------------------------------------------
                                ;  SEARCH is called to compare a command or option
                                ;  keyword with a table of names.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of the command keyword.
                                ;   R4 = Points to the start of the command name table which is
                                ;        built by use of the TBLDEF, CMDDEF, and TBLEND macros.
                                ;
                                ;  Outputs:
                                ;   R3 = Points to 1st non-blank character after keyword.
                                ;   C-flag reset ==> Command successfully identified.
                                ;      R4 = Pointer to 2nd word of command table entry.
                                ;   C-flag set ==> Command not successfully identified.
                                ;      R4=0         ==> Command not recognized.
                                ;      R4=non-zero ==> Command is ambigious.
                                ;
010342  010046                  SEARCH: MOV     R0,-(SP)
010344  010146                          MOV     R1,-(SP)
010346  010246                          MOV     R2,-(SP)
010350  010546                          MOV     R5,-(SP)
                                ;
                                ;  Skip leading spaces, tabs, and form-feeds.
                                ;
010352  112300                  14$:    MOVB    (R3)+,R0        ;Get next character
010354  120027  000040                  CMPB    R0,#' '         ;Is this a space?
010360  001774                          BEQ     14$             ;Skip leading spaces
010362  120027  000011                  CMPB    R0,#TAB         ;Skip leading tabs
010366  001771                          BEQ     14$
010370  120027  000014                  CMPB    R0,#FF          ;Skip leading form-feeds
010374  001766                          BEQ     14$
010376  005303                          DEC     R3              ;Point to first non-blank character
                                ;
                                ;  Move keyword to a holding buffer and convert lower-case letters
                                ;  to upper case
                                ;
010400  012705  000000G                 MOV     #KEYBUF,R5      ;Point to keyword holding buffer
010404  112300                  15$:    MOVB    (R3)+,R0        ;Get next character from command string
010406  120027  000141                  CMPB    R0,#141         ;Is this a lower-case letter?
010412  103405                          BLO     16$             ;Br if not
010414  120027  000172                  CMPB    R0,#172
010420  101035                          BHI     17$
010422  162700  000040                  SUB     #40,R0          ;Convert lower-case to upper-case
010426  120027  000132          16$:    CMPB    R0,#'Z          ;Is this character a letter?
010432  101020                          BHI     23$             ;Br if not
010434  120027  000101                  CMPB    R0,#'A
010440  103020                          BHIS    18$             ;Br if it is a letter
010442  120027  000071                  CMPB    R0,#'9          ;Is character a digit?
                                ; NEXT 5 LINES REPLACE 2 FOLLOWING TO REJECT COMMAND .SY:FILNAM
                                ; AND LOOK FOR COMMAND FILE INSTEAD OF DOING SYSTAT COMMAND
                                ;**     BHI     17$             ;Br if delimiter
                                ;**     CMPB    R0,#'0
010446  101404                          BLOS    24$             ;BRANCH IF MAYBE                ;**
010450  120027  000072                  CMPB    R0,#':          ;CHAR RANGE : TO @, IS IT : ?   ;**
010454  001412                          BEQ     18$             ;INCLUDE IF SO                  ;**
010456  000416                          BR      17$             ;ELSE DELIMITER                 ;**
```

```
     58 010460  120027  000060          24$:    CMPB    R0,#'0          ;See if in range 0 - 9          ;**
     59 010464  103006                          BHIS    18$             ;Br if digit
     60 010466  120027  000044                  CMPB    R0,#'$          ;Allow "$" in command names
     61 010472  001403                          BEQ     18$
     62 010474  120027  000137          23$:    CMPB    R0,#'_          ;Also allow underscore
     63 010500  001005                          BNE     17$             ;Br if character is a delimiter
     64 010502  020527  177777G         18$:    CMP     R5,#KEYEND-1    ;Have we reached end of keyword buffer?
     65 010506  103336                          BHIS    15$             ;Br if yes
     66 010510  110025                          MOVB    R0,(R5)+        ;Move character to buffer
     67 010512  000734                          BR      15$
     68                                 ;
     69                                 ;  Reached end of keyword
     70                                 ;
     71 010514  105015                  17$:    CLRB    (R5)            ;Put null at end of keyword name
     72                                 ;
     73                                 ;  Skip up to start of next field after keyword
     74                                 ;
     75 010516  005303                          DEC     R3              ;Point to delimiter
     76 010520  122327  000040          22$:    CMPB    (R3)+,#' '      ;Skip spaces following keyword
     77 010524  001775                          BEQ     22$
     78 010526  005303                          DEC     R3              ;Point to 1st non-blank character
     79 010530  010346                          MOV     R3,-(SP)        ;Save delimiter pointer on stack for later
     80 010532  105767  000000G               TSTB    KEYBUF          ;Did we have a null keyword?
     81 010536  001433                          BEQ     21$             ;Br if yes
     82                                 ;
     83                                 ;  The keyword has been converted to upper-case and is stored in KEYBUF.
     84                                 ;  We are now ready to begin comparing the keyword.
     85                                 ;
     86 010540  012402                          MOV     (R4)+,R2        ;Get # of bytes per table entry
     87 010542  012405                  5$:     MOV     (R4)+,R5        ;Point to asciz name string
     88 010544  001433                          BEQ     20$             ;Br if end of table hit
     89 010546  012703  000000G               MOV     #KEYBUF,R3      ;Point to our keyword
     90 010552  005001                          CLR     R1              ;Say no star seen yet
     91                                 ;
     92                                 ;  Begin to compare command string pointed to by R3 with
     93                                 ;  keyword in table entry pointed to by R5
     94                                 ;
     95 010554  112300                  6$:     MOVB    (R3)+,R0        ;Get a char from the keyword
     96 010556  001414                          BEQ     1$              ;Br if hit end of keyword
     97 010560  105715                  2$:     TSTB    (R5)            ;Hit end of name in table?
     98 010562  001410                          BEQ     4$              ;Br if yes -- no match
     99 010564  121527  000052                  CMPB    (R5),#'*        ;Is next char a star?
    100 010570  001003                          BNE     19$             ;Br if not
    101 010572  005201                          INC     R1              ;Remember star seen
    102 010574  005205                          INC     R5              ;Point beyond star
    103 010576  000770                          BR      2$              ;Continue comparison
    104 010600  122500                  19$:    CMPB    (R5)+,R0        ;Do names match
    105 010602  001764                          BEQ     6$              ;Yes -- keep checking
    106                                 ;
    107                                 ;  Names do not match
    108                                 ;  Compare keyword with next entry in table
    109                                 ;
    110 010604  060204                  4$:     ADD     R2,R4           ;Point to next table entry
    111 010606  000755                          BR      5$              ;Go check it
    112                                 ;
    113                                 ;  Reached end of keyword. All chars match so far.
    114                                 ;  If we are also at end of name in table or if we have seen a star,
```

```
                                ;  then we have a match.  Otherwise we may have an ambigious keyword
                                ;  or we may have to continue searching.
                                ;  This algorithm depends on shortest legitimate match occurring first
                                ;  in table. Otherwise, will get spurious ambiguities.
                                ;
010610  105715                  1$:     TSTB    (R5)            ;Are we at the end of the table entry also?
010612  001413                          BEQ     7$              ;yes, we have an exact match
010614  121527  000052                  CMPB    (R5),#'*        ;Is next table char "*"?
010620  001410                          BEQ     7$              ;If yes, then this is acceptable abbrev
010622  005701                          TST     R1              ;Have we already seen "*"?
010624  001006                          BNE     7$              ;If yes then we have an acceptable abbrev
                                ;
                                ;  We have an ambiguous keyword
                                ;
010626  010504                  21$:    MOV     R5,R4           ;Make R4 non-zero to signal ambigious case
010630  000261                          SEC                     ;Signal failure on return
010632  000404                          BR      10$             ;Finished
                                ;
                                ;  Cannot find keyword in table
                                ;
010634  005004                  20$:    CLR     R4              ;Signal unrecognizable command
010636  000261                          SEC                     ;Signal failure on return
010640  000401                          BR      10$
                                ;
                                ;  We found keyword in table
                                ;
010642  000241                  7$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
010644  012603                  10$:    MOV     (SP)+,R3        ;Recover command string pointer
010646  012605                          MOV     (SP)+,R5
010650  012602                          MOV     (SP)+,R2
010652  012601                          MOV     (SP)+,R1
010654  012600                          MOV     (SP)+,R0
010656  000207                          RETURN
```

```
                                        .SBTTL  FPRINT -- Print fatal error message
                                ;-------------------------------------------------------
                                ;  FPRINT IS CALLED TO PRINT A FATAL ERROR MESSAGE FROM
                                ;  WITHIN TSKMON.  USE THE FERROR MACRO TO INVOKE FPRINT.
                                ;
010660                          FPRINT: .PRINT  #KMFTXT         ;PRINT ERROR MESSAGE HEADER
010666                                  .PRINT  R5              ;NOW PRINT ERROR MESSAGE
010672  004767  000044                  CALL    KMNERR          ;SEE IF WE SHOULD ABORT COMMAND FILES
010676  000207                          RETURN

                                        .SBTTL  PRTWRN -- Print warning message
                                ;-------------------------------------------------------------
                                ;  PRTWRN is called to print a KMON warning message.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to asciz text string.
                                ;
010700                          PRTWRN: .PRINT  #WRNHED         ;Print warning heading
010706                                  .PRINT  R5              ;Print warning message
010712  152767  000000G 000000G        BISB    #SC$WRN,INDERR  ;Set warning severity for IND
010720  000207                          RETURN

                                        .SBTTL  FKILL  -- Print error message and abort
                                ;-------------------------------------------------------
                                ;  FKILL IS JUMPED TO TO PRINT A FATAL ERROR MESSAGE,
                                ;  RESET THE STACK AND JUMP TO RDCMD.
                                ;  USE THE FABORT MACRO TO CALL FKILL.
                                ;
010722  004767  177732          FKILL:  CALL    FPRINT          ;PRINT FATAL ERROR MESSAGE
010726  012706  000000G                 MOV     #KMSTK,SP       ;CLEAN OFF STACK
010732  004767  000004                  CALL    KMNERR          ;SEE IF WE SHOULD ABORT COMMAND FILES
010736  000167  000000G                 JMP     RDCMD           ;READ NEXT COMMAND

                                        .SBTTL  KMNERR -- Abort command files on KMON error
                                ;-------------------------------------------------------------
                                ;  KMNERR is called when a command error is detected by TSKMON.
                                ;  If error abort severity is set to abort on errors or warnings,
                                ;  all currently open command files are aborted.  Otherwise execution
                                ;  continues
                                ;
010742  010146                  KMNERR: MOV     R1,-(SP)
010744  152767  000000G 000000G        BISB    #SC$SEV,INDERR  ;SET ERROR SEVERITY FOR IND
010752  116701  000000G                MOVB    CORUSR,R1       ;GET CURRENT JOB INDEX NUMBER
010756  122767  000000G 000000G        CMPB    #SC$SEV,ERRSEV  ;ABORT ON ERRORS?
010764  103410                          BLO     1$              ;BR IF NOT
010766  004767  172312                  CALL    ABRTCF          ;ABORT ALL OPEN COMMAND FILES
010772  032761  000000G 000000G        BIT     #$INDAB,LSW7(R1);SHOULD WE ABORT IND FILES ON ERRORS?
011000  001402                          BEQ     1$              ;BR IF NOT
011002  004767  172412                  CALL    INDABT          ;Abort execution of IND & nested command files
011006  012601                  1$:     MOV     (SP)+,R1
011010  000207                          RETURN
```

```
                                        .SBTTL  ACRFN  -- Accrue a file name
                                ;-------------------------------------------------------
                                ;  ACRFN IS CALLED TO ACCRUE A FILE NAME IN THE STANDARD
                                ;  FORM "DV:NAME.EXT".
                                ;  WHEN CALLED, R3 MUST POINT TO THE START OF THE NAME
                                ;  AND R5 MUST POINT TO A 2 WORD BLOCK IN RAD50 FORM
                                ;  CONTAINING THE DEFAULT DEVICE NAME AND DEFAULT EXTENSION.
                                ;  ON RETURN, THE FILE SPEC IS IN RAD50 FORM IN THE 4 WORD
                                ;  BLOCK "FILNAM" AND R3 POINTS PAST THE END OF THE NAME.
                                ;  R3 AND R5 ARE ALTERED, ALL OTHER REGISTERS ARE PRESERVED.
                                ;  If an error occurs while accruing the file name, the
                                ;  C-flag is set on return.
                                ;
011012  012567  000000G        ACRFN:  MOV     (R5)+,FILNAM    ;SET DEFAULT DEVICE
011016  011567  000006G                MOV     (R5),FILNAM+6   ;SET DEFAULT EXTENSION
011022  005067  000010G                CLR     FILNAM+8.       ;NO FILE SIZE
                                ;  ACCRUE NEXT FIELD OF NAME
011026  004767  174264         2$:    CALL    GTRD50          ;ACCRUE NAME IN RAD50 FORMAT
                                ;  SEE IF THIS IS THE DEVICE OR FILE NAME
011032  121327  000072                CMPB    (R3),#':        ;WAS THAT THE DEVICE NAME
011036  001011                       BNE     1$              ;BR IF NOT.  MUST BE FILE NAME
                                ;  WE HAVE JUST ACCRUED THE DEVICE NAME
011040  016767  000000G 000000G       MOV     R50BUF,FILNAM   ;SET DEVICE NAME
011046  005203                       INC     R3              ;POINT PAST COLON
011050  111300                       MOVB    (R3),R0         ;GET 1ST CHAR OF FILE NAME
011052  004767  003552               CALL    CHKDLM          ;SEE IF IT IS A DELIMITER
011056  103363                       BCC     2$              ;BR IF NOT DELIMITER
011060  000431                       BR      3$              ;INVALID FILE NAME
                                ;  WE JUST GOT THE FILE NAME
011062  016767  000000G 000002G 1$:    MOV     R50BUF,FILNAM+2 ;STORE THE FILE NAME
011070  001425                       BEQ     3$              ;MUST NOT BE NULL
011072  016767  000002G 000004G       MOV     R50BUF+2,FILNAM+4
011100  026727  000002G 132500       CMP     FILNAM+2,#132500;WAS 1ST PART OF NAME "*"?
011106  001003                       BNE     5$              ;BR IF NOT
011110  012767  132500  000004G       MOV     #132500,FILNAM+4;IF YES THEN MAKE 2ND PART BE "*" TOO
                                ;  SEE IF AN EXTENSION WAS SPECIFIED
011116  121327  000056         5$:    CMPB    (R3),#'.        ;IS EXTENSION PRESENT?
011122  001006                       BNE     4$              ;BR IF NOT
011124  005203                       INC     R3              ;SKIP OVER PERIOD
011126  004767  174164               CALL    GTRD50          ;ACCRUE THE EXTENSION
011132  016767  000000G 000006G       MOV     R50BUF,FILNAM+6 ;STORE THE EXTENSION
011140  000241                 4$:    CLC                     ;SIGNAL SUCCESS ON RETURN
011142  000207                       RETURN
                                ;
                                ;  ERROR -- INVALID FILE NAME
011144                         3$:    FERR    #BDFNAM
011160  000261                       SEC                     ;SIGNAL ERROR ON RETURN
011162  000207                       RETURN
```

```
                                        .SBTTL  ACRFIL -- Accrue full file specification
                                ;-----------------------------------------------------------------
                                ;  ACRFIL is called to accrue a full file specification of the form
                                ;       dev:file.ext[size]
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to file name which can be terminated by a null,
                                ;        comma, blank, or equal sign.
                                ;   R4 = Pointer to word containing default file extension.
                                ;   R5 = 0==>Input file, 1==>Output file.
                                ;
                                ;  Outputs:
                                ;   R3 = Pointer to delimiter at end of file spec.
                                ;   FILNAM = 5 word block containing file spec in RAD50 form.
                                ;   C-flag set on return if invalid file spec.
                                ;
011164  010146                  ACRFIL: MOV     R1,-(SP)
011166  010246                          MOV     R2,-(SP)
011170  010446                          MOV     R4,-(SP)
011172  010546                          MOV     R5,-(SP)
                                ;
                                ;  Skip over leading spaces in front of the file spec
                                ;
011174  122327  000040          6$:     CMPB    (R3)+,#' ;SKIP LEADING SPACES
011200  001775                          BEQ     6$
011202  005303                          DEC     R3              ;POINT TO 1ST NON-BLANK CHAR
                                ;
                                ;  Move file spec to a holding buffer
                                ;
011204  012702  000000G                 MOV     #BLK0,R2        ;POINT TO HOLDING BUFFER
011210  112300                  1$:     MOVB    (R3)+,R0        ;GET NEXT CHAR FROM FILE NAME
011212  001416                          BEQ     2$              ;BR IF HIT END
011214  120027  000057                  CMPB    R0,#'/          ;TERMINATE ON SLASH
011220  001413                          BEQ     2$
011222  120027  000054                  CMPB    R0,#',          ;TERMINATE ON COMMA
011226  001410                          BEQ     2$
011230  120027  000040                  CMPB    R0,#' '         ;TERMINATE ON BLANK
011234  001405                          BEQ     2$
011236  120027  000075                  CMPB    R0,#'=          ;TERMINATE ON EQUAL SIGN
011242  001402                          BEQ     2$
011244  110022                          MOVB    R0,(R2)+        ;MOVE CHAR TO HOLDING BUFFER
011246  000760                          BR      1$              ;GO GET NEXT CHAR
                                ;
                                ;  Set up the file-spec as an input or output file.
                                ;
011250  005705                  2$:     TST     R5              ;INPUT OR OUTPUT TYPE FILE?
011252  001407                          BEQ     4$              ;BR IF INPUT FILE
011254  112722  000075                  MOVB    #'=,(R2)+       ;MAKE SPEC LOOK LIKE OUTPUT FILE
011260  162704  000002                  SUB     #2,R4           ;CORRECT DEFAULT EXTENSION POINTER FOR OUTPUT
011264  012705  000000G                 MOV     #KCSIBF,R5      ;CSISPC WILL PUT RESULT HERE
011270  000402                          BR      5$
011272  012705  000360G          4$:     MOV     #KCSIBF+30.,R5  ;CSISPC WILL PUT RESULT HERE
011276  105012                  5$:     CLRB    (R2)            ;PUT IN ASCIZ NULL AT END OF NAME
011300  005303                          DEC     R3              ;POINT BACK TO DELIMITER
                                ;
                                ;  Use .CSISPC to parse the file spec
                                ;
```

