```
TSKMN3 -- TSKMON Subroutines    MACRO V05.04  Monday 21-Dec-87 11:41
Table of contents
```

Table of contents

```
                                .TITLE  TSKMN3 -- TSKMON Subroutines
                                .ENABL  LC
                                .DSABL  GBL
 000000                         .CSECT  TSKMN3
 000000                 TSKMN3:
                        ;
                        ;   Copyright 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985.
                        ;   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,$CLTST,SKPSPC,SKPDLM,SC$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  BLK0,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  SUBARG,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,P0$ALC,0,+1
      000026                            CMDDEF  NOALLO*CATE,PFLRTN,P0$ALC,0,-1
      000040                            CMDDEF  BYP*ASS,PFLRTN,P0$BYP,0,+1
      000052                            CMDDEF  NOBYP*ASS,PFLRTN,P0$BYP,0,-1
      000064                            CMDDEF  DEB*UG,PFLRTN,P0$DBG,0,+1
      000076                            CMDDEF  NODEB*UG,PFLRTN,P0$DBG,0,-1
      000110                            CMDDEF  DET*ACH,PFLRTN,P0$DET,0,+1
      000122                            CMDDEF  NODET*ACH,PFLRTN,P0$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,P0$LOK,0,+2
      000172                            CMDDEF  NOMEML*OCK,PFLRTN,P0$LOK,0,-2
      000204                            CMDDEF  MEMM*AP,PFLRTN,P0$MEM,0,+1
      000216                            CMDDEF  NOMEMM*AP,PFLRTN,P0$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,P0$NFR,0,+1
      000266                            CMDDEF  NONFSR*EAD,PFLRTN,P0$NFR,0,-1
      000300                            CMDDEF  NFSW*RITE,PFLRTN,P0$NFW,0,+1
      000312                            CMDDEF  NONFSW*RITE,PFLRTN,P0$NFW,0,-1
      000324                            CMDDEF  OP*ER,PFLRTN,P0$OPR,0,+1
      000336                            CMDDEF  NOOP*ER,PFLRTN,P0$OPR,0,-1
      000350                            CMDDEF  PSWAP*M,PFLRTN,P0$LOK,0,+1
      000362                            CMDDEF  NOPSWAP*M,PFLRTN,P0$LOK,0,-1
      000374                            CMDDEF  REAL*TIME,PFLRTN,P0$RT,0,+1
      000406                            CMDDEF  NOREAL*TIME,PFLRTN,P0$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,P0$SND,0,+1
      000456                            CMDDEF  NOSEN*D,PFLRTN,P0$SND,0,-1
      000470                            CMDDEF  SETNA*ME,PFLRTN,P0$NAM,0,+1
      000502                            CMDDEF  NOSETNA*ME,PFLRTN,P0$NAM,0,-1
      000514                            CMDDEF  SETP*RV,PFLRTN,P0$SPV,0,+1
      000526                            CMDDEF  NOSETP*RV,PFLRTN,P0$SPV,0,-1
      000540                            CMDDEF  SPF*UN,PFLRTN,P0$SPF,0,+1
      000552                            CMDDEF  NOSPF*UN,PFLRTN,P0$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,P0$SYS,0,+1
      000622                            CMDDEF  NOSYSP*RV,PFLRTN,P0$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
```

```
   000716                                       CMDDEF  NOGRO*UP,PFLRTN,P2$GRP,2,-1
   000730                                       CMDDEF  SAM*E,PFLRTN,P2$SAM,2,+1
   000742                                       CMDDEF  NOSAM*E,PFLRTN,P2$SAM,2,-1
   000754                                       CMDDEF  SUB*PROCESS,PFLRTN,P2$VIR,2,+1
   000766                                       CMDDEF  NOSUB*PROCESS,PFLRTN,P2$VIR,2,-1
   001000                                       CMDDEF  VIR*TUAL,PFLRTN,P2$VIR,2,+2
   001012                                       CMDDEF  NOVIR*TUAL,PFLRTN,P2$VIR,2,-2
   001024                                       CMDDEF  UP1,PFLRTN,P2$UP1,2,+1
   001036                                       CMDDEF  NOUP1,PFLRTN,P2$UP1,2,-1
   001050                                       CMDDEF  UP2,PFLRTN,P2$UP2,2,+1
   001062                                       CMDDEF  NOUP2,PFLRTN,P2$UP2,2,-1
   001074                                       CMDDEF  UP3,PFLRTN,P2$UP3,2,+1
   001106                                       CMDDEF  NOUP3,PFLRTN,P2$UP3,2,-1
   001120                                       CMDDEF  UP4,PFLRTN,P2$UP4,2,+1
   001132                                       CMDDEF  NOUP4,PFLRTN,P2$UP4,2,-1
   001144                                       CMDDEF  NONE,PFLNON,0,0,-2
   001156                                       CMDDEF  STA*NDARD,PFLSTD,0,0,+2
   001170                                       CMDDEF  STD,PFLSTD,0,0,+2
   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     PFSO:   .WORD   0,0,0,0
 000072  000000
 000074  000000  000000  000000     PFCO:   .WORD   0,0,0,0
 000102  000000
                                    ;
                                    ;   Misc data