```
     58 011302  010601                          MOV     SP,R1               ;SAVE SP ACROSS .CSISPC
     59 011304                                  .CSISPC #KCSIBF,R4,#BLKO ;PARSE THE FILE SPEC
     60 011320  010106                          MOV     R1,SP               ;RESTORE SP (IGNORE SWITCH INFO FROM .CSISPC)
     61 011322  103410                          BCS     9$                  ;BR IF INVALID
     62                                 ;
     63                                 ;  Move file spec to result area
     64                                 ;
     65 011324  012700  000000G                 MOV     #FILNAM,R0          ;POINT TO RESULT AREA
     66 011330  012520                          MOV     (R5)+,(R0)+
     67 011332  012520                          MOV     (R5)+,(R0)+
     68 011334  012520                          MOV     (R5)+,(R0)+
     69 011336  012520                          MOV     (R5)+,(R0)+
     70 011340  011510                          MOV     (R5),(R0)
     71                                 ;
     72                                 ;  Finished
     73                                 ;
     74 011342  000241                          CLC                         ;SIGNAL SUCCESS ON RETURN
     75 011344  012605                  9$:     MOV     (SP)+,R5
     76 011346  012604                          MOV     (SP)+,R4
     77 011350  012602                          MOV     (SP)+,R2
     78 011352  012601                          MOV     (SP)+,R1
     79 011354  000207                          RETURN
```

```
 1                                              .SBTTL  DMTALL -- Dismount and deallocate all devices
 2                                      ;-----------------------------------------------------------------------
 3                                      ;  Dismount and deallocate all devices that are mounted by the current job.
 4                                      ;
 5 011356  010246                       DMTALL: MOV     R2,-(SP)
 6 011360  010346                               MOV     R3,-(SP)
 7 011362  010546                               MOV     R5,-(SP)
 8                                      ;
 9                                      ;  Deallocate all devices allocated by this job
10                                      ;
11 011364  005067  000000G                     CLR     ALCDEV          ;Say to deallocate all devices for this job
12 011370  012700  000000G                     MOV     #DLCEMT,R0      ;Point to EMT argument block
13 011374  104375                              EMT     375             ;Deallocate all devices for this job
14                                      ;
15                                      ;  Search through mount table looking for devices mounted by our job
16                                      ;
17 011376  016705  000000G                     MOV     CSHDEV,R5       ;Point to table of mounted devices
18 011402  010500                       1$:    MOV     R5,R0           ;Get address of mount entry
19 011404  004767  001070                      CALL    CDGET           ;Read mount entry into CDBUF
20 011410  005767  000001C                     TST     CDBUF+CD$DVU    ;Is this table entry in use?
21 011414  001421                              BEQ     2$              ;Br if not
22 011416  004767  000152                      CALL    CDJFLG          ;Get mount-flag for our job
23 011422  130312                              BITB    R3,(R2)         ;Is this device mounted by us?
24 011424  001415                              BEQ     2$              ;Br if not
25                                      ;
26                                      ;  Found a device that is mounted by us,
27                                      ;  reset mount flag for our job and see if device is mounted
28                                      ;  by any other jobs.
29                                      ;
30 011426  140312                              BICB    R3,(R2)         ;Reset mount flag for our job
31 011430  010500                              MOV     R5,R0           ;Get address where block is to be stored
32 011432  004767  001062                      CALL    CDPUT           ;Write updated block back into kernel data
33 011436  012703  000000C                     MOV     #CDBUF+CD$JOB,R3;Get address of mount-flag table
34 011442  012702  000000G                     MOV     #CD$$UB,R2      ;Get # bytes in mount-flag table
35 011446  105723                       3$:    TSTB    (R3)+           ;Any other jobs using this device?
36 011450  001003                              BNE     2$              ;Br if yes
37 011452  077203                              SOB     R2,3$
38                                      ;
39                                      ;  No other jobs have this device mounted.
40                                      ;  Free the mount table entry for this device and remove
41                                      ;  any file entries from cache.
42                                      ;
43 011454  004767  000022                      CALL    DMTSUB          ;Dismount this device and all of its files
44                                      ;
45                                      ;  Check next entry in mount table
46                                      ;
47 011460  062705  000000G              2$:    ADD     #CD$$SZ,R5      ;Point to next entry in mount table
48 011464  020567  000000G                     CMP     R5,CSHDVN       ;Any more entries?
49 011470  103744                              BLO     1$              ;Loop if yes
50                                      ;
51                                      ;  Finished
52                                      ;
53 011472  012605                              MOV     (SP)+,R5
54 011474  012603                              MOV     (SP)+,R3
55 011476  012602                              MOV     (SP)+,R2
56 011500  000207                              RETURN
```

```
                                        .SBTTL  DMTSUB -- Remove a device from directory cache
                                ;-----------------------------------------------------------------------
                                ;  DMTSUB is called to remove from the mount table a specific device
                                ;  and to remove from the directory cache any files associated with
                                ;  the device.
                                ;
                                ;  Inputs:
                                ;   CDBUF contains mount table entry for device to be dismounted.
                                ;
011502  010346                 DMTSUB: MOV     R3,-(SP)
                                ;
                                ;  See if this device is a logical disk mounted by us
                                ;
011504  016700  000000G                 MOV     R50LD0,R0       ;Get LD0 as initial device name
011510  005003                          CLR     R3              ;Init LD table index
011512  026763  000001C 000000G 1$:     CMP     CDBUF+CD$DVU,LDPDEV(R3) ;Is this LD on same physical device?
011520  001004                          BNE     2$              ;Br if not
011522  026763  000001C 000000G         CMP     CDBUF+CD$BAS,LDBASE(R3) ;Same starting block numbers?
011530  001412                          BEQ     3$              ;Br if yes -- Found logical disk that matches
011532  005200                  2$:     INC     R0              ;Advance logical disk name
011534  062703  000002                  ADD     #2,R3           ;Advance LD table index
011540  020327  000016                  CMP     R3,#14.         ;Checked all logical disks?
011544  101762                          BLOS    1$              ;Br if not
                                ;
                                ;  This is not a logical disk.
                                ;  Get physical device name.
                                ;
011546  016700  000001C                 MOV     CDBUF+CD$DVU,R0 ;Get physical device and unit number
011552  004767  001724                  CALL    CVDVNM          ;Convert to device name
                                ;
                                ;  Do dismount EMT
                                ;
011556  010067  000000G         3$:     MOV     R0,MNTDEV       ;Set name of device to dismount
011562  012700  000000G                 MOV     #DMTARG,R0      ;Point to EMT argument block
011566  104375                          EMT     375             ;Dismount the device
                                ;
                                ;  Finished
                                ;
011570  012603                          MOV     (SP)+,R3
011572  000207                          RETURN
```

```
                                        .SBTTL  CDJFLG -- Get user-flag for cached device entry
                                ;-----------------------------------------------------------------------
                                ;  CDJFLG is called to locate within a cached-device table entry the
                                ;  specific mount-flag that corresponds to the current job.
                                ;
                                ;  Inputs:
                                ;   CORUSR = Current job index number.
                                ;
                                ;  Outputs:
                                ;   R2 = Address of byte that contains mount-flag.
                                ;   R3 = Mount-flag bit positioned correctly within byte.
                                ;
011574  116703  0000000G       CDJFLG: MOVB    CORUSR,R3       ;Get current job index number
011600  006203                          ASR     R3              ;Convert to index by 1
011602  005303                          DEC     R3              ;Make base job number 0
011604  005002                          CLR     R2              ;Clear for divide
011606  071227  000010                  DIV     #8.,R2          ;Divide by 8 jobs per byte
011612  062702  000000C                 ADD     #CDBUF+CD$JOB,R2;Get address of byte within entry in CDBUF
011616  012700  000001                  MOV     #1,R0           ;Get a mount flag
011622  072003                          ASH     R3,R0           ;Position flag according to job number
011624  010003                          MOV     R0,R3           ;Return flag in R3
011626  000207                          RETURN
```

```
                                        .SBTTL  CHKDEV -- See if requested device is legal
                                ;-----------------------------------------------------------------------
                                ;  CHKDEV is called to convert a device name into the corresponding
                                ;  device index number and unit number.
                                ;
                                ;  Inputs:
                                ;   R5 = Device-unit specification in rad50 form (e.g., "RK1")
                                ;
                                ;  Outputs:
                                ;   R0 = Unit number of device
                                ;   R4 = Index into device tables
                                ;   C-flag set on return if the device is not recognized.
                                ;
011630  010146                  CHKDEV: MOV     R1,-(SP)
011632  010246                          MOV     R2,-(SP)
                                ;
                                ;  If this name has been assigned, substitute physical device name for
                                ;  logical device name.
                                ;
011634  010501                          MOV     R5,R1           ;GET LOGICAL DEVICE NAME
011636  010500                          MOV     R5,R0
011640  004767  001776                  CALL    ASNSRC          ;SEE IF DEVICE NAME HAS BEEN ASSIGNED
011644  103402                          BCS     11$             ;BR IF NOT ASSIGNED
011646  016201  000004                  MOV     4(R2),R1        ;REPLACE LOGICAL DEVICE NAME WITH PHYSICAL
                                ;
                                ;  Get device name and split off unit number.
                                ;
011652  005000                  11$:    CLR     R0              ;SET FOR DIVIDE
011654  071027  000050                  DIV     #50,R0          ;SPLIT OFF LOW-ORDER RAD50 CHARACTER
011660  012702  177777                  MOV     #-1,R2          ;ASSUME NO UNIT NUMBER SPECIFIED
011664  005701                          TST     R1              ;WAS A UNIT NUMBER SPECIFIED?
011666  001406                          BEQ     6$              ;BR IF NOT
011670  162701  000036                  SUB     #36,R1          ;CONVERT RAD50 DIGIT TO BINARY VALUE
011674  010102                          MOV     R1,R2           ;GET BINARY VALUE OF UNIT NUMBER
011676  020227  000007                  CMP     R2,#7           ;RESTRICT UNIT NUMBER TO RANGE 0-7
011702  101027                          BHI     8$              ;BR IF INVALID UNIT NUMBER
011704  070027  000050          6$:     MUL     #50,R0          ;GET DEVICE NAME WITHOUT UNIT NUMBER
                                ;
                                ;  The rad50 device name less unit number is now in R1.
                                ;  R2 has the binary value of the unit number or -1 if no unit number
                                ;  was specified.
                                ;
                                ;  Translate "SY:" and "DK:" to physical device.
                                ;
011710  020167  000000G                  CMP     R1,R50DK        ;IS DEVICE NAME "DK"?
011714  001403                          BEQ     2$              ;BR IF YES
011716  020167  000000G                  CMP     R1,R50SY        ;IS DEVICE NAME "SY"?
011722  001007                          BNE     3$              ;BR IF NOT
011724  016704  000000G          2$:     MOV     SYINDX,R4       ;GET SY DEVICE INDEX NUMBER
011730  005702                          TST     R2              ;DID USER SPECIFY A UNIT NUMBER?
011732  002015                          BGE     7$              ;BR IF YES
011734  116702  000001G                  MOVB    SYUNIT+1,R2     ;GET SYSTEM DEVICE UNIT NUMBER
011740  000412                          BR      7$
                                ;
                                ;  Look up device name in permanent device name table.
                                ;
011742  016704  000000G          3$:     MOV     NUMDEV,R4       ;GET INDEX NUMBER OF LAST DEVICE
```

```
     58 011746  020164  000000G          5$:     CMP     R1,PNAME(R4)    ;SEARCH FOR NAME IN TABLE
     59 011752  001405                           BEQ     7$              ;BR IF FOUND
     60 011754  162704  000002                   SUB     #2,R4           ;TRY NEXT ENTRY
     61 011760  002372                           BGE     5$              ;LOOP IF MORE TO CHECK
     62                                  ;
     63                                  ;  Invalid device name
     64                                  ;
     65 011762  000261                   8$:     SEC                     ;SIGNAL INVALID DEVICE NAME
     66 011764  000414                           BR      10$             ;RETURN
     67                                  ;
     68                                  ;  Found device name.  Translate no unit number into # 0.
     69                                  ;
     70 011766  010200                   7$:     MOV     R2,R0           ;GET UNIT NUMBER VALUE
     71 011770  002001                           BGE     1$              ;BR IF UNIT NUMBER WAS SPECIFIED
     72 011772  005000                           CLR     R0              ;SAY UNIT # = 0 IF NONE SPECIFIED
     73                                  ;
     74                                  ;  If the device is a logical disk (LDn), check to make sure the
     75                                  ;  particular unit is mapped to a file
     76                                  ;
     77 011774  020467  000000G          1$:     CMP     R4,LDDEVX       ;IS THIS A LOGICAL DISK?
     78 012000  001005                           BNE     9$              ;BR IF NOT
     79 012002  010002                           MOV     R0,R2           ;GET UNIT NUMBER
     80 012004  006302                           ASL     R2              ;CONVERT TO WORD TABLE INDEX
     81 012006  005762  000000G                  TST     LDPDEV(R2)      ;IS UNIT MAPPED TO A FILE?
     82 012012  001763                           BEQ     8$              ;BR IF NOT
     83 012014  000241                   9$:     CLC                     ;SIGNAL GOOD RETURN
     84                                  ;
     85                                  ;  Finished -- Return
     86                                  ;
     87 012016  012602                   10$:    MOV     (SP)+,R2
     88 012020  012601                           MOV     (SP)+,R1
     89 012022  000207                           RETURN
```

```
 1                                              .SBTTL  CHKMNT -- See if device is mounted
 2                                      ;-----------------------------------------------------------------
 3                                      ;  CHKMNT is called to determine if a specified device is mounted
 4                                      ;  by any users.
 5                                      ;
 6                                      ;  Inputs:
 7                                      ;   R5 = Rad50 device-unit name.
 8                                      ;
 9                                      ;  Outputs:
10                                      ;   C-flag cleared ==> Device is mounted.
11                                      ;   C-flag set ==> Device is not mounted.
12                                      ;   CDBUF contains mount table entry for device.
13                                      ;
14 012024  010346                      CHKMNT: MOV     R3,-(SP)
15 012026  010446                              MOV     R4,-(SP)
16 012030  010546                              MOV     R5,-(SP)
17                                      ;
18                                      ;  Convert device name into device index number and unit number
19                                      ;
20 012032  004767  177572                      CALL    CHKDEV          ;Convert device name to device # and unit #
21 012036  103437                              BCS     9$              ;Br if invalid device name
22                                      ;
23                                      ;  If this device is a logical disk, get base block # and physical dev info
24                                      ;
25 012040  020467  000000G                     CMP     R4,LDDEVX       ;Is this a logical disk?
26 012044  001006                              BNE     1$              ;Br if not
27 012046  006300                              ASL     R0              ;Convert unit # to word table index
28 012050  016004  000000G                     MOV     LDPDEV(R0),R4   ;Get physical disk device # and unit #
29 012054  016003  000000G                     MOV     LDBASE(R0),R3   ;Get base block # of logical disk file
30 012060  000403                              BR      2$
31 012062  000300                      1$:     SWAB    R0              ;Put unit # in high-order byte
32 012064  050004                              BIS     R0,R4           ;Combine unit # and device #
33 012066  005003                              CLR     R3              ;Base block # = 0 if not logical disk
34                                      ;
35                                      ;  Search mount table for this device
36                                      ;
37 012070  016705  000000G              2$:     MOV     CSHDEV,R5       ;Point to start of mount table
38 012074  010500                       3$:     MOV     R5,R0           ;Get addr of mount table entry
39 012076  004767  000376                      CALL    CDGET           ;Copy mount table entry into CDBUF
40 012102  020467  000001C                     CMP     R4,CDBUF+CD$DVU ;Is entry for this device?
41 012106  001003                              BNE     4$              ;Br if not
42 012110  020367  000001C                     CMP     R3,CDBUF+CD$BAS ;Check base block numbers too
43 012114  001407                              BEQ     5$              ;Br if found entry for this device
44 012116  062705  000000G              4$:     ADD     #CD$$SZ,R5      ;Point to next mount table entry
45 012122  020567  000000G                     CMP     R5,CSHDVN       ;Hit end of table?
46 012126  103762                              BLO     3$              ;Loop if not
47 012130  000261                       6$:     SEC                     ;Device is not mounted at all
48 012132  000401                              BR      9$
49                                      ;
50                                      ;  We found entry for this device in mount table.
51                                      ;
52 012134  000241                       5$:     CLC                     ;Signal that device is mounted
53                                      ;
54                                      ;  Finished
55                                      ;
56 012136  012605                       9$:     MOV     (SP)+,R5
57 012140  012604                               MOV     (SP)+,R4
```

```
     58 012142  012603                          MOV     (SP)+,R3
     59 012144  000207                          RETURN
```

```
                                        .SBTTL  CHKMTX -- See if device is mounted by other users
                                ;-----------------------------------------------------------------------
                                ;  CHKMTX is called to see if a mounted device has been mounted by
                                ;  any users other than the current user.
                                ;
                                ;  Inputs:
                                ;   CDBUF contains mount table entry for device
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Device is mounted by other users.
                                ;
012146  010246                  CHKMTX: MOV     R2,-(SP)
012150  010346                          MOV     R3,-(SP)
012152  010446                          MOV     R4,-(SP)
012154  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get mount flag for current job
                                ;
012156  004767  177412                  CALL    CDJFLG          ;Get mount flag for our job
012162  111204                          MOVB    (R2),R4         ;Save mount flags for byte with our mount flag
012164  140312                          BICB    R3,(R2)         ;Clear mount flag for our job
                                ;
                                ;  See if any other jobs have this device mounted
                                ;
012166  012705  0000000C                MOV     #CDBUF+CD$JOB,R5;Point to table with mount flags
012172  012700  0000000G                MOV     #CD$$UB,R0      ;Get # bytes in table
012176  105725                  7$:     TSTB    (R5)+           ;Any other mount flags set?
012200  001004                          BNE     8$              ;Br if yes
012202  077003                          SOB     R0,7$
012204  110412                          MOVB    R4,(R2)         ;Reset mount flag for our job
012206  000241                  6$:     CLC                     ;Signal that device is not mounted by others
012210  000402                          BR      9$
012212  110412                  8$:     MOVB    R4,(R2)         ;Reset mount flag for our job
012214  000261                          SEC                     ;Signal that device is mounted by others
                                ;
                                ;  Finished
                                ;
012216  012605                  9$:     MOV     (SP)+,R5
012220  012604                          MOV     (SP)+,R4
012222  012603                          MOV     (SP)+,R3
012224  012602                          MOV     (SP)+,R2
012226  000207                          RETURN
```

```
                                        .SBTTL  CKCLUS -- Check to see if a CL unit is in use
                                ;-----------------------------------------------------------------------
                                ;  Check to see if a CL unit is in use by any job.
                                ;
                                ;  Inputs:
                                ;   R0 = CL unit index (2 * CL unit number)
                                ;
                                ;  Outputs:
                                ;   R0 = Number of any job that is using CL unit (0 if free)
                                ;
012230                          CKCLUS:
                                ;
                                ;  See if CL unit is in use by SET HOST
                                ;
012230  005760  000000G                 TST     CL$XLN(R0)      ;Any job using with SET HOST?
012234  001404                          BEQ     1$              ;Br if not
012236  016000  000000G                 MOV     CL$XLN(R0),R0   ;Get number of job
012242  006200                          ASR     R0              ;Convert index # to job #
012244  000420                          BR      9$
                                ;
                                ;  See if any job has CL unit open or allocated
                                ;
012246  006200                  1$:     ASR     R0              ;Convert to unit number
012250  020027  000007                  CMP     R0,#7.          ;Is this a CL or C1 unit?
012254  101405                          BLOS    2$              ;Br if CL
012256  162700  000010                  SUB     #8.,R0          ;Remove C1 unit bias
012262  066700  165632                  ADD     R50C10,R0       ;Form Rad50 device name
012266  000402                          BR      3$
012270  066700  165616          2$:     ADD     R50CL0,R0       ;Form rad50 device name
012274  010067  000000G         3$:     MOV     R0,ALCDEV       ;Set device name for EMT
012300  012700  000000G                 MOV     #TALEMT,R0      ;Point to check-allocation EMT
012304  104375                          EMT     375             ;See if this unit is allocated or in use
                                ;
                                ;  Finished
                                ;
012306  000207                  9$:     RETURN
```

```
                        .SBTTL  CHKALC -- Determine if device is allocated to another user
                ;-----------------------------------------------------------------------
                ;  CHKALC is called to determine if a device is allocated to another user.
                ;  If the device is allocated to another user, an error message is printed
                ;  and control is returned to RDCMD.
                ;
                ;  Inputs:
                ;   R0 = RAD50 device name
                ;
012310  010046  CHKALC: MOV     R0,-(SP)
012312  010446          MOV     R4,-(SP)
012314  010546          MOV     R5,-(SP)
                ;
                ;  Set name of device in EMT argument block
                ;
012316  010067  000000G         MOV     R0,ALCDEV       ;Set name of device for EMT
                ;
                ;  Don't do allocation test for LD device
                ;
012322  010005          MOV     R0,R5           ;Get name of device
012324  004767  177300  CALL    CHKDEV          ;Convert to device index number
012330  120467  000000G CMPB    R4,LDDEVX       ;Is this a LD device?
012334  001455          BEQ     9$              ;Br if yes -- allocation ok
                ;
                ;  Execute EMT that will test for allocation conflict
                ;
012336  012700  000000G         MOV     #TALEMT,R0      ;Point to EMT argument block
012342  104375          EMT     375             ;Do allocation test
012344  010005          MOV     R0,R5           ;Save # of any job that is using device
012346  103017          BCC     15$             ;Br if allocation is ok
                ;
                ;  An error occurred on the test allocation.
                ;  Print error message.
                ;
012350  123727  000000G 000002 CMPB    @#ERRLOC,#2     ;Invalid device?
012356  001444          BEQ     9$              ;Br if yes
012360  123727  000000G 000001 CMPB    @#ERRLOC,#1     ;Device already allocated by someone else?
012366  001007          BNE     15$             ;Br if not
012370                  FERR    #EM$DAA         ;Device allocated by another job
012404  000422          BR      17$
                ;
                ;  Device is either not allocated or is allocated to another
                ;     job from the same primary line.
                ;  See if device is in use by another job
                ;
012406  010500  15$:    MOV     R5,R0           ;Get # of job using device
012410  001427          BEQ     9$              ;Br if no job using device
012412  120467  000000G CMPB    R4,CLDEVX       ;Is device a CL unit?
012416  001424          BEQ     9$              ;Br if family member wants CL unit
012420  006300          ASL     R0              ;Convert to job index number
012422  120067  000000G CMPB    R0,CORUSR       ;In use by our job?
012426  001420          BEQ     9$              ;Br if yes
012430  005727  000000G TST     #KUSECK         ;Do we want to consider this as an error?
012434  001415          BEQ     9$              ;Br if not
012436                  FERR    #EM$DIU         ;Device is in use by another user
012452  004767  173476  17$:    CALL    PRTDEC          ;Print the number of the job that has the dev
012456                  .PRINT  #CRLF           ;Terminate the print line
```

```
     58 012464  000167  000000G                    JMP     RDCMD           ;Abort the command
     59                                ;
     60                                ;  We can access the device
     61                                ;
     62 012470  012605                 9$:        MOV     (SP)+,R5
     63 012472  012604                            MOV     (SP)+,R4
     64 012474  012600                            MOV     (SP)+,R0
     65 012476  000207                            RETURN
```

```
                                        .SBTTL  CDGET  -- Get local copy of mount device entry
                                ;-------------------------------------------------------------------
                                ;  CDGET is called to move a copy of a system mount device (cache)
                                ;  entry into CDBUF.
                                ;
                                ;  Inputs:
                                ;   R0 = Address within kernel of mount block to get.
                                ;
                                ;  Outputs:
                                ;   Copy of block is in CDBUF.
                                ;   R0 = Pointer to CDBUF.
                                ;
012500  010067  000000G         CDGET:  MOV     R0,CDGADR       ;Set address of block to get
012504  012700  000000G                 MOV     #CDGEMT,R0      ;Get address of EMT arg block
012510  104375                          EMT     375             ;Get copy of block
012512  012700  000000G                 MOV     #CDBUF,R0       ;Return pointer to CDBUF in R0
012516  000207                          RETURN

                                        .SBTTL  CDPUT  -- Store mount descriptor block into kernel
                                ;-------------------------------------------------------------------
                                ;  CDPUT is called to copy a device mount (cache) descriptor block from
                                ;  CDBUF into the kernel data base.
                                ;
                                ;  Inputs:
                                ;   R0 = Address within kerenl where block is to be stored.
                                ;   CDBUF = Copy of block to be moved.
                                ;
012520  010067  000000G         CDPUT:  MOV     R0,CDPADR       ;Set destination address
012524  012700  000000G                 MOV     #CDPEMT,R0      ;Point to EMT argument block
012530  104375                          EMT     375             ;Store the block
012532  000207                          RETURN
```

```
                                        .SBTTL  LDCLEN -- Perform SET LD CLEAN operation
                                ;-------------------------------------------------------------------
                                ;  LDCLEN call be called to perform the SET LD CLEAN function.
                                ;  It also causes the file directory cache to be cleaned out.
                                ;
012534  010246                  LDCLEN: MOV     R2,-(SP)
012536  010546                          MOV     R5,-(SP)
                                ;
                                ;  Perform LD CLEAN operation (reset logical disk information)
                                ;
012540  016705  000000G                 MOV     R50LDO,R5       ;GET NAME OF FIRST LOGICAL DISK (LDO)
012544  005002                          CLR     R2              ;GET INDEX TO 1ST LOGICAL DISK (LDO)
                                ;
                                ;  See if this logical disk is in use
                                ;
012546  010200                  1$:     MOV     R2,R0           ;GET LD INDEX NUMBER
012550  006300                          ASL     R0              ;* 4 WORDS PER ENTRY
012552  006300                          ASL     R0
012554  005760  000000G                 TST     LDNAME(R0)      ;IS THIS LOGICAL DISK IN USE?
012560  001423                          BEQ     2$              ;BR IF NOT
                                ;
                                ;  Dismount the logical disk
                                ;
012562  010567  000000G                 MOV     R5,MNTDEV       ;SET DEVICE NAME FOR DISMOUNT
012566  012700  000000G                 MOV     #DMTARG,R0      ;DISMOUNT THE DEVICE
012572  105267  000000G                 INCB    SERFLG          ;DO .SERR
012576  104375                          EMT     375
012600  105067  000000G                 CLRB    SERFLG          ;DO .HERR
                                ;
                                ;  Now reinitialize information about the logical disk
                                ;
012604  004767  000042                  CALL    LDMNT           ;CHECK IT
                                ;
                                ;  Now remount the logical disk
                                ;
012610  005762  000000G                 TST     LDPDEV(R2)      ;IS THE LOGICAL DISK THERE?
012614  001405                          BEQ     2$              ;BR IF NOT
012616  010567  000000G                 MOV     R5,MNTDEV       ;SET DEVICE NAME FOR THE MOUNT
012622  012700  000000G                 MOV     #MNTARG,R0      ;MOUNT THE DEVICE
012626  104375                          EMT     375
                                ;
                                ;  Check next logical disk
                                ;
012630  005205                  2$:     INC     R5              ;ADVANCE LDn NAME
012632  062702  000002                  ADD     #2,R2           ;ADVANCE LOGICAL DISK INDEX NUMBER
012636  020227  000016                  CMP     R2,#14.         ;HAVE WE CHECKED ALL LOGICAL DISKS?
012642  101741                          BLOS    1$              ;BR IF NOT
                                ;
                                ;  Finished
                                ;
012644  012605                          MOV     (SP)+,R5
012646  012602                          MOV     (SP)+,R2
012650  000207                          RETURN
```

```
                                        .SBTTL  LDMNT  -- Set up information about a logical disk
                                ;-----------------------------------------------------------------------
                                ;  LDMNT is called to set up information about a logical disk.
                                ;  Inputs:
                                ;   R2 = 2* logical disk # (0 to 14.)
                                ;   LDNAME(unit) = Name of file associated with logical disk.
                                ;
                                ;  Outputs:
                                ;   LDPDEV(unit) = Physical device index # and unit #
                                ;   LDSIZE(unit) = Size of file
                                ;   LDBASE(unit) = Base block on physical disk of start of logical disk
                                ;   Carry-flag is set on return if file cannot be found.
                                ;
012652  010446                  LDMNT:  MOV     R4,-(SP)
012654  010546                          MOV     R5,-(SP)
                                ;
                                ;  Remove any entry for this logical disk from access control table
                                ;
012656  004767  000466                  CALL    DLLDAC          ;REMOVE LD ENTRY FROM ACCESS CONTROL TABLE
                                ;
                                ;  Do lookup on file
                                ;
012662  010205                          MOV     R2,R5           ;GET UNIT #
012664  006305                          ASL     R5              ;*4 WORDS PER ENTRY
012666  006305                          ASL     R5
012670  062705  000000G                 ADD     #LDNAME,R5      ;POINT TO FILE SPEC IN LDNAME TABLE
012674  005715                          TST     (R5)            ;IS THERE A FILE SPEC FOR THIS DISK?
012676  001475                          BEQ     9$              ;BR IF NOT
012700  112767  000001  000000G         MOVB    #1,SERFLG       ;DO .SERR TO AVOID ABORT FOR ILLEGAL DEVICE
012706                                  .LOOKUP #XAREA,#1,R5    ;LOOKUP THE FILE
012724  112767  000000  000000G         MOVB    #0,SERFLG       ;DO .HERR (DON'T CLEAR C-FLAG)
012732  103457                          BCS     9$              ;BR IF CAN'T FIND THE FILE
012734  010062  000000G                 MOV     R0,LDSIZE(R2)   ;SAVE THE SIZE OF THE FILE
                                ;
                                ;  Do .SAVESTATUS to get information about the file
                                ;
012740  012705  000000G                 MOV     #BLK0,R5        ;POINT TO AREA FOR SAVESTATUS DATA
012744                                  .SAVEST #XAREA,#1,R5    ;SAVE FILE STATUS
012762  016500  000000G                 MOV     C.CSW(R5),R0    ;GET CSW
012766  042700  177701                  BIC     #^C76,R0        ;EXTRACT DEVICE UNIT #
012772  110062  000000G                 MOVB    R0,LDPDEV(R2)   ;SAVE PHYSICAL DEVICE INDEX NUMBER
012776  116504  000000G                 MOVB    C.DEVQ(R5),R4   ;GET PHYSICAL UNIT NUMBER
013002  042704  177770                  BIC     #^C7,R4
013006  110462  000001G                 MOVB    R4,LDPDEV+1(R2) ;SET PHYS UNIT # FOR LOGICAL DISK
013012  016562  000000G 000000G         MOV     C.SBLK(R5),LDBASE(R2) ;GET BASE BLOCK # OF LOGICAL DISK
                                ;
                                ;  If we have read-only access to the file associated with the logical
                                ;  disk, set no-write flag for this LD entry.
                                ;
013020  032765  000000G 000000G         BIT     #CS$RON,C.CSW(R5);DO WE HAVE READ-ONLY ACCESS TO FILE?
013026  001403                          BEQ     2$              ;BR IF NOT
013030  052762  000000G 000000G         BIS     #LD$RON,LDFLAG(R2);SET NO-WRITE FLAG FOR THIS LD
                                ;
                                ;  See if the logical disk file is itself within a logical disk
                                ;  (i.e., this is a nested logical disk)
                                ;
013036  020067  000000          2$:     CMP     R0,LDDEVX       ;IS FILE ON A LOGICAL DISK?
```