                                    ;
 000104  100076                     R50TTO: .RAD50  /TTO/
 000106  100105                     R50TT7: .RAD50  /TT7/
 000110  012240                     R50CL:  .RAD50  /CL/
 000112  012276                     R50CLO: .RAD50  /CLO/
 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/
 000132  075273
```

```
     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  014566                       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$:     MOV     #PRVHD,R4       ;Point to table of privilege keywords
000204  004767  000476                       CALL    SETWRD          ;Process the next privilege keyword
                                     ;
                                     ;  See if we have reached the end of the list
                                     ;
000210  004767  014536                       CALL    SKPSPC          ;Skip over spaces
000214  005705                               TST     R5              ;Is this a multi-item list?
000216  001413                               BEQ     9$              ;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
```

```
      1                                                .SBTTL  CLRPRV -- Clear all parsing privilege flags
      2                                        ;-----------------------------------------------------------------------
      3                                        ;  Clear the words used to hold the privilege flags gotten during parsing.
      4                                        ;
      5                                        ;  Outputs:
      6                                        ;   PFS0...PFSn and PFC0..PFCn are set to zero.
      7                                        ;
      8 000254  010446                         CLRPRV: MOV     R4,-(SP)
      9 000256  010546                                 MOV     R5,-(SP)
     10 000260  012704  000064'                        MOV     #PFS0,R4        ;Words of bits to set
     11 000264  012705  000074'                        MOV     #PFC0,R5        ;Words of bits to clear
     12 000270  012700  000000G                        MOV     #PVNPW,R0       ;Get # words to clear
     13 000274  005024                         2$:     CLR     (R4)+
     14 000276  005025                                 CLR     (R5)+
     15 000300  077003                                 SOB     R0,2$
     16 000302  012605                                 MOV     (SP)+,R5
     17 000304  012604                                 MOV     (SP)+,R4
     18 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     #PRIVSO,R4      ;Point to set privileges
000412  012705   000000G                 MOV     #PRIVFO,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$:     MOV     #PRIVFO,R4      ;Point to cells with command file privileges
000436  012705   000000G                 MOV     #PRIVCO,R5      ;Point to current priv cells
000442  012700   000000G                 MOV     #PVNPW,R0       ;Get # words to move
000446  012425                   1$:     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  014122         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  014076          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  014032                 CALL    SKPSPC          ;Skip over leading spaces
 000720  010302                         MOV     R3,R2           ;Save keyword pointer
 000722  004767  004356                 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  014006                 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$:     MOV     R2,R3           ;Get back pointer to keyword
                                ;
                                ;  Look up the option keyword
                                ;
 000752  004767  007352                 CALL    SEARCH          ;Look up keyword in table
 000756  103405                         BCS     10$             ;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  014072                 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 PFSO or PFCO vector
                                 ;
001032  012705  000064'                  MOV     #PFSO,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     #PFCO,R5        ;Point to clear-flag words
001050  066405  000002           1$:     ADD     2(R4),R5        ;Point to correct privilege word
001054  051415                           BIS     (R4),(R5)       ;Set correct flag bit
                                 ;
                                 ;  Now clear the bit in the complementary vector
                                 ;
001056  012705  000074'                  MOV     #PFCO,R5        ;Assume we are granting privilege
001062  005764  000004                   TST     4(R4)           ;Are we setting or clearing privilege?
001066  002002                           BGE     2$              ;Br if granting privilege
001070  012705  000064'                  MOV     #PFSO,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$$NP,PFSO    ;Grant normal privileges
001114  012767  000000G 176744           MOV     #P2$$NP,PFSO+2
001122  012767  000000C 176744           MOV     #^C<P0$$NP>,PFCO;Remove all other privileges
001130  012767  000000C 176740           MOV     #^C<P2$$NP>,PFCO+2
001136  000207                           RETURN

                                 ;-----------------------------------------------------------------------
                                 ;  Set all privilege flags.
                                 ;
001140  010246                   PFLALL: MOV     R2,-(SP)
001142  012702  000064'                  MOV     #PFSO,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  0000009                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
```

```
001326  016704  000062              MOV     INDCOL,R4       ;Get column to indent to
001332  010401                      MOV     R4,R1           ;Reset column counter
001334  005304                      DEC     R4              ;Get # spaces to print
001336                      7$:     .TTYOUT #BLANK          ;Print a space
001346  077405                      SOB     R4,7$           ;Indent to desired column
001350  011504              6$:     MOV     (R5),R4         ;Get pointer to keyword string
001352  112400              4$:     MOVB    (R4)+,R0        ;Get next character of keyword
001354  001407                      BEQ     2$              ;Br if finished
001356  120027  000052              CMPB    R0,#'*          ;Don't print "*"
001362  001773                      BEQ     4$
001364                              .TTYOUT                 ;Print a character
001370  005201                      INC     R1              ;Count another column
001372  000767                      BR      4$              ;Go print rest
                            ;
                            ;  Check next entry
                            ;
001374  062705  000012      2$:     ADD     #10.,R5         ;Point to next entry
001400  005715                      TST     (R5)            ;Is there another entry?
001402  001323                      BNE     1$              ;Loop if yes
                            ;
                            ;  Finished
                            ;
001404  010104                      MOV     R1,R4           ;Return updated column # in R4
001406  012605              9$:     MOV     (SP)+,R5
001410  012601                      MOV     (SP)+,R1
001412  000207                      RETURN
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
                                ;
```

```
001544                      6$:     FERR    #EM$CAJ         ;Cannot access that job
001560  000261                      SEC                     ;Signal error on return
001562  000401                      BR      9$
                            ;
                            ;  We can access the job
                            ;
001564  000241              7$:     CLC                     ;Signal success on return
                            ;
                            ;  Finished
                            ;
001566  012601              9$:     MOV     (SP)+,R1
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  0000000                 MOV     #SDCB,R0        ;POINT TO CONTROL BLOCK FOR 1ST SPOOLED DEV
 001600  020027  0000000         4$:     CMP     R0,#SDCBND      ;CHECKED ALL?
 001604  103010                          BHIS    2$              ;BR IF YES
 001606  005760  0000000                 TST     SDFHD(R0)       ;ANY PENDING PRINT FILES?
 001612  001003                          BNE     3$              ;BR IF YES
 001614  062700  0000000                 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  000000G      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 is lookup
 001714  005067  000000G                          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$:     MOV     SYINDX,R2          ;GET DEVICE TABLE INDEX # FOR SY DEVICE
 001742  016202  000000G                          MOV     PNAME(R2),R2       ;GET RAD50 NAME OF SYSTEM DEVICE
 001746  010267  176154                           MOV     R2,BOTHAN+2        ;Put name in lookup
 001752  116703  000001G                          MOVB    SYUNIT+1,R3        ;GET UNIT # OF SYSTEM DISK
 001756  060302                                   ADD     R3,R2              ;ADD UNIT NUMBER TO BOOT DEVICE NAME
 001760  062702  000036                           ADD     #36,R2             ;CONVERT TO RAD50 UNIT NUMBER
 001764  010267  000000G                          MOV     R2,BOTDEV          ;BOOT FROM THIS DEVICE
 001770  010367  000000G                  3$:     MOV     R3,BOTUNI          ;SAVE BOOT DEVICE UNIT #
                                          ;
                                          ;  Find the device's CSR address.