```
     58 013042  001007                          BNE     1$                  ;BR IF NOT
     59 013044  006304                          ASL     R4                  ;CVT UNIT # TO LOG DISK TABLE INDEX
     60 013046  016462  000000G 000000G          MOV     LDPDEV(R4),LDPDEV(R2) ;GET REAL PHYSICAL DEVICE & UNIT #
     61 013054  066462  000000G 000000G          ADD     LDBASE(R4),LDBASE(R2) ;BIAS BASE BLOCK # BY LOG DISK BASE
     62                                ;
     63                                ;  Make entry for the LD in the access control table.
     64                                ;
     65 013062  004767  000100          1$:     CALL    ADLDAC              ;MAKE ENTRY IN ACCESS CONTROL TABLE
     66                                ;
     67                                ;  Success
     68                                ;
     69 013066  000241                          CLC                         ;SIGNAL SUCCESS ON RETURN
     70 013070  000403                          BR      10$
     71                                ;
     72                                ;  Error -- Could not find the file
     73                                ;
     74 013072  005062  000000G         9$:     CLR     LDPDEV(R2)          ;SAY LOGICAL DISK IS NOT ACTIVE
     75 013076  000261                          SEC                         ;SIGNAL FAILURE ON RETURN
     76 013100  012605                  10$:    MOV     (SP)+,R5
     77 013102  012604                          MOV     (SP)+,R4
     78 013104  000207                          RETURN
```

```
                                        .SBTTL  CKLDAC -- Check if LD is in access control table
                                ;-------------------------------------------------------------------------
                                ;  CKLDAC is called to determine if a certain logical disk is in the
                                ;  device/file access control table.
                                ;
                                ;  Inputs:
                                ;   R2 = Logical disk index number (2 * unit #)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found logical disk entry in access control table.
                                ;   C-flag set     ==> Logical disk entry is not in access table.
                                ;   R0 = Pointer to access control entry.
                                ;
013106  010246                  CKLDAC: MOV     R2,-(SP)        ;SAVE ORIGINAL LD INDEX NUMBER
013110  006202                          ASR     R2              ;CONVERT INDEX # TO UNIT #
013112  005767  000000G                 TST     RESDEV          ;ARE THERE ANY ENTRIES IN ACCESS TABLE?
013116  001416                          BEQ     4$              ;BR IF NOT
                                ;
                                ;  There are entries in the access control table.
                                ;  Search for entry matching our logical disk.
                                ;
013120  012700  000000G                 MOV     #OKFILE,R0      ;POINT TO START OF ACCESS TABLE
013124  126067  000000G 000000G 1$:     CMPB    OF$DEV(R0),LDDEVX ;IS THIS ENTRY FOR A LOGICAL DISK?
013132  001003                          BNE     2$              ;BR IF NOT
013134  120260  000000G                 CMPB    R2,OF$UNT(R0)   ;IS ENTRY FOR SPECIFIED UNIT?
013140  001407                          BEQ     3$              ;BR IF YES -- FOUND ENTRY
013142  062700  000000G         2$:     ADD     #OF$$SZ,R0      ;POINT TO NEXT ACCESS ENTRY
013146  020067  000000G                 CMP     R0,OKFAND       ;CHECKED ALL ENTRIES?
013152  103764                          BLO     1$              ;BR IF NOT
                                ;
                                ;  LD entry is not in access table
                                ;
013154  000261                  4$:     SEC                     ;SIGNAL FAILURE ON RETURN
013156  000401                          BR      9$
                                ;
                                ;  Found LD entry in table
                                ;
013160  000241                  3$:     CLC                     ;SIGNAL SUCCESS ON RETURN
013162  012602                  9$:     MOV     (SP)+,R2        ;RECOVER LD INDEX NUMBER
013164  000207                          RETURN
```

```
                                        .SBTTL  ADLDAC -- Add LD entry to access control table
                                ;-----------------------------------------------------------------------
                                ;  ADLDAC is called to add a logical disk entry to the device/file
                                ;  access control table.  If there are no protected devices or files
                                ;  no entry is made.
                                ;
                                ;  Inputs:
                                ;   R2 = Logical disk index number (2 * unit number)
                                ;
013166  005767  000000G         ADLDAC: TST     RESDEV          ;ANY ENTRIES IN ACCESS TABLE?
013172  001465                        BEQ     9$              ;BR IF NOT
                                ;
                                ;  There are entries in the access control table
                                ;
013174  010246                      MOV     R2,-(SP)
013176  010546                      MOV     R5,-(SP)
013200  006202                      ASR     R2              ;CONVERT LD INDEX # TO UNIT #
                                ;
                                ;  Find a free entry in the access table
                                ;
013202  012705  000000G             MOV     #OKFILE,R5      ;POINT TO START OF ACCESS TABLE
013206  020567  000000G         1$: CMP     R5,OKFNND       ;REACHED END OF TABLE?
013212  103036                      BHIS    3$              ;BR IF YES -- TABLE OVERFLOW
013214  005765  000000G             TST     OF$FIL(R5)      ;IS THIS ENTRY FREE?
013220  001403                      BEQ     2$              ;BR IF FREE
013222  062705  000000G             ADD     #OF$$SZ,R5      ;POINT TO NEXT ENTRY
013226  000767                      BR      1$              ;GO CHECK IT
                                ;
                                ;  We found a free entry.  Add entry for LD.
                                ;
013230  116765  000000G 000000G 2$:      MOVB    LDDEVX,OF$DEV(R5);SET LOGICAL DISK DEVICE INDEX NUMBER
013236  110265  000000G             MOVB    R2,OF$UNT(R5)   ;SET LOGICAL DISK UNIT #
013242  012700  000000G             MOV     #WLDNAM,R0      ;SET FILE NAME TO WILDCARDS
013246  010065  000000G             MOV     R0,OF$FIL(R5)
013252  010065  000002G             MOV     R0,OF$FIL+2(R5)
013256  010065  000004G             MOV     R0,OF$FIL+4(R5)
013262  105065  000000G             CLRB    OF$FLG(R5)      ;INITIALLY CLEAR ALL CONTROL FLAGS
013266  006302                      ASL     R2              ;CVT UNIT # TO LD INDEX #
013270  032762  000000G 000000G     BIT     #LD$RON,LDFLAG(R2);IS LOGICAL DISK WRITE PROTECTED?
013276  001412                      BEQ     4$              ;BR IF NOT
013300  152765  000000G 000000G     BISB    #OT$RON,OF$FLG(R5);SET READ-ONLY FLAG IN ACCESS TABLE
013306  000406                      BR      4$
                                ;
                                ;  Error -- Access table overflow
                                ;
013310                          3$: FERR    #TBLOVF         ;TABLE OVERFLOW
                                ;
                                ;  Finished.  Extend ACCESS table end ptr if we need to.
                                ;
013324  062705  000000G         4$: ADD     #OF$$SZ,R5      ;Point past this entry
013330  020567  000000G             CMP     R5,OKFAND       ;Is it past the end?
013334  101402                      BLOS    41$             ;Br if not, don't need to extend
013336  010567  000000G             MOV     R5,OKFAND       ;Extend the table end ptr
013342  012605                  41$: MOV     (SP)+,R5
013344  012602                      MOV     (SP)+,R2
013346  000207                  9$: RETURN
```

DLLDAC -- Delete LD entry from access control table

```
                                        .SBTTL  DLLDAC -- Delete LD entry from access control table
                                ;-------------------------------------------------------------------------
                                ;  DLLDAC is called to delete any entry in the access control table
                                ;  for a specified logical disk.
                                ;
                                ;  Inputs:
                                ;   R2 = Logical disk index number (2 * unit number)
                                ;
013350  004767  177532          DLLDAC: CALL    CKLDAC          ;IS THERE AN ENTRY FOR THIS LD IN TABLE?
013354  103416                          BCS     1$              ;BR IF NOT
                                ;
                                ;  We have found an entry in the access table for this LD.
                                ;  R0 = Pointer to the entry.
                                ;
013356  105060  000000G                 CLRB    OF$DEV(R0)      ;MARK ENTRY AS FREE
013362  105060  000000G                 CLRB    OF$UNT(R0)
013366  005060  000000G                 CLR     OF$FIL(R0)
                                ;
                                ;  If we had to extend the ACCESS table end ptr to include this entry,
                                ;  then reduce it back to below this entry.
                                ;
013372  062700  000000G                 ADD     #OF$$SZ,R0      ;Point past table entry
013376  020067  000000G                 CMP     R0,OKFAND       ;Was it the last entry?
013402  103403                          BLO     1$              ;Br if not
013404  162767  000000G 000000G        SUB     #OF$$SZ,OKFAND  ;It was last entry, drop below it
                                ;
                                ;  Finished
                                ;
013412  000207                  1$:     RETURN
```

```
                                        .SBTTL  DOASGN -- Add entry to the ASSIGN table
                                ;-----------------------------------------------------------------------
                                ;  DOASGN is called to make an entry in the ASSIGN table.
                                ;
                                ;  Inputs:
                                ;   R5 = Logical device name.
                                ;   R0 = Physical device name.
                                ;
013414  010246                 DOASGN: MOV     R2,-(SP)
                                ;
                                ;  Determine if the "physical" device name is actually a logical name
                                ;
013416  004767  000220          1$:     CALL    ASNSRC          ;SEE IF THIS IS ACTUALLY A LOGICAL NAME
013422  103402                          BCS     2$              ;BR IF NOT
013424  016200  000004                  MOV     4(R2),R0        ;REPLACE PHYSICAL NAME WITH NEW NAME
                                ;
                                ;  See if an entry already exists in the assign table for this logical name
                                ;
013430  010046                 2$:     MOV     R0,-(SP)        ;SAVE PHYSICAL DEVICE NAME
013432  010500                          MOV     R5,R0           ;GET LOGICAL NAME
013434  004767  000202                  CALL    ASNSRC          ;LOOK IT UP
013440  103010                          BCC     3$              ;BR IF FOUND ENTRY -- REUSE THE ENTRY
                                ;
                                ;  Add a new entry to the assign table
                                ;
013442  005000                          CLR     R0              ;SEARCH FOR A FREE ENTRY IN THE ASSIGN TABLE
013444  004767  000172                  CALL    ASNSRC          ;LOOK FOR FREE ENTRY
013450  103004                          BCC     3$              ;BR IF FOUND ONE
013452                                  FABORT  #ASNOVF         ;ASSIGN TABLE OVERFLOW
                                ;
                                ;  Move information into assign table
                                ;
013462  010522                 3$:     MOV     R5,(R2)+        ;LOGICAL NAME
013464  005022                          CLR     (R2)+           ;NO FILE SIZE
013466  012622                          MOV     (SP)+,(R2)+     ;PHYSICAL DEVICE NAME
013470  005022                          CLR     (R2)+           ;NO FILE NAME
013472  005022                          CLR     (R2)+
013474  005022                          CLR     (R2)+           ;NO EXTENSION
                                ;
                                ;  Finished
                                ;
013476  012602                          MOV     (SP)+,R2
013500  000207                          RETURN
```

```
      1                                                .SBTTL  CVDVNM -- Convert device number to device name
      2                                       ;------------------------------------------------------------------
      3                                       ;  CVDVNM is called to convert a device number / unit number combination
      4                                       ;  into a RAD50 device name.
      5                                       ;
      6                                       ;  Inputs:
      7                                       ;   R0 = Device number (low-order byte), unit number (high-order byte)
      8                                       ;
      9                                       ;  Outputs:
     10                                       ;   R0 = RAD50 device name.
     11                                       ;
     12 013502  010346                        CVDVNM: MOV     R3,-(SP)
     13 013504  010003                                MOV     R0,R3           ;Copy device # and unit #
     14 013506  000303                                SWAB    R3              ;Put unit # in low-order byte
     15 013510  042703  177770                        BIC     #^C7,R3         ;Clear all but unit number in R3
     16 013514  042700  177400                        BIC     #^C377,R0       ;Clear all but device number in R0
     17 013520  016000  000000G                       MOV     PNAME(R0),R0    ;Get base device name
     18 013524  062703  000036                        ADD     #30.,R3         ;Convert unit number to RAD50 character
     19 013530  060300                                ADD     R3,R0           ;Combine unit number with device name
     20 013532  012603                                MOV     (SP)+,R3
     21 013534  000207                                RETURN
```

```
                                        .SBTTL  CHKCLU -- See if device name is CL or C1 unit
                                ;-----------------------------------------------------------------------
                                ;  Determine if a rad50 device name is a CL or C1 unit.
                                ;  If so, determine the unit number.
                                ;
                                ;  Inputs:
                                ;   R0 = RAD50 device name (e.g., CL2)
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> This is a CL or C1 unit
                                ;   C-flag set     ==> This is not a CL or C1 unit
                                ;   R0 = CL unit number (0-15)
                                ;
013536                          CHKCLU:
                                ;
                                ;  See if this is a CL unit
                                ;
013536  020067  164346                  CMP     R0,R50CL        ;Is name "CL"?
013542  001002                          BNE     1$              ;Br if not
013544  005000                          CLR     R0              ;Translate to unit 0
013546  000431                          BR      7$
013550  020067  164336          1$:     CMP     R0,R50CL0       ;Is unit in the range CL0 to CL7?
013554  103406                          BLO     2$              ;Br if not
013556  020067  164332                  CMP     R0,R50CL7
013562  101003                          BHI     2$              ;Br if not
013564  166700  164322                  SUB     R50CL0,R0       ;Get unit number
013570  000420                          BR      7$
                                ;
                                ;  See if this is a C1 unit
                                ;
013572  020067  164320          2$:     CMP     R0,R50C1        ;Is unit name "C1"?
013576  001003                          BNE     3$              ;Br if not
013600  012700  000010                  MOV     #8.,R0          ;C1 = unit 8
013604  000412                          BR      7$
013606  020067  164306          3$:     CMP     R0,R50C10       ;Is unit in the range C10 to C17?
013612  103411                          BLO     8$              ;Br if not
013614  020067  164302                  CMP     R0,R50C17
013620  101006                          BHI     8$              ;Br if not
013622  166700  164272                  SUB     R50C10,R0       ;Get C1 unit number
013626  062700  000010                  ADD     #8.,R0          ;Add C1 unit bias
                                ;
                                ;  This is a CL or C1 unit
                                ;
013632  000241                  7$:     CLC                     ;Signal success on return
013634  000401                          BR      9$
                                ;
                                ;  This is not a CL or C1 unit
                                ;
013636  000261                  8$:     SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
013640  000207                  9$:     RETURN
```

```
                                        .SBTTL  ASNSRC -- Search assign table for logical name
                                ;-------------------------------------------------------------
                                ;  ASNSRC is called to search the assign table for an entry
                                ;  with a specified logical name.
                                ;
                                ;  Inputs:
                                ;   R0 = Logical name to search for.
                                ;
                                ;  Outputs:
                                ;   C-flag set on return if no assign block found with matching name.
                                ;   R2 = Address of assign block if one found.
                                ;
013642  010146                  ASNSRC: MOV     R1,-(SP)
013644  012701  0000000G                MOV     #MAXASN,R1      ;GET # ASSIGN BLOCKS
013650  012702  0000000G                MOV     #ASNTBL,R2      ;POINT TO ASSIGN TABLE
013654  020062  0000000G        1$:     CMP     R0,AT$LOG(R2)   ;COMPARE LOGICAL NAMES
013660  001405                        BEQ     2$              ;BR IF WE FOUND BLOCK WE ARE LOOKING FOR
013662  062702  0000000G              ADD     #AT$$SZ,R2      ;POINT TO NEXT ASSIGN BLOCK
013666  077106                        SOB     R1,1$           ;LOOP IF MORE BLOCKS TO CHECK
013670  000261                        SEC                     ;SIGNAL FAILURE
013672  000401                        BR      3$
013674  000241                2$:     CLC                     ;SIGNAL SUCCESS
013676  012601                3$:     MOV     (SP)+,R1
013700  000207                        RETURN
```

```
                                        .SBTTL  LOGASN -- Perform full logical device assignment
                                ;-------------------------------------------------------------
                                ;  LOGASN is called to perform a full logical device name assignment.
                                ;  The logical name associated with a file specification is translated
                                ;  into the corresponding physical device.  The file name, extension,
                                ;  and size may also be translated if a file spec was specified with
                                ;  the assignment of the logical name.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to 5 word block containing file spec (dev,file,file,ext,size)
                                ;
                                ;  Outputs:
                                ;   File spec is updated to have physical device name and possibly
                                ;   altered file name.
                                ;
013702  010246                  LOGASN: MOV     R2,-(SP)
013704  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if device name is in our assign table
                                ;
013706  011500                          MOV     (R5),R0         ;Get logical device name
013710  004767  177726                  CALL    ASNSRC          ;See if name is in assign table
013714  103421                          BCS     1$              ;Br if name is not in assign table
                                ;
                                ;  Found logical device name in the assign table.
                                ;  Translate to physical device.
                                ;
013716  010504                          MOV     R5,R4           ;Get pointer to file spec buffer
013720  016224  000000G                 MOV     AT$DEV(R2),(R4)+;Put in physical device name
013724  016200  000000G                 MOV     AT$FIL(R2),R0   ;Was file name assigned?
013730  001413                          BEQ     1$              ;Br if not
013732  010024                          MOV     R0,(R4)+        ;Translate file name
013734  016224  000002G                 MOV     AT$FIL+2(R2),(R4)+
013740  016200  000000G                 MOV     AT$EXT(R2),R0   ;Was file extension specified?
013744  001405                          BEQ     1$              ;Br if not
013746  010024                          MOV     R0,(R4)+        ;Translate file extension
013750  016200  000000G                 MOV     AT$SIZ(R2),R0   ;Was file size specified?
013754  001401                          BEQ     1$              ;Br if not
013756  010014                          MOV     R0,(R4)         ;Translate file size
                                ;
                                ;  Translate "DK" and "SY" to physical device names
                                ;
013760  021567  000000G         1$:     CMP     (R5),R50SY      ;Is device name "SY"?
013764  001403                          BEQ     2$              ;Br if yes
013766  021567  000000G                 CMP     (R5),R50DK      ;Is device name "DK"?
013772  001002                          BNE     3$              ;Br if not
013774  016715  000000G         2$:     MOV     SYNAME,(R5)     ;Translate to physical device
                                ;
                                ;  Finished
                                ;
014000  012604                  3$:     MOV     (SP)+,R4
014002  012602                          MOV     (SP)+,R2
014004  000207                          RETURN
```