                                          ;
 001774  016703  000000G                  4$:     MOV     SYINDX,R3          ;Get device table index # for SY device
 002000  016367  000000G 176116                   MOV     PNAME(R3),BOTHAN;Get RAD50 name of system device
 002006                                           .SERR                      ;Don't abort on lookup errors
 002014                                           .LOOKUP #XAREA,#1,#BOTHAN ;Try to open handler file
 002034  103437                                   BCS     99$                ;Error on lookup
                                          ;
                                          ;  Read block 0 of handler and save information about CSR address
                                          ;
 002036                                           .READW  #XAREA,#1,#BLK0,#256.,#0 ;Read block 0 of handler
 002074  103417                                   BCS     99$                ;Error on read
 002076  016767  000000C 000000G                  MOV     BLK0+H.CSR,BOTCSR          ;Save CSR info
 002104                                           .CLOSE  #1                 ;Close channel
                                          ;
                                          ;  Read the primary and secondary bootstrap.
                                          ;
 002112                                           .LOOKUP #XAREA,#1,#BOTDEV;DO NON-FILE-STRUCTURED LOOKUP ON BOOT DEVICE
 002132  103004                                   BCC     1$                 ;BR IF LOOKUP SUCCESSFUL
 002134                                   99$:    FABORT  #BADBOT            ;ERROR ON BOOT LOOKUP
                                          ;  Read primary driver into memory.
 002144                                   1$:     .READW  #XAREA,#1,#START,#256.,#0;Read primary bootstrap
 002202                                           .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)
                                          ;
```

```
     58 002242  012700  0000000        5$:     MOV     #BOTEMT,R0
     59 002246  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.
                                        ;
002250  010146                          PUSHCF: MOV     R1,-(SP)
002252  116701  000000G                         MOVB    CORUSR,R1          ;GET USER INDEX #
002256  032761  000000G 000000G                 BIT     #$CFOPN,LSW4(R1);IS A COMMAND FILE OPEN NOW?
002264  001030                                  BNE     6$                 ;BR IF YES
                                        ;  THERE IS NO COMMAND FILE OPEN NOW
002266  132767  000000G 000000G                 BITB    #IN$ACT,INDSTA     ;IS IND ACTIVE NOW?
002274  001403                                  BEQ     11$                ;BR IF NOT
002276  116767  000000G 000000G                 MOVB    INDSTA,CFIND       ;IF IND IS ACTIVE, SAVE ITS STATUS
002304  005767  000000G                 11$:    TST     CFPNT              ;IS A COMMAND FILE IN USE NOW?
002310  001003                                  BNE     8$                 ;BR IF YES
002312  116767  000000G 000000G                 MOVB    INDSTA,CFIND       ;SAVE IND STATUS FLAGS
002320  042761  000000G 000000G 8$:     BIC     #CFLFL4,LSW4(R1);CLEAR MISC COMMAND FILE FLAGS
002326  032761  000000G 000000G                 BIT     #$QTSET,LSW2(R1);DOES HE WANT QUIET OR NOQUIET?
002334  001524                                  BEQ     9$                 ;BR IF NOQUIET WANTED
002336  052761  000000G 000000G                 BIS     #$QUIET,LSW4(R1);SET QUIET
002344  000520                                  BR      9$
                                        ;  THERE IS AN OPEN COMMAND FILE WHICH NEEDS TO BE PUSHED
002346  010346                          6$:     MOV     R3,-(SP)
002350  010446                                  MOV     R4,-(SP)
002352  010546                                  MOV     R5,-(SP)
002354  016705  000000G                         MOV     CFSP,R5            ;GET SAVE STACK POINTER
002360  020527  000000C                         CMP     R5,#<CFSEND+20.+<2*PVNPW>>;ROOM ENOUGH TO START PUSH?
002364  103535                                  BLO     CFOVFL             ;BR IF STACK OVERFLOW WOULD OCCUR
                                        ;  DO .SAVESTATUS TO SAVE FILE NAME
002366  162705  000012                          SUB     #10.,R5            ;NEED 5 WORDS FOR .SAVEST
002372                                          .SAVEST #XAREA,#CFCHAN,R5
                                        ;  NOW SAVE BUFFER POINTERS
002410  016745  000000G                         MOV     CFBLK,-(R5)        ;CURRENT FILE BLOCK #
002414  016745  000000G                         MOV     CFPNT,-(R5)        ;CURRENT POINTER INTO BUFFER
                                        ;  Save command file privileges
002420  012703  000000C                         MOV     #PRIVFO+<2*PVNPW>,R3 ;Point past last privilege word
002424  014345                          12$:    MOV     -(R3),-(R5)        ;Push each privilege word
002426  020327  000000G                         CMP     R3,#PRIVFO         ;Pushed all yet?