```
                                .SBTTL  FORCE0 -- Force a 2-char dev name to unit 0
                        ;-------------------------------------------------------------------
                        ;  Inputs: R3 points to a RAD50 device name
                        ;
                        ;  Outputs: If the 3rd char of the device name pointed to by R3 is
                        ;       blank, then it is changed to 0
                        ;
014006  010346          FORCE0: MOV     R3,-(SP)
014010  010446                  MOV     R4,-(SP)
014012  010546                  MOV     R5,-(SP)
014014  011305                  MOV     (R3),R5         ;MOVE CURRENT DEV NAME TO R5
014016  005004                  CLR     R4              ;SET UP FOR DIVIDE
014020  071427  000050          DIV     #50,R4          ;SEPARATE INTO NAME AND UNIT
014024  005705                  TST     R5              ;WAS 3RD CHAR BLANK?
014026  001012                  BNE     9$              ;RETURN IF NOT
014030  010405                  MOV     R4,R5           ;GET HIGH 2 CHARS
014032  005004                  CLR     R4              ;SET UP FOR ANOTHER DIVIDE
014034  071427  000050          DIV     #50,R4          ;SEPARATE 1 & 2 CHARS
014040  005704                  TST     R4              ;WAS CHAR 1 BLANK?
014042  001404                  BEQ     9$              ;EMPTY OR INVALID DEV NAME!
014044  005705                  TST     R5              ;WAS CHAR 2 BLANK?
014046  001402                  BEQ     9$              ;1-CHAR DEV NAME SHOULD BE INVALID???
014050  062713  000036          ADD     #^R  0,(R3)     ;FORCE TO UNIT 0
014054  012605          9$:     MOV     (SP)+,R5
014056  012604                  MOV     (SP)+,R4
014060  012603                  MOV     (SP)+,R3
014062  000207                  RETURN
```

```
                                        .SBTTL  DEADEV -- Deassign physical device
                                ;-------------------------------------------------------------------
                                ;  DEADEV is called to remove from the assign table all entries
                                ;  for logical device names that are assigned to a specified
                                ;  physical device.
                                ;
                                ;  Inputs:
                                ;   R0 = Name of physical device.
                                ;
014064  010246                  DEADEV: MOV     R2,-(SP)
014066  010346                          MOV     R3,-(SP)
014070  012702  000000G                 MOV     #ASNTBL,R2      ;Point to assign table
014074  012703  000000G                 MOV     #MAXASN,R3      ;Get # assign table entries
014100  020062  000000G         1$:     CMP     R0,AT$DEV(R2)   ;Is this entry for specified phys device?
014104  001004                          BNE     2$              ;Br if not
014106  005062  000000G                 CLR     AT$LOG(R2)      ;Clear logical device name
014112  005062  000000G                 CLR     AT$DEV(R2)      ;Clear physical device name
014116  062702  000000G         2$:     ADD     #AT$$SZ,R2      ;Point to next assign entry
014122  077312                          SOB     R3,1$           ;Loop if more to check
014124  012603                          MOV     (SP)+,R3
014126  012602                          MOV     (SP)+,R2
014130  000207                          RETURN
```

```
                                        .SBTTL  INSSRC -- Search for program in INSTALL table
                                ;-----------------------------------------------------------------------
                                ;  Search the INSTALL table for an entry corresponding to the program
                                ;  being started.
                                ;
                                ;  Inputs:
                                ;   R0 = Address of buffer with file specification.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found entry for program.
                                ;   C-flag set ==> No entry for program.
                                ;   IIBUF = Install entry for program if one is found.
                                ;
014132  010246                  INSSRC: MOV     R2,-(SP)
014134  010346                          MOV     R3,-(SP)
014136  010446                          MOV     R4,-(SP)
014140  010546                          MOV     R5,-(SP)
                                ;
                                ;  Copy file specification to INSSPC and perform any assigns
                                ;
014142  012702  000134'                 MOV     #INSSPC,R2      ;Point to result buffer
014146  012703  000004                  MOV     #4,R3           ;Get # words to move
014152  012022                  10$:    MOV     (R0)+,(R2)+     ;Copy the file spec
014154  077302                          SOB     R3,10$
014156  012705  000134'                 MOV     #INSSPC,R5      ;Point to name
014162  004767  177514                  CALL    LOGASN          ;Perform full assignment
                                ;
                                ;  Check next entry in INSTALL table
                                ;
014166  016705  000000G                 MOV     INSTBL,R5       ;Point to 1st entry in install table
014172  010567  000000G         1$:     MOV     R5,INGADR       ;Set address of entry to get
014176  012700  000000G                 MOV     #INGEMT,R0      ;Point to EMT arg block
014202  104375                          EMT     375             ;Get the install entry
014204  012702  000000C                 MOV     #IIBUF+II$NAM,R2;Point to entry we just got
014210  012703  000134'                 MOV     #INSSPC,R3      ;Point to target name
014214  012700  000004                  MOV     #4,R0           ;# words to compare
014220  021227  000000G         3$:     CMP     (R2),#WLDNAM    ;Wildcard in install entry?
014224  001402                          BEQ     7$              ;Br if yes
014226  021213                          CMP     (R2),(R3)       ;Compare file specs
014230  001003                          BNE     2$              ;Br if they don't match
014232  022223                  7$:     CMP     (R2)+,(R3)+     ;Advance both pointers
014234  077007                          SOB     R0,3$           ;Loop if more to compare
014236  000407                          BR      4$              ;We found the entry
014240  062705  000000G         2$:     ADD     #II$$SZ,R5      ;Point to next install entry
014244  020567  000000G                 CMP     R5,INSTBN       ;Checked all entries?
014250  103750                          BLO     1$              ;Loop if more to check
                                ;
                                ;  Cannot find entry for this program
                                ;
014252  000261                  6$:     SEC                     ;Signal failure on return
014254  000415                          BR      9$
                                ;
                                ;  Found entry for program
                                ;
014256  026727  000001C 000000G 4$:     CMP     IIBUF+II$NAM,#WLDNAM ;Is install device wild ("*")?
014264  001410                          BEQ     8$              ;Br if yes
014266  016705  000000G                 MOV     RUNDEV,R5       ;Get device spec for program
```

```
     58 014272  004767  175332                    CALL    CHKDEV          ;Convert device name to dev index number
     59 014276  103403                            BCS     8$              ;Br if invalid device
     60 014300  020467  000000G                   CMP     R4,LDDEVX       ;Is program on a logical disk?
     61 014304  001762                            BEQ     6$              ;Br if yes -- Cannot be installed
     62 014306  000241                  8$:       CLC                     ;Signal success on return
     63                                 ;
     64                                 ;  Finished
     65                                 ;
     66 014310  012605                  9$:       MOV     (SP)+,R5
     67 014312  012604                            MOV     (SP)+,R4
     68 014314  012603                            MOV     (SP)+,R3
     69 014316  012602                            MOV     (SP)+,R2
     70 014320  000207                            RETURN
```

```
                                        .SBTTL  LSTSPL -- List pending spool files for a device
                                ;-------------------------------------------------------------------------
                                ;  LSTSPL is called to display information about spool files pending
                                ;  for a specified spooled device.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of SDCB for device for which file info is to be printed.
                                ;
014322  010146                  LSTSPL: MOV     R1,-(SP)
014324  010246                          MOV     R2,-(SP)
014326  010346                          MOV     R3,-(SP)
014330  010446                          MOV     R4,-(SP)
014332  010546                          MOV     R5,-(SP)
014334  016102  000000G                 MOV     SDFHD(R1),R2    ;GET ADDRESS OF FIRST SFCB FOR THIS DEVICE
014340  001525                          BEQ     9$              ;BR IF NO FILES PENDING FOR THIS DEVICE
                                ;
                                ;  Print info about next SFCB for this device.
                                ;  R1 = Address of SDCB.
                                ;  R2 = Address of SFCB.
                                ;
                                ;  Move SFCF from kernel area to TSKMON buffer (BLKO)
                                ;
014342  010267  163476          1$:     MOV     R2,PEKADR       ;Set address of block in kernel
014346  012767  000000G 163472          MOV     #SFCBSZ,PEKSIZ  ;Set size of data to get
014354  012700  000040'                 MOV     #PEKEMT,R0      ;Point to EMT argument block
014360  104375                          EMT     375             ;Move SFCB from kernel to our buffer
014362  012702  000000G                 MOV     #BLKO,R2        ;Point to buffer with SFCB we got
                                ;
                                ;  See if 1st write has been done to this spool file.
                                ;
014366  132762  000000G 000000G        BITB    #SF$1ST,SFFLAG(R2);HAS 1ST WRITE BEEN DONE?
014374  001504                        BEQ     5$              ;BR IF NOT
                                ;
                                ;  Print the file ID number
                                ;
014376  016205  000000G                 MOV     SFID(R2),R5     ;Get spool file ID number
014402  012703  000004                  MOV     #4.,R3          ;Print 4 digits
014406  004767  171644                  CALL    PRTFIX          ;Print ID value
014412                                  .PRINT  #SPACE2         ;Print 2 spaces
                                ;
                                ;  Print device name.
                                ;
014420  016100  000000G                 MOV     SDNAME(R1),R0   ;GET NAME OF SPOOLED DEVICE (RAD50)
014424  004767  171146                  CALL    PRTR50          ;PRINT THE DEVICE NAME
                                ;
                                ;  Print a star if this file is being printed now.
                                ;
014430  012700  000040                  MOV     #' ,R0          ;ASSUME FILE NOT BEING PRINTED
014434  132762  000000G 000000G        BITB    #SF$BSY,SFFLAG(R2);IS FILE BEING PRINTED NOW?
014442  001402                        BEQ     4$              ;BR IF NOT
014444  012700  000052                MOV     #'*,R0          ;PRINT *
014450                          4$:     .TTYOUT
014454                                  .PRINT  #SPACE2
                                ;
                                ;  Print user number
                                ;
014462  116205  000000G                 MOVB    SFUSER(R2),R5   ;GET USER INDEX #
```

```
                        ASR     R5              ;CONVERT TO SEQUENTIAL NUMBER
                        MOV     #2.,R3          ;PRINT 2 CHARS
                        CALL    PRTFIX          ;PRINT VALUE
                        .PRINT  #SPACE2
                ;
                ;  Print file name
                ;
                        MOV     SFFILE(R2),R0   ;GET 1ST 3 CHARS OF FILE NAME (RAD50)
                        CALL    PRTR50          ;PRINT THEM
                        MOV     SFFILE+2(R2),R0 ;PRINT 2ND 3 CHARS
                        CALL    PRTR50
                        .PRINT  #SPACE2
                ;
                ;  Print form name
                ;
                        MOV     R2,R3
                        ADD     #SFFORM,R3      ;POINT TO CELL WITH FORM NAME
                        MOV     #6.,R4          ;PRINT 6 CHARS
                3$:     MOVB    (R3)+,R0        ;GET NEXT CHAR OF NAME
                        .TTYOUT                 ;PRINT IT
                        SOB     R4,3$
                        .PRINT  #SPACE2
                ;
                ;  Print number of blocks in spool file
                ;
                        MOV     SFNMBL(R2),R5   ;GET # BLOCKS IN SPOOL FILE
                        MOV     #4.,R3          ;PRINT 4 CHARS
                        CALL    PRTFIX          ;PRINT VALUE
                ;
                ;  end of print line
                ;
                        .PRINT  #CRLF
                ;
                ;  See if there are more spool files for this device.
                ;
                5$:     MOV     SFQLNK(R2),R2   ;CHAIN TO NEXT SFCB
                        BNE     1$              ;BR IF MORE TO PRINT
                ;
                ;  Finished
                ;
                9$:     MOV     (SP)+,R5
                        MOV     (SP)+,R4
                        MOV     (SP)+,R3
                        MOV     (SP)+,R2
                        MOV     (SP)+,R1
                        RETURN
```

```
                                        .SBTTL  CHKDLM -- See if char is a delimiter
                                ;-------------------------------------------------------------
                                ;  CHKDLM IS CALLED TO SEE IF THE CHARACTER IN R0 IS
                                ;  AN ALPHANUMERIC CHARACTER.
                                ;  IF IT IS THE C-FLAG IS RESET ON RETURN.
                                ;  IF CHAR IS A DELIMITER THE C-FLAG IS SET ON RETURN.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
014630  120027  000060        CHKDLM: CMPB    R0,#'0          ;IS CHAR A DIGIT?
014634  103422                        BLO     1$              ;BR IF NOT
014636  120027  000071                CMPB    R0,#'9
014642  101421                        BLOS    2$              ;BR IF DIGIT
014644  120027  000141                CMPB    R0,#141         ;IS THIS A LOWER CASE LETTER?
014650  103406                        BLO     3$              ;BR IF NOT
014652  120027  000172                CMPB    R0,#172         ;LOWER CASE Z
014656  101011                        BHI     1$              ;BR IF DELIMITER
014660  162700  000040                SUB     #40,R0          ;CONVERT LOWER-CASE TO UPPER CASE
014664  000410                        BR      2$
014666  120027  000101        3$:     CMPB    R0,#'A          ;IS CHAR A LETTER?
014672  103403                        BLO     1$              ;BR IF NOT
014674  120027  000132                CMPB    R0,#'Z
014700  101402                        BLOS    2$              ;BR IF LETTER
                                ;  CHARACTER IS A DELIMITER
014702  000261                1$:     SEC                     ;SIGNAL DELIMITER
014704  000207                        RETURN
                                ;  CHARACTER IS ALPHANUMERIC
014706  000241                2$:     CLC
014710  000207                        RETURN
```

```
                                        .SBTTL  CVTTAB -- Convert tab and FF chars to spaces
                                ;-----------------------------------------------------------------------
                                ;  CVTTAB is called to convert tab and form-feed characters in a
                                ;  command line to space characters. After the conversion is done,
                                ;  leading space characters are skipped over.
                                ;
                                ;  Inputs:
                                ;   R3 = Address of start of asciz command line.
                                ;
                                ;  Outputs:
                                ;   R3 = Pointer to first non-blank character in buffer.
                                ;   Command line has had tab and FF chars converted to spaces.
                                ;
014712  010346                  CVTTAB: MOV     R3,-(SP)
014714  112300                  1$:     MOVB    (R3)+,R0        ;Get next character from command
014716  001412                          BEQ     4$              ;Br if end of command
014720  120027  000011                  CMPB    R0,#TAB         ;Is this a tab character
014724  001403                          BEQ     2$              ;Br if yes
014726  120027  000014                  CMPB    R0,#FF          ;Is this a form-feed character?
014732  001370                          BNE     1$              ;Br if not
014734  112763  000040  177777  2$:     MOVB    #' ,-1(R3)      ;Replace control character with space
014742  000764                          BR      1$
                                ;
                                ;  Finished converting tabs and Form-feeds to spaces.
                                ;  Now skip over leading spaces.
                                ;
014744  012603                  4$:     MOV     (SP)+,R3        ;Get pointer to start of command
014746  122327  000040          3$:     CMPB    (R3)+,#' '      ;Skip leading spaces
014752  001775                          BEQ     3$
014754  005303                          DEC     R3              ;Point to 1st non-blank character
                                ;
                                ;  Finished
                                ;
014756  000207                          RETURN
```

```
                                        .SBTTL  CVTUC  -- Convert chars in command line to upper case
                                ;-----------------------------------------------------------------------
                                ;  Convert all lower case characters in an asciz string to upper case.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to asciz string to be converted.
                                ;
014760  010246                  CVTUC:  MOV     R2,-(SP)
014762  010302                          MOV     R3,R2           ;Get pointer to start of string
014764  112200                  1$:     MOVB    (R2)+,R0        ;Get next char from string
014766  001413                          BEQ     9$              ;Br if hit end of string
014770  120027  000141                  CMPB    R0,#141         ;Is this a lower case letter?
014774  103773                          BLO     1$              ;Br if not
014776  120027  000172                  CMPB    R0,#172
015002  101370                          BHI     1$              ;Br if not
015004  162700  000040                  SUB     #40,R0          ;Convert letter to upper case
015010  110062  177777                  MOVB    R0,-1(R2)       ;Store converted char back into string
015014  000763                          BR      1$
                                ;
                                ;  Finished
                                ;
015016  012602                  9$:     MOV     (SP)+,R2
015020  000207                          RETURN
```

```
                                        .SBTTL  SKPSPC -- Skip over spaces in command line
                                ;-------------------------------------------------------------------------
                                ;  Subroutine to skip over spaces in a command line.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer into command line.
                                ;
                                ;  Outputs:
                                ;   R3 = Pointer to next non-blank character.
                                ;
015022  122327  000040          SKPSPC: CMPB    (R3)+,#' ;IS NEXT CHARACTER A SPACE?
015026  001775                          BEQ     SKPSPC  ;BR IF YES -- SKIP IT
015030  005303                          DEC     R3      ;BACKUP POINTER TO FIRST NON-BLANK CHAR
015032  000207                          RETURN

                                        .SBTTL  SKPDLM -- Skip delimiters in command line
                                ;-------------------------------------------------------------------------
                                ;  Subroutine to check for legal delimiters (space(s), comma, or end of line)
                                ;  and skip over in a command line.
                                ;
                                ;  Inputs:
                                ;    R3 = Pointer to command line.
                                ;
                                ;  Outputs:
                                ;    R3 = Pointer to next command input character.
                                ;    C-bit = clear: legal delimiter found (blank(s), comma, end of line)
                                ;            set: no delimiter detected
                                ;

015034  010046                  SKPDLM: MOV     R0,-(SP)        ;Save register
015036  112300                          MOVB    (R3)+,R0        ;Get next command character
015040  001413                          BEQ     2$              ;Br if end of command hit
015042  120027  000040                  CMPB    R0,#BLANK       ;Check for space
015046  001403                          BEQ     1$              ;Br if space found
015050  120027  000054                  CMPB    R0,#COMMA       ;Check for comma
015054  001007                          BNE     3$              ;Character not blank or comma
                                ;
                                ;  Found legal delimiter - skip multiple spaces and commas.
                                ;
015056  004767  177740          1$:     CALL    SKPSPC          ;Skip multiple spaces
015062  122327  000054                  CMPB    (R3)+,#COMMA    ;Next character a comma (following spaces?)
015066  001773                          BEQ     1$              ;Br if comma found
015070  000241                  2$:     CLC                     ;Flag legal delimiter found
015072  000401                          BR      10$             ;Finished
                                ;
                                ;  Did not find a legal separator.
                                ;
015074  000261                  3$:     SEC                     ;Flag no legal delimiter found
                                ;  Use no instructions which alter the c-bit before the RETURN.
015076  005303                  10$:    DEC     R3              ;Back up to last not blank or comma character
015100  012600                          MOV     (SP)+,R0        ;Restore register
015102  000207                          RETURN                  ; and return

                                        .SBTTL  GETKCH -- Get next char from command line
                                ;-------------------------------------------------------------------------
                                ;  GETKCH is called to get the next character from a command line.
                                ;  Lower case characters are converted to upper-case before being returned.
```

```
                                        ;
                                        ;  Inputs:
                                        ;   R3 = Pointer to next character to be gotten.
                                        ;
                                        ;  Outputs:
                                        ;   R0 = Character gotten.
                                        ;   R3 = Updated to point to next character.
                                        ;
015104  112300          GETKCH: MOVB    (R3)+,R0        ;GET NEXT CHARACTER
015106  120027  000141          CMPB    R0,#141         ;IS IT A LOWER CASE LETTER?
015112  103405                  BLO     1$              ;BR IF DEFINITELY NOT
015114  120027  000172          CMPB    R0,#172         ;CHECK UPPER RANGE
015120  101002                  BHI     1$              ;BR IF NOT LOWER CASE
015122  162700  000040          SUB     #40,R0          ;CONVERT LOWER-CASE TO UPPER-CASE
015126  000207                  1$:     RETURN
```

```
                                        .SBTTL  DELSPC -- Delete spaces from command line
                                ;-------------------------------------------------------------------
                                ;  DELSPC is called to delete all space characters from a command line.
                                ;
                                ;  Inputs:
                                ;   R3 = Address of start of asciz command line.
                                ;
                                ;  Outputs:
                                ;   R3 = Address of start of command line that has been blank squeezed.
                                ;
015130  010246                  DELSPC: MOV     R2,-(SP)
015132  010346                          MOV     R3,-(SP)
015134  010302                          MOV     R3,R2
015136  112300                  1$:     MOVB    (R3)+,R0        ;GET NEXT CHAR FROM COMMAND LINE
015140  001405                          BEQ     2$              ;BR IF END OF COMMAND HIT
015142  120027  000040                  CMPB    R0,#' '         ;IS THIS CHAR A SPACE?
015146  001773                          BEQ     1$              ;IF YES THEN SKIP OVER IT
015150  110022                          MOVB    R0,(R2)+        ;MOVE CHAR INTO NEW COMMAND LINE
015152  000771                          BR      1$
015154  105012                  2$:     CLRB    (R2)            ;PUT IN ASCIZ NULL AT END
015156  012603                          MOV     (SP)+,R3
015160  012602                          MOV     (SP)+,R2
015162  000207                          RETURN

                                        .SBTTL  CHKEQ  -- Check that next command character is equal sign
                                ;-------------------------------------------------------------------
                                ;  Check to make sure the next character is an equal sign or a colon.
                                ;
015164  004767  177632          CHKEQ:  CALL    SKPSPC          ;Skip over any spaces
015170  112300                          MOVB    (R3)+,R0        ;Get next command character
015172  120027  000075                  CMPB    R0,#'=          ;Is character equal sign?
015176  001407                          BEQ     1$              ;Br if yes
015200  120027  000072                  CMPB    R0,#':          ;Is it colon?
015204  001404                          BEQ     1$              ;Br if yes
015206                                  FABORT  #EM$CSE         ;Command syntax error
015216  004767  177600          1$:     CALL    SKPSPC          ;Skip over any spaces
                                ;
                                ;  Finished
                                ;
015222  000207                          RETURN
```

```
                                        .SBTTL  CKPRIV -- Check for OPER privilege
                                ;-------------------------------------------------------
                                ;  Determine if the current user has OPER privilege.
                                ;
015224  032767  000000G 000000G CKPRIV: BIT     #PO$OPR,PRIVCO  ;Does user have OPER privilege?
015232  001004                        BNE     9$              ;Br if yes
015234                                  FABORT  #EM$OPR         ;Operator privilege required
015244  000207                  9$:     RETURN

                                        .SBTTL  CKSYPV -- Check for SYSPRV privilege
                                ;-------------------------------------------------------
                                ;  Check for SYSPRV privilege.
                                ;
015246  032767  000000G 000000G CKSYPV: BIT     #PO$SYS,PRIVCO  ;Does user have SYSPRV privilege?
015254  001004                        BNE     9$              ;Br if yes
015256                                  FABORT  #EM$SPR
015266  000207                  9$:     RETURN

                                        .SBTTL  CKTERM -- Check for TERMINAL privilege
                                ;-------------------------------------------------------
                                ;  Check to see if the user has TERMINAL privilege.
                                ;
015270  032767  000000G 000000G CKTERM: BIT     #P2$TRM,PRIVC2  ;Does user have TERMINAL privilege?
015276  001004                        BNE     9$              ;Br if yes
015300                                  FABORT  #EM$TPR         ;Terminal privilege required
015310  000207                  9$:     RETURN

                                        .SBTTL  PRGALL -- Purge all channels for job
                                ;-------------------------------------------------------
                                ;  PRGALL is called to purge all channels for the job.
                                ;  Channel 17 is not purged since it is used for TSKMON overlays.
                                ;
015312  010346                  PRGALL: MOV     R3,-(SP)
015314  012703  177777G                MOV     #NUCHN-1,R3     ;GET # OF LAST CHANNEL TO PURGE
015320  020327  000017          2$:     CMP     R3,#17          ;Don't purge channel 17
015324  001404                        BEQ     3$
015326                                  .PURGE  R3              ;PURGE ALL OF USER'S CHANNELS
015336  005303                  3$:     DEC     R3
015340  002367                        BGE     2$
015342                                  .PURGE  #RUNCHN         ;Purge channel used to start SAV files
015350  012603                        MOV     (SP)+,R3
015352  000207                        RETURN
        000001                        .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  11981 Words  ( 47 Pages)
Size of core pool:  18176 Words  ( 71 Pages)
Operating  system:  RT-11

Elapsed time:  00:02:22.08
,LP:TSKMN3=DK:TSKMN3/C/N:SYM
```

Cross reference table (CREF V05.05)

```
$1STLG   1-46
$CARUP   1-59
$CCLRN   1-60
$CFABT   1-75    22-28    23-19
$CFALL   1-82    21-24
$CFCCL   1-82    21-9
$CFDCC   1-82    22-30    23-27
$CFKIL   1-66    23-19
$CFOPN   1-88    20-9     20-77    21-16    21-24    21-81
$CFSOT   1-80    21-24
$CLTST   1-71
$CTRLC   1-74
$CTRLO   1-29
$CTRLS   1-65
$DEAD    1-125
$DEFER   1-93
$DETCH   1-63
$DIBOL   1-46
$DILUP   1-78
$DISCN   1-64
$DOOFF   1-84
$DUPRN   1-79
$ECHO    1-81
$EMTTR   1-69
$FORM    1-80
$FORMO   1-82
$HITTY   1-35
$INCOR   1-97
$INDAB   1-67    56-47
$INDDF   1-124
$INDRN   1-124    8-16    23-25
$INIT    1-125
$INKMN   1-74
$KED     1-97
$KINIT   1-31    46-12    49-14
$LC      1-81
$MLOCK   1-52
$NOIN    1-35    21-25
$NOVLN   1-42     9-13     9-14     9-21     9-22
$NOWIN   1-24     8-46     8-49
$NOWTT   1-35
$NTGCC   1-57    22-29    23-26
$PAGE    1-81
$PHONE   1-125
$PRGLK   1-61
$QTSET   1-102   20-19
$QUIET   1-94    20-21
$SCCA    1-24     8-39     8-42
$SCOPE   1-81
$SNWTT   1-122
$SPLJB   1-65
$SUCF    1-36    21-26
$TAB     1-80
$TAPE    1-120
$TECO    1-102
$TTGAG   1-88
```

Cross reference table (CREF V05.05)

```
$UCLCF    1-89
$UCLRN    1-90
$VNOTT    1-28
$VTESC    1-56
$WILD     1-103
...V1     15-53   15-57   15-61   15-68   19-39   19-39   19-39   19-44   19-44   19-47   19-47   19-51
          19-51   19-51   19-55   19-55   19-56   19-56   20-32   20-32   21-19   21-19   21-74   21-74
          21-77   21-77   23-36   23-39   27-19   27-19   27-27   27-27   31-13   32-25   36-33   37-26
          42-19   44-29   45-10   45-26   45-29   45-42   46-20   46-23   49-30   49-38   49-41   50-14
          50-23   50-25   50-29   50-33   51-40   51-42   52-27   52-32   52-34   53-26   53-32   53-33
          53-34   53-35   54-10   54-21   54-23   54-43   54-45   56-6    56-7    56-18   56-19   58-59
          66-57   69-30   69-30   69-30   69-38   69-38   81-39   81-52   81-53   81-61   81-69   81-77
          81-79   81-89   87-37   87-40
...V2     19-39   19-39   19-39#  19-39#  19-44   19-44   19-44#  19-44#  19-47   19-47#  19-51   19-51
          19-51#  19-51#  19-55   19-55   19-55#  19-55#  19-56   19-56   19-56#  19-56#  20-32   20-32
          20-32#  20-32#  21-19   21-19#  21-74   21-74   21-74#  21-74#  21-77   21-77   21-77#  21-77#
          27-19   27-19   27-19#  27-19#  27-27   27-27#  69-30   69-30   69-30#  69-30#  69-38   69-38
          69-38#  69-38#  87-37   87-37#  87-40   87-40#
ABRTAD    1-28
ABRTCD    1-28
ABRTCF    1-132   22-8#   56-46
ACRDEC    1-140   28-14#  30-16
ACRFIL    1-139   58-17#
ACRFN     1-132   57-14#
ACROCT    1-142   29-14#
ACRPRV    1-46     5-14#  13-22
ACRSPD    1-141   30-12#
ACRSTR    1-27    33-68   34-14#
ACRTXT    1-26    33-14#
ADLDAC   69-65    71-10#
ADM3A     1-116   46-37
ADM3FL    1-117
ADM3NO    1-118
AF$CCA    1-24     8-40
AF$NPW    1-24     8-47
AFCF      1-24     8-25*    8-38    20-42   21-62*
ALCDEV    1-50     1-60    59-11*  65-30*  66-16*
ALDBLK    1-155
ALDEX     1-154    1-155
ALFN      1-166
AMBOPT    1-140   12-50
ASDEX     1-137
ASKLNM    1-136
ASNEND    1-79
ASNHD1    1-149
ASNHD2    1-149
ASNOVF    1-162   73-29
ASNSRC    1-170   62-22   73-13   73-21   73-27   76-13#  77-22
ASNTBL    1-78    76-15   79-12
AT$$SZ    1-76    76-18   79-18
AT$DEV    1-76    77-29   79-14   79-17*
AT$EXT    1-76    77-34
AT$FIL    1-76    77-30   77-33
AT$LOG    1-76    76-16   79-16*
AT$SIZ    1-76    77-37
AUTHFN    1-159
```

Cross reference table (CREF V05.05)

```
BADBOT   1-168    19-53
BADCMD   1-136
BADPMT   1-143
BADPRI   1-143
BADSAV   1-136
BASMAP   1-111
BDFNAM   1-139    57-46
BDLGOP   1-144
BDLIN    1-157
BELL     1-177#
BLANK    1-176#   15-61     85-33
BLKO     1-135     4-30     19-44     19-46     33-36     33-62     34-35     34-62     58-30     58-59     69-37     81-27
BLKWDS   1-181#
BOTCSR   1-73     19-46*
BOTDEV   1-73     19-6      19-10     19-28*    19-51
BOTEMT   1-168    19-61
BOTHAN   4-54#    19-12*    19-24*    19-37*    19-39
BOTUNI   1-73     19-13*    19-29*
C.CSW    1-85     69-39     69-50
C.DEVQ   1-85     69-42
C.SBLK   1-85     69-45
CALUCL   1-148
CCLNAM   1-163
CCLSAV   1-68
CCSPRV   1-43      7-9#
CD$$SZ   1-86     59-47     63-44
CD$$UB   1-86     59-34     64-26
CD$BAS   1-86     60-18     63-42
CD$DVU   1-86     59-20     60-16     60-28     63-40
CD$JOB   1-86     59-33     61-18     64-25
CD$NAM   1-86
CD$TOP   1-102
CDBUF    1-45     59-20     59-33     60-16     60-18     60-28     61-18     63-40     63-42     64-25     67-16
CDGADR   1-45     67-13*
CDGEMT   1-45     67-14
CDGET    1-45     59-19     63-39     67-13#
CDJFLG  59-22     61-13#    64-19
CDPADR   1-45     67-28*
CDPEMT   1-45     67-29
CDPUT    1-45     59-32     67-28#
CF$IND   1-126
CF$QUT   1-126
CF2DEP   1-168    20-98
CFABLV   1-127
CFACFL   1-26     24-14     24-38
CFBLK    1-94     20-34     20-85*    21-72*    21-77
CFBUF    1-68     20-83     21-77     25-22     25-41
CFCHAN   1-93     20-32     20-32     21-19     21-19     21-74     21-74     21-77     21-77
CFEND    1-68
CFHOLD   1-101    21-15*
CFIND    1-84     20-14*    20-17*    20-46     20-47*    21-14     21-54*
CFLFL4   1-89     20-18     21-56
CFNEST   1-87     20-78*    21-10     21-21     21-80*    22-17
CFOVFL  20-29     20-57     20-70     20-98#
CFPNT    1-94      8-14     20-15     20-35     21-12     22-19     23-15     24-9*     24-32*    25-23     25-36*
CFSEND   1-88     20-28     20-56     20-69
```

Cross reference table (CREF V05.05)