002432  101374                                  BHI     12$                ;Br if more to push
                                        ;  Save command file attribute flags
002434  016745  000000G                         MOV     AFCF,-(R5)         ;Push attribute flags
                                        ;  SAVE QUIET STATUS
002440  016145  000000G                         MOV     LSW4(R1),-(R5)     ;SAVE QUIET FLAG
                                        ;  Save IND status
002444  016745  000000G                         MOV     CFIND,-(R5)        ;PUSH IND STATUS FLAGS
002450  116767  000000G 000000G                 MOVB    INDSTA,CFIND       ;SET NEW IND STATUS FOR COMMAND FILE
                                        ;  NOW SAVE INFO ABOUT PARAMETERS
002456  016745  000000G                         MOV     CURPRM,-(R5)       ;CURRENT PARAMETER POINTER
002462  016704  000000G                         MOV     PBFEND,R4          ;ADDR OF END OF PARAMETER STRING
002466  005204                                  INC     R4                 ;ROUND UP TO NEXT WORD
002470  042704  000001                          BIC     #1,R4
002474  012703  000000G                         MOV     #PRMBUF,R3         ;POINT TO START OF PARAM STRING
002500  020304                          2$:     CMP     R3,R4              ;PUSHED ALL ON STACK?
002502  103005                                  BHIS    1$                 ;BR IF YES
002504  020527  000002G                         CMP     R5,#<CFSEND+2.>    ;STACK OVERFLOW?
002510  101463                                  BLOS    CFOVFL             ;BR IF OVERFLOW
```

```
002512  012345                          MOV     (R3)+,-(R5)     ;PUSH PARAMETER STRING
002514  000771                          BR      2$
002516  010445                  1$:     MOV     R4,-(R5)        ;PUSH POINTER TO END OF STRING
                                ;  PUSH PARAMETER POINTERS
002520  005045                          CLR     -(R5)           ;PUSH ZERO TO MARK END
002522  012703  000000G                 MOV     #LSTPRM,R3      ;POINT TO LAST PARAM PTR CELL
002526  005743                  4$:     TST     -(R3)           ;IS POINTER IN USE?
002530  001004                          BNE     3$              ;BR IF YES
002532  020327  000000G                 CMP     R3,#PRMPNT      ;CHECKED ALL?
002536  101373                          BHI     4$              ;BR IF NOT
002540  000410                          BR      5$              ;BR IF NO PARAMETERS DEFINED
002542  020527  000000G         3$:     CMP     R5,#CFSEND      ;ROOM TO PUSH INFO?
002546  101444                          BLOS    CFOVFL          ;BR IF STACK OVERFLOW
002550  011345                          MOV     (R3),-(R5)      ;PUSH PARAMETER POINTER
002552  005743                          TST     -(R3)           ;POINT TO NEXT PARAM POINTER CELL
002554  020327  000000G                 CMP     R3,#PRMPNT      ;MORE TO PUSH?
002560  103370                          BHIS    3$              ;BR IF YES
                                ;  WE HAVE SAVED ALL INFORMATION NEEDED TO RESTART INDIRECT FILE
002562  010567  000000G         5$:     MOV     R5,CFSP         ;SAVE STACK POINTER
002566  042761  000000G 000000G         BIC     #$CFOPN,LSW4(R1);SAY NO FILE IS OPEN NOW
002574  105267  000000G                 INCB    CFNEST          ;SAY WE ARE NESTED DEEPTER
002600  012605                          MOV     (SP)+,R5
002602  012604                          MOV     (SP)+,R4
002604  012603                          MOV     (SP)+,R3
                                ;  SET UP POINTERS FOR NEW COMMAND FILE
002606  012700  000000G         9$:     MOV     #CFBUF,R0       ;SET POINTER TO COMMAND BUFFER
002612  004767  001002                  CALL    CFSTRT          ;INIT POINTER INTO BUFFER
002616  005067  000000G                 CLR     CFBLK           ;RESET FILE BLOCK # TO ZERO
002622  005067  000000G                 CLR     CURPRM          ;CLEAR PARAM STRING POINTER
002626  012701  000000G                 MOV     #PRMPNT,R1      ;CLEAR ALL PARAM POINTERS
002632  005021                  7$:     CLR     (R1)+
002634  020127  000000G                 CMP     R1,#LSTPRM
002640  103774                          BLO     7$
002642  012767  000000G 000000G         MOV     #PRMBUF,PBFEND  ;SAY NO PARAM STRING YET
                                ;  Say that IND is not active now
002650  105067  000000G                 CLRB    INDSTA          ;SAY IND IS NOT ACTIVE NOW
002654  012601                          MOV     (SP)+,R1
002656  000207                          RETURN
                                ;
                                ;  ERROR -- OVERFLOW OF FILE SAVE STACK
002660                          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.