```
CFSP      1-88      20-27     20-76*    21-35     21-79*
CFSPND    1-100     22-10     22-13*    23-11     23-14*
CFSQEZ    1-33      25-16#
CFSTK     1-31
CFSTOP    1-26      21-27     24-5#
CFSTRT    1-26      20-84     21-71     22-12     23-13     24-28#
CFSTS     1-126
CHAIN     1-74
CHKALC    1-30      66-10#
CHKCLU    1-26      75-14#
CHKDEV    1-163     26-24     62-14#    63-20     66-21     80-58
CHKDLM    1-153     35-61     57-26     82-9#
CHKEQ     1-43      13-18     86-29#
CHKMNT    1-138     63-14#
CHKMTX    1-138     64-12#
CHKTTD    1-157     18-11#
CINDAT    1-110
CINFLG    1-28
CKACOJ    1-39      16-13#
CKCLUS    1-32      65-11#
CKLDAC   70-14#     72-9
CKPRIV    1-140     87-5#
CKSYPV    1-142     87-14#
CKTERM    1-140     87-23#
CL$XLN    1-27      65-15     65-17
CLDEVX    1-93      66-48
CLRPRV    1-43       5-19      6-8#
CMDBUF    1-131      1-148
CMDCCL    1-163
COAD      1-154      1-155
COAF      1-167
COAL      1-154      1-155
COLOO     1-148     50-25
COMMA     1-180#    15-53     85-35     85-41
CONFIG    1-83
CORUSR    1-29       8-13      9-9      16-17     16-36     20-8      21-8      22-9      23-7      56-43     61-13     66-51
CPUAH     1-148     51-63*    51-72*    54-10     54-11
CPUAL     1-148     51-64*    51-71*    54-12
CPUMSG    1-169     50-29
CR        1-175#
CRLF      1-143     15-57     50-33     54-45     66-57     81-89
CS$RON    1-55      69-50
CSHDEV    1-78      59-17     63-37
CSHDVN    1-78      59-48     63-45
CSHHD     1-130
CSIMS1    1-142
CSIMS2    1-140
CSIMS4    1-162
CTMSG     1-169     50-14
CTRLTT    1-72
CURMTX    1-153
CURPRM    1-99      20-49     20-86*    21-52*
CVDVNM    1-150     60-29     74-12#
CVTTAB    1-131     83-14#
CVTUC     1-127     84-8#
CW$5OH    1-83
```

Cross reference table (CREF V05.05)

```
CXTBAS   1-77
CXTPAG   1-54
CXTRMN   1-26     24-13     24-37
CXTWDS   1-77
DATTIM   1-33     54-38#
DCCRD    1-106
DCCWR    1-106
DCTRD    1-106
DCTWR    1-106
DEADEV   1-138    79-10#
DELSPC   1-152    86-11#
DETARG   1-167
DETHD    1-167
DETTXT   1-146
DEVHD1   1-149
DEVIDL   1-154     1-154     1-155
DEVUNT   1-141
DFJMEM   1-31
DIABFL   1-94
DIABLO   1-115    46-40
DIABNO   1-95
DIV32   36-23     40-14#
DIVIDE   1-147    39-8#     50-20     51-23     51-29     51-34     51-54     52-15     52-17     52-19     54-14     54-16
        54-18
DIVSOR   1-160    40-22     40-25     40-27     40-29
DJABMS   1-158
DKSAV    1-131
DLCEMT   1-50     59-12
DLLDAC  69-19     72-9#
DLMSG    1-158
DLTXT    1-145
DMTALL   1-158    59-5#
DMTARG   1-137    60-34     68-25
DMTSUB   1-163    59-43     60-10#
DOASGN   1-59     73-9#
DOSTOP   1-159    19-6#
DZTXT    1-165
EDIT     1-46
EDTFIL   1-151    47-14#
EDTR50  47-20     47-27     47-30     47-37     48-14#
EM$CAJ   1-39     16-58
EM$CSE   1-44      5-46     86-35
EM$DAA   1-60     66-39
EM$DIU   1-60     66-55
EM$ENM   1-49     28-70
EM$IOV   1-49     29-65
EM$IST   1-28     33-78     34-70
EM$ISV   1-49     30-37
EM$OPR   1-38     87-7
EM$SPR   1-38     87-16
EM$TPR   1-38     87-25
ERRLOC   1-30     66-35     66-37
ERRSEV   1-103    56-44
FC$CDX   1-130
FC$LNK   1-130
FD$NAM   1-130
```

Cross reference table (CREF V05.05)

```
FF        1-179#  55-31   83-19
FILNAM    1-132    1-133  57-14*  57-15*  57-16*  57-23*  57-30*  57-32*  57-33   57-35*  57-41*  58-65
FIXPRV    1-43     8-34    9-7#
FKILL     1-132    5-46   12-50   12-51   19-53   20-98   28-70   29-65   30-37   33-78   34-70   56-29#
         73-29    86-35   87-7    87-16   87-25
FORCEO    1-170   78-8#
FPRINT    1-132   16-58   27-21   56-6#   56-29   57-46   66-39   66-55   71-46
FSTDL     1-63
GAGMSG    1-157
GENTOP    1-73
GETKCH    1-50    85-66#
GRT1      1-112
GTRD50    1-137   12-21   35-10#  57-18   57-40
H.CSR     1-158   19-46
HANBSY    1-142
HANCHN    1-182#
HANENT    1-51
HANIDX    1-141
HANSIZ    1-51
HAZEL     1-52    46-36
HAZLFL    1-52
HAZLNO    1-52
HIMAP     1-111
HIPRI     1-143
HNBUF     1-141
HUPARG    1-162
II$$SZ    1-25    80-44
II$NAM    1-25    80-34   80-55
IIBUF     1-25    80-34   80-55
ILLCMD    1-138
ILSW2     1-58
IN$ACT    1-124    8-18   20-12   22-21
IN$CMD    1-124
IN$CNT    1-124    8-18
INDABT    1-127   23-6#   56-49
INDACT    1-134
INDCOL   15-16*   15-58   15-84#
INDERR    1-75    56-20*  56-42*
INDRDB    4-34#   23-35*  23-36   23-38*  23-39
INDSAV    1-124
INDSTA    1-75     8-18   20-12   20-14   20-17   20-47   20-93*  21-14*  22-21   23-24*
INFOMT    1-138
INGADR    1-25    80-31*
INGEMT    1-25    80-32
INSSPC    4-55#   80-21   80-25   80-35
INSSRC    1-170   80-14#
INSTBL    1-25    80-30
INSTBN    1-25    80-45
INVDAT    1-167
INVDEV    1-166
INVEC     1-125
INVLDM    1-162
INVLDN    1-139
INVOPT    1-132    1-138   1-153   12-51
INVTIM    1-158
IOABFL    1-29
```

Cross reference table (CREF V05.05)

```
ITRMTP    1-126     46-14
JCXPGS    1-111
JCXSMS    1-165
JSTKND    1-72
JSWLOC    1-30
K52       1-46
KBMSG     1-147
KBTX      1-151
KCSIBF    1-136     58-50      58-52      58-59
KED       1-46
KEYBUF    1-169     55-38      55-80      55-89
KEYEND    1-169     55-64
KILEMT    1-159
KL3CLR    1-61
KL4CLR    1-96
KMFTXT    1-169     56-6
KMNBAS    1-121
KMNCHN    1-68
KMNERR   56-8       56-31      56-41#
KMNHI     1-53
KMNNAM    1-163
KMNPGS    1-54
KMNSTK    1-54
KMNSTR    1-54
KMNTOP    1-54
KMPRMT    1-107
KMSTK     1-170     56-30
KUSECK    1-49      66-53
L         3-12       3-12#     3-13       3-13#     3-14       3-14#     3-15       3-15#     3-16       3-16#     3-17       3-17#
          3-18       3-18#     3-19       3-19#     3-20       3-20#     3-21       3-21#     3-22       3-22#     3-23       3-23#
          3-24       3-24#     3-25       3-25#     3-26       3-26#     3-27       3-27#     3-28       3-28#     3-29       3-29#
          3-30       3-30#     3-31       3-31#     3-32       3-32#     3-33       3-33#     3-34       3-34#     3-35       3-35#
          3-36       3-36#     3-37       3-37#     3-38       3-38#     3-39       3-39#     3-40       3-40#     3-41       3-41#
          3-42       3-42#     3-43       3-43#     3-44       3-44#     3-45       3-45#     3-46       3-46#     3-47       3-47#
          3-48       3-48#     3-49       3-49#     3-50       3-50#     3-51       3-51#     3-52       3-52#     3-53       3-53#
          3-54       3-54#     3-55       3-55#     3-56       3-56#     3-57       3-57#     3-58       3-58#     3-59       3-59#
          3-60       3-60#     3-61       3-61#     3-62       3-62#     3-63       3-63#     3-64       3-64#     3-65       3-65#
          3-66       3-66#     3-67       3-67#     3-68       3-68#     3-69       3-69#     3-70       3-70#     3-71       3-71#
          3-72       3-72#     3-73       3-73#     3-74       3-74#     3-75       3-75#
LA120     1-115     46-39
LA12FL    1-116
LA12NO    1-116
LA36      1-115     46-38
LA36FL    1-95
LA36NO    1-95
LACTIV    1-55
LAFSIZ    1-58
LCBIT     1-115
LCOL      1-57       1-102
LCONTM    1-65      50-16
LCPUHI    1-65      50-30
LCPULO    1-65      50-31
LD$RON    1-91      69-52      71-39
LDBASE    1-92      26-45      60-18      63-29      69-45*     69-61      69-61*
LDCLEN    1-33      68-6#
LDDEVX    1-93      26-30      62-77      63-25      66-22      69-57      70-23      71-31      80-60
```

Cross reference table (CREF V05.05)

```
LDFLAG    1-92      69-52*    71-39
LDMNT     1-34      68-32     69-14#
LDNAM     1-136
LDNAME    1-92      68-19     69-26
LDOPHD    1-144
LDPDEV    1-92      26-44     60-16     62-81     63-28     68-36     69-41*    69-44*    69-60     69-60*    69-74*
LDSIZE    1-92      26-47     69-33*
LF        1-174#
LF$OPN    1-129     26-18     27-6      27-26
LF$WRT    1-129
LFWLIM    1-58
LGOVER    1-168     27-21
LICTXT    1-112
LINBUF    1-53
LINCNT    1-55
LINCUR    1-58
LINFRE    1-158
LINNXT    1-53
LINPNT    1-55
LJSW      1-72
LMXLN     1-125
LMXNUM    1-123
LMXPRM    1-126
LNBLKS    1-77
LNMAP     1-83
LNPRIM    1-83      16-37     16-37
LNSBLK    1-78
LNSPAC    1-87
LOCKTX    1-147
LOGASN    1-139     77-16#    80-26
LOGBAS    1-59      26-50     26-52
LOGBLK    1-128     27-19
LOGBUF    1-128     27-16     27-19
LOGCHK    1-59      26-11#
LOGCHN    1-128     27-19     27-19     27-27     27-27
LOGCLS    1-143     26-59     27-5#
LOGDVU    1-59      26-37     26-48
LOGFLG    1-128     26-18     27-6      27-26*
LOGPTR    1-128     27-11     27-25*
LOMAP     1-111
LOTBUF    1-56
LOTNXT    1-56
LOTPNT    1-56
LOTSIZ    1-57
LOTSPC    1-57
LPARNT    1-35      16-39
LPRG1     1-113
LPRG2     1-113
LPROG     1-64      16-51     16-51     49-42
LPROJ     1-64      16-44     16-44     49-36
LRBFIL    1-84
LRDTIM    1-55
LSCCA     1-80
LSECPT    1-69
LSTACT    1-53
LSTATE    1-109
```

Cross reference table (CREF V05.05)

```
LSTDL     1-63      16-24
LSTMX     1-123
LSTPL     1-104     16-22
LSTPRM    1-100     20-63     20-89     21-41
LSTSL     1-109
LSTSPL    1-28      81-9#
LSUCF     1-60
LSW       1-29      46-12     49-14
LSW11     1-24       8-46*     8-49*
LSW2      1-74       9-13*     9-21*     20-19
LSW2S     1-42       9-14*     9-22*
LSW3      1-79      21-25*
LSW4      1-96      20-9      20-18*    20-21*    20-44     20-77*    21-9*     21-16     21-24*    21-57*    21-60*    21-81*
         22-30*    23-27*
LSW5      1-61       8-16      8-39*     8-42*     23-25*
LSW6      1-122     22-28*    23-19*
LSW7      1-126     56-47
LSW9      1-52      21-26*    22-29*    23-26*
LTRMTP    1-116     46-11
LTSCMD    1-87
LUNAME    1-64      49-27
LWINDO    1-24
MAXASN    1-75      76-14     79-13
MAXAVL    1-141
MAXMEM    1-30
MAXMTX    1-153
MAXSEC    1-69
MDT       1-52
MINTIM    1-69      50-15
MISSEQ    1-142
MNTARG    1-162     68-39
MNTDEV    1-137     60-33*    68-24*    68-38*
MNTFUL    1-139
MNTTXT    1-164
MONAR1    1-152
MONAR2    1-152
MONHD     1-152
MONTAB    1-169     53-31
MONVEC    1-127
MSGBUF    1-157
MSGEND    1-157
MTOPHD    1-138
MUL32     1-167     36-19     41-14#    51-62
MXCSR     1-123
MXDTR     1-123
MXJADR    1-32
MXJMEM    1-31
MXPRMT    1-107
MXVEC     1-125
NAMTOP    1-112
NARGS     3-11#     3-12      3-12      3-12      3-13      3-13      3-13      3-14      3-14      3-14      3-15      3-15
          3-15      3-16      3-16      3-16      3-17      3-17      3-17      3-18      3-18      3-18      3-19      3-19
          3-19      3-20      3-20      3-20      3-21      3-21      3-21      3-22      3-22      3-22      3-23      3-23
          3-23      3-24      3-24      3-24      3-25      3-25      3-25      3-26      3-26      3-26      3-27      3-27
          3-27      3-28      3-28      3-28      3-29      3-29      3-29      3-30      3-30      3-30      3-31      3-31
          3-31      3-32      3-32      3-32      3-33      3-33      3-33      3-34      3-34      3-34      3-35      3-35
```

```
                3-35    3-36    3-36    3-36    3-37    3-37    3-37    3-38    3-38    3-38    3-39    3-39
                3-39    3-40    3-40    3-40    3-41    3-41    3-41    3-42    3-42    3-42    3-43    3-43
                3-43    3-44    3-44    3-44    3-45    3-45    3-45    3-46    3-46    3-46    3-47    3-47
                3-47    3-48    3-48    3-48    3-49    3-49    3-49    3-50    3-50    3-50    3-51    3-51
                3-51    3-52    3-52    3-52    3-53    3-53    3-53    3-54    3-54    3-54    3-55    3-55
                3-55    3-56    3-56    3-56    3-57    3-57    3-57    3-58    3-58    3-58    3-59    3-59
                3-59    3-60    3-60    3-60    3-61    3-61    3-61    3-62    3-62    3-62    3-63    3-63
                3-63    3-64    3-64    3-64    3-65    3-65    3-65    3-66    3-66    3-66    3-67    3-67
                3-67    3-68    3-68    3-68    3-69    3-69    3-69    3-70    3-70    3-70    3-71    3-71
                3-71    3-72    3-72    3-72    3-73    3-73    3-73    3-74    3-74    3-74    3-75    3-75
                3-75
NATXT           1-145
NEGFLG          4-59#   28-26*  28-29*  28-62   29-26*  29-29*  29-57
NEWJSW          1-72
NFRESB          1-98
NINTXT          1-145
NLINES          1-85
NOCCL           1-144
NOCIN           1-136
NODAT           1-150
NOFIL           1-148    1-156
NOFLAG          1-138
NOFRDL          1-167
NOIND           1-142
NOKMON          1-163
NOLDMT          1-150
NONEMS          1-150
NOOPTT          1-156
NOPMGN          1-152
NOPRG           1-132     1-136
NOSTRT          1-135
NOTAVL          1-151
NOTON           1-157
NOTXT           1-145
NOUDC           1-149
NSWPMS          1-153
NTRMTP         46-15    46-44#
NUCHN           1-84    87-34
NUMDEV          1-118   62-57
NUMON           1-58
OCTFIX          1-145   32-9#
OCTPRT          1-165   31-7#
OF$$SZ          1-120   70-27   71-26   71-50   72-22   72-25
OF$DEV          1-119   70-23   71-31*  72-15*
OF$FIL          1-119   71-24   71-34*  71-35*  71-36*  72-17*
OF$FLG          1-119   71-37*  71-41*
OF$UNT          1-119   70-25   71-32*  72-16*
OFFEMT          1-159
OKFAND          1-70    70-28   71-51   71-53*  72-23   72-25*
OKFEND          1-70
OKFILE          1-70    70-22   71-21
OKFNND          1-70    71-22
OPTLST          1-47    10-10#
OT$RON          1-120   71-41
OTHRON          1-161
OTRMNT          1-163
```

Cross reference table (CREF V05.05)

```
OVRCOR   1-135
PO$$NP   1-38    14-41    14-43
PO$ALC   1-42     3-13     3-14
PO$BYP   1-37     3-15     3-16
PO$DBG   1-39     3-17     3-18
PO$DET   1-40     3-19     3-20    16-26
PO$LOK   1-40     3-23     3-24     3-35     3-36
PO$MEM   1-40     3-25     3-26
PO$NAM   1-41     3-43     3-44
PO$NFR   1-37     3-29     3-30
PO$NFW   1-37     3-31     3-32
PO$OPR   1-40     3-33     3-34    87-5
PO$RT    1-40     3-37     3-38
PO$SND   1-41     3-41     3-42
PO$SPF   1-37     3-47     3-48
PO$SPV   1-39     1-41     3-45     3-46
PO$SYS   1-41     3-51     3-52    87-14
P2$$NP   1-38    14-42    14-44
P2$CGR   1-41     3-49     3-50
P2$CXT   1-36     3-21     3-22
P2$GRP   1-39     3-57     3-58    16-46
P2$MSG   1-40     3-27     3-28
P2$RLK   1-41     3-39     3-40
P2$SAM   1-39     3-59     3-60    16-53
P2$TRM   1-38     3-53     3-54    87-23
P2$UP1   1-36     3-65     3-66
P2$UP2   1-36     3-67     3-68
P2$UP3   1-36     3-69     3-70
P2$UP4   1-36     3-71     3-72
P2$VIR   1-42     3-61     3-62     3-63     3-64     9-19
P2$WRL   1-39     3-55     3-56    16-31
PASLIN   1-103
PAUMSG   1-131
PBFEND   1-88    20-50    20-91*   21-47*
PBUFND   1-168   42-11    44-17
PEKADR   1-27     4-28#   81-23*
PEKEMT   1-27     4-26#   81-25
PEKSIZ   1-27     4-29#   81-24*
PF$IOW   1-122
PF$SYS   1-122
PFCO     1-44     4-42#    6-11    14-20    14-26    14-43*   14-44*   14-62
PFLALL   3-12    14-50#
PFLNON   3-73    14-61#
PFLRTN   3-13     3-14     3-15     3-16     3-17     3-18     3-19     3-20     3-21     3-22     3-23     3-24
         3-25     3-26     3-27     3-28     3-29     3-30     3-31     3-32     3-33     3-34     3-35     3-36
         3-37     3-38     3-39     3-40     3-41     3-42     3-43     3-44     3-45     3-46     3-47     3-48
         3-49     3-50     3-51     3-52     3-53     3-54     3-55     3-56     3-57     3-58     3-59     3-60
         3-61     3-62     3-63     3-64     3-65     3-66     3-67     3-68     3-69     3-70     3-71     3-72
        14-13#
PFLSTD   3-74     3-75    14-41#
PFSO     1-44     4-41#    6-10    14-17    14-29    14-41*   14-42*   14-51
PMBUSY   1-152
PNAME    1-118    1-141    19-23    19-36    62-58    74-17
POPCF    1-102   21-7#     22-23    23-17
PPNMSG   1-169   49-38
PRGALL   1-34    87-33#
```

Cross reference table (CREF V05.05)

```
PRGSIZ   1-53
PRGTOP   1-53
PRIVC0   1-44     7-12     8-30     16-26    87-5     87-14
PRIVC2   1-42     9-18     16-31    16-46    16-53    87-23
PRIVF0   1-44     8-21     8-29     20-37    20-39    21-64    21-66
PRIVS0   1-44     7-13     8-20
PRMBUF   1-100    20-53    20-91    21-48
PRMEND   1-100
PRMPNT   1-99     20-66    20-73    20-87    21-36
PRTBUF   1-150
PRTDAT   1-166    53-6#    54-42
PRTDC2   1-147    45-7#    50-21    50-24    51-38    51-41    51-44    52-30    52-33    52-36    53-40    54-19
         54-22    54-25
PRTDC3   1-147    45-24#
PRTDEC   1-141    36-28    42-6#    45-12    45-30    49-39    49-43    52-25    53-25    66-56
PRTFIX   1-144    43-14    44-12#   81-38    81-60    81-85
PRTFNM   1-150    38-14#
PRTLN    1-146    43-10#
PRTPCT   1-160    36-10#
PRTR50   1-166    37-8#    81-44    81-66    81-68
PRTSPC   1-144    45-40#
PRTTIM   1-166    50-7#
PRTTMD   1-151    52-9#
PRTTMV   1-148    50-32    51-14#
PRTTOD   1-166    54-6#    54-44
PRTTTP   1-145    46-9#
PRTUNM   1-146    49-9#
PRTWRN   1-133    26-58    56-18#
PRVHD    3-11#     5-33    15-38
PRVLST   1-43     15-16#
PRVOPT   1-27     13-14#
PUSHCF   1-133    20-7#
PVNPW    1-44      6-12     7-14     8-22     8-31     14-52    14-63    15-23    20-28    20-37    21-66
PVON     1-61
QUME     1-115    46-41
QUMEFL   1-117
QUMENO   1-117
R$CFST   1-26     24-14*   24-38*
R.GSTS   1-47     23-35*   23-38*
R50BUF   1-137    12-22    35-55*   35-57*   57-23    57-30    57-32    57-41
R50C1    4-51#    75-31
R50C10   4-52#    65-27    75-35    75-39
R50C17   4-53#    75-37
R50CHR   1-168    37-18    38-28    48-33
R50CL    4-48#    75-18
R50CL0   4-49#    65-29    75-22    75-26
R50CL7   4-50#    75-24
R50DIR   1-133
R50DK    1-166    62-45    77-45
R50DSK   1-139
R50DUP   1-134
R50IND   1-133
R50K52   1-134
R50KED   1-134
R50KEX   1-134
R50LD    1-139
```

Cross reference table (CREF V05.05)

```
R50LD0    1-137     60-14     68-11
R50LD7    1-139
R50LOG    1-143
R50MON    1-167
R50NO     1-140     12-22
R50PIP    1-134
R50SY     1-133     62-47     77-43
R50TT     1-162     18-11
R50TTO    4-46#     18-13
R50TT7    4-47#     18-15
R50VIR    1-135
RDB       1-108
RDBEND    1-108
RDCMD     1-131     56-32     66-58
RDERM     1-135
REMNDR    1-168     40-32*    40-33*
RESDEV    1-120     70-16     71-10
RNMS      1-146
RONTXT    1-151
RS.CRR    1-47
RS.EGR    1-47      23-38
RS.GBL    1-47      23-35
RS.PVT    1-47      23-35
RSR       1-123
RSTPRV    1-43       8-6#     21-28     21-68     22-35     23-31
RT$$SZ    1-108
RT$NAM    1-108
RUNCHN    1-50      87-40     87-40
RUNDEV    1-77      80-57
RUNEMT    1-135
RUNFLG    1-37       8-38*     8-40      8-47
RUNHD     1-131
RUNMS     1-167
S$INWT    1-67
S$IOFN    1-62
S$IOWT    1-113
S$MSWT    1-68
S$OTFN    1-62
S$OTLO    1-62
S$OTWT    1-67
S$QUSR    1-113
S$SFWT    1-67       1-113
S$SPCB    1-114
S$SPDB    1-114
S$SPND    1-61
S$TMWT    1-67
S$TTFN    1-62
S$TWFN    1-62
SC$SEV    1-71      56-42     56-44
SC$WRN    1-69      56-20
SCHAIN    1-93
SCNOPS    1-49      11-10#
SD$BAK    1-96
SD$DEL    1-89
SD$FLK    1-90
SD$HLD    1-99
```

Cross reference table (CREF V05.05)

```
SD$SNG    1-98
SD$WFM    1-90
SDBLK     1-91
SDBU      1-96
SDBUF1    1-91
SDCB      1-104    17-11
SDCBND    1-104    17-12
SDCBSZ    1-109    17-16
SDFHD     1-101    17-14     81-14
SDFLAG    1-90
SDFORM    1-90
SDNAME    1-109     1-153    81-43
SDSFCB    1-89
SDSKIP    1-96
SEARCH    1-131    12-32     55-19#
SERFLG    1-29     68-26*    68-28*    69-29*    69-31*
SETHD     1-140
SETWRD    1-46      5-34     10-21     11-31     12-13#
SF$1ST    1-99     81-31
SF$BSY    1-98     81-49
SF$HLD    1-99
SFCBSZ    1-48     81-24
SFFILE    1-114    81-65     81-67
SFFLAG    1-101    81-31     81-49
SFFORM    1-98     81-74
SFID      1-50     81-36
SFNMBL    1-98     81-83
SFQLNK    1-101    81-93
SFUSER    1-114    81-57
SH$$SZ    1-48
SH$FLG    1-48
SH$NAM    1-48
SH$RTN    1-48
SH$VAL    1-48
SHMTH1    1-149
SHMTH2    1-149
SHOHD     1-164
SHTMSG    1-159
SIZEMT    1-162
SIZVAL    1-136
SJSPPN    1-170     4-60#     49-19
SKPDLM    1-71     85-30#
SKPSPC    1-71      5-24      5-38     10-14     11-14     12-19     12-25     28-18     28-22     28-31     29-18     29-22
         29-31     33-19     33-23     34-19     34-23     85-11#    85-12     85-40     86-29     86-36
SO$NO     1-51
SO$NVL    1-51
SO$OCT    1-51
SPACE2    1-146    54-43     81-39     81-53     81-61     81-69     81-79
SPACE3    1-146
SPACE5    1-147
SPACE6    1-150
SPACTV    1-154     1-155
SPCF      1-156
SPDVAL    4-7#     30-30
SPFLK     1-156
SPFUL     1-156
```

Cross reference table (CREF V05.05)

```
SPGEMT   1-156
SPLACT   1-159    17-10#
SPLCHN   1-66
SPLHD    1-153
SPLHLA   1-144
SPLND    1-91
SPLPND   1-161
SPSNG    1-154     1-156
SPUBUF   1-32
SPWFM    1-154     1-155
SRTSMS   1-161
SRTTXT   1-164
START    1-168    19-55     19-56
STLGHD   1-143
STPASK   1-161
STPFLG   1-66
SUBARG   1-151
SUBTXT   1-164
SUCS     1-112
SUM1     1-160
SUM2     1-160
SUM3     1-160
SUM4     1-160
SUM5     1-161
SUM6     1-161
SUM7     1-161
SUMS     1-112
SUPCOD   1-112
SWPTX    1-147
SXBPNT   1-32
SYHD1    1-146
SYHD2    1-146
SYINDX   1-118    19-22     19-33     62-49
SYNAME   1-119    77-47
SYSAV    1-131
SYSDAT   1-110
SYTIMH   1-110
SYTIML   1-110
SYUNIT   1-118    19-25     19-34     62-52
TAB      1-178#   55-29     83-17
TALEMT   1-60     65-31     66-27
TBLOVF   1-140    71-46
TECO     1-46
TK1SEC   1-57     51-22     51-53     54-13
TK1VAL   1-110    51-70
TK5VAL   1-97     51-20
TM$CLG   1-49     26-58
TMIDLH   1-34
TMIOH    1-34
TMIOWH   1-33
TMSWPH   1-34
TMSWTH   1-34
TMTOTH   1-33      1-160
TMTOTL   1-33      1-160
TMUSRH   1-33
TOTMMS   1-164
```

Cross reference table (CREF V05.05)

```
TOTON     1-66
TOTXT     1-143
TRGRET    1-112
TRMSTR    1-133
TSKMN3    1-5#        1-29
TSR       1-123
TSXLN     1-112
TSXSIT    1-112
TSXSMS    1-165
TTFTBL   46-16       46-34#     46-44
TTNADM   46-51       46-67#
TTNDIA   46-54       46-70#
TTNHZL   46-50       46-66#
TTNL12   46-53       46-69#
TTNL36   46-52       46-68#
TTNQUM   46-55       46-71#
TTNTBL   46-22       46-48#
TTNV10   46-49       46-64#
TTNV20   46-56       46-57      46-65#
TTNV52   46-48       46-63#
TTNXXX   46-20       46-62#
UC$MDC    1-130
UC$NDC    1-130
UCHAN     1-82
UCLBLK    1-129
UCLDAT    1-129
UCLNAM    1-87
UERSEV    1-103      22-31*     23-20*
UFORM     1-74
UFPTRP    1-89
UHIMEM    1-77
UMSSMS    1-164
UMSYTP    1-63
UPTMMS    1-159
USPLCH    1-66
USRMMS    1-165
USRSTK    1-31
USTART    1-73
UTRPAD    1-30
VCORTM    1-107
VHIPCT    1-105
VIMAGE    1-72
VQUAN1    1-105
VQUAN2    1-105
VQUN1A    1-105
VT100     1-115      46-35
VT10FL    1-117
VT10ND    1-117
VT2007    1-114      46-42
VT2008    1-114      46-43
VT52      1-115      46-34
VT52FL    1-116
VT52ND    1-95
WILDFL    1-35
WLDNAM    1-35       48-19      71-33      80-37      80-55
WRNHED    1-134      56-18
```

Cross reference table (CREF V05.05)

```
XAREA     1-132     19-39     19-44     19-51     19-55     19-56     20-32     21-74     21-77     23-36     23-39     27-19
         54-10      69-30     69-38
YESTXT    1-145
ZCLR      1-123
```

Cross reference table (CREF V05.05)

```
...CM0   58-59    58-59    58-59
...CM1   19-39    19-44    19-51    19-55    19-56    20-32    21-74    21-77    27-19    69-30    69-38
...CM2   19-39    19-39    19-44    19-44    19-44    19-44    19-51    19-51    19-55    19-55    19-55    19-55
         19-56    19-56    19-56    19-56    20-32    21-74    21-77    21-77    21-77    21-77    23-36    23-39
         27-19    27-19    27-19    27-19    54-10    69-30    69-30    69-38
...CM3   19-47    21-19    27-27    87-37    87-40
...CM5   15-53    15-57    15-61    15-68    19-39    19-44    19-51    19-55    19-56    20-32    21-74    21-77
         23-36    23-39    27-19    31-13    32-25    36-33    37-26    42-19    44-29    45-10    45-26    45-29
         45-42    46-20    46-23    49-30    49-38    49-41    50-14    50-23    50-25    50-29    50-33    51-40
         51-42    52-27    52-32    52-34    53-26    53-32    53-33    53-34    53-35    54-10    54-21    54-23
         54-43    54-45    56-6     56-7     56-18    56-19    66-57    69-30    69-38    81-39    81-52    81-53
         81-61    81-69    81-77    81-79    81-89
...CM6   23-36    23-39    54-10
...CM7   19-44    19-55    19-56    21-77    27-19
.CLOSE    1-17#   19-47    21-19    27-27
.CRRG     1-20#   23-36
.CSIGE    1-15#
.CSISP    1-13#   58-59
.DATE     1-16#   53-9     54-39
.ELRG     1-20#   23-39
.ENTER    1-18#
.EXIT     1-18#
.FPROT    1-19#
.GTIM     1-16#   54-10
.GTLIN    1-16#
.GVAL     1-19#
.HERR     1-19#
.LOOKU    1-17#   19-39    19-51    69-30
.PRINT    1-17#   15-57    42-19    44-29    46-20    46-23    49-38    50-14    50-25    50-29    50-33    54-43
         54-45    56-6     56-7     56-18    56-19    66-57    81-39    81-53    81-61    81-69    81-79    81-89
.PURGE    1-14#   87-37    87-40
.PVAL     1-19#
.READW    1-14#   19-44    19-55    19-56    21-77
.REOPE    1-15#   21-74
.SAVES    1-15#   20-32    69-38
.SERR     1-19#   19-38
.SRESE    1-13#
.TTOUT    1-13#
.TTYIN    1-14#
.TTYOU    1-14#   15-53    15-61    15-68    31-13    32-25    36-33    37-26    45-10    45-26    45-29    45-42
         49-30    49-41    50-23    51-40    51-42    52-27    52-32    52-34    53-26    53-32    53-33    53-34
         53-35    54-21    54-23    81-52    81-77
.WRITW    1-18#   27-19
CMDDEF    2-47#    3-12     3-13     3-14     3-15     3-16     3-17     3-18     3-19     3-20     3-21     3-22
          3-23     3-24     3-25     3-26     3-27     3-28     3-29     3-30     3-31     3-32     3-33     3-34
          3-35     3-36     3-37     3-38     3-39     3-40     3-41     3-42     3-43     3-44     3-45     3-46
          3-47     3-48     3-49     3-50     3-51     3-52     3-53     3-54     3-55     3-56     3-57     3-58
          3-59     3-60     3-61     3-62     3-63     3-64     3-65     3-66     3-67     3-68     3-69     3-70
          3-71     3-72     3-73     3-74     3-75
FABORT    2-15#    5-46     12-50    12-51    19-53    20-98    28-70    29-65    30-37    33-78    34-70    73-29
         86-35    87-7     87-16    87-25
FERR      2-4#    16-58    27-21    57-46    66-39    66-55    71-46
FWARN     2-23#   26-58
TBLDEF    2-35#    3-11
TBLEND    2-62#    3-76
```