                                ;
002670  010146                  POPCF:  MOV     R1,-(SP)
002672  116701  0000000                 MOVB    CORUSR,R1          ;GET USER INDEX NUMBER
002676  042761  0000000 0000000         BIC     #$CFCCL,LSW4(R1);SAY WE HAVE FINISHED ANY CCL COMMAND
002704  105767  0000000                 TSTB    CFNEST             ;Any nested command files?
002710  001003                          BNE     10$                ;Br if yes
002712  005767  0000000                 TST     CFPNT              ;ANY INDIRECT FILES IN USE?
002716  001563                          BEQ     9$                 ;BR IF NOT
002720  116767  0000000 0000000 10$:    MOVB    CFIND,INDSTA       ;RESTORE IND STATUS FLAGS
002726  105067  0000000                 CLRB    CFHOLD             ;CLEAR ANY COMMAND FILE HOLDING CHAR
002732  032761  0000000 0000000         BIT     #$CFOPN,LSW4(R1);IS @FILE CHANNEL OPEN NOW?
002740  001403                          BEQ     1$                 ;BR IF NOT
                                ;  CLOSE CURRENTLY OPEN FILE
002742                                  .CLOSE  #CFCHAN            ;CLOSE THE FILE
                                ;  SEE IF THERE IS A HIGHER LEVEL FILE TO RESTORE
002750  105767  0000000         1$:     TSTB    CFNEST             ;ANY FILES ON STACK NOW?
002754  001016                          BNE     2$                 ;BR IF YES
                                ;  THERE ARE NO HIGHER LEVEL FILES
002756  042761  000000C 0000000         BIC     #<$CFOPN!$CFALL!$CFSOT>,LSW4(R1);CLEAR COMMAND FILE FLAGS
002764  042761  0000000 0000000         BIC     #$NOIN,LSW3(R1) ;Allow input to be accepted for line
002772  042761  0000000 0000000         BIC     #$SUCF,LSW9(R1) ;Say start-up command file finished
003000  004767  000570                  CALL    CFSTOP             ;Suspend command file input
003004  004767  175336                  CALL    RSTPRV             ;Reset command file privileges
003010  000526                          BR      9$
                                ;  REOPEN NEXT HIGHER LEVEL FILE
                                ;  RESTORE PARAMETER POINTERS
003012  010346                  2$:     MOV     R3,-(SP)
003014  010446                          MOV     R4,-(SP)
003016  010546                          MOV     R5,-(SP)
003020  016705  0000000                 MOV     CFSP,R5            ;GET STACK POINTER
003024  012703  0000000                 MOV     #PRMPNT,R3         ;POINT TO PARAM POINTER CELLS
003030  012504                  4$:     MOV     (R5)+,R4           ;GET A PARAMETER POINTER
003032  001402                          BEQ     3$                 ;BR IF END OF LIST HIT
003034  010423                          MOV     R4,(R3)+           ;RESTORE POINTER
003036  000774                          BR      4$
003040  020327  0000000         3$:     CMP     R3,#LSTPRM         ;ZERO ALL OTHER PARAM POINTERS
003044  103002                          BHIS    5$
003046  005023                          CLR     (R3)+
003050  000773                          BR      3$
                                ;  RESTORE PARAMETER STRING
003052  012504                  5$:     MOV     (R5)+,R4           ;GET ADDRESS OF END OF STRING
003054  010467  0000000                 MOV     R4,PBFEND
003060  020427  0000000         7$:     CMP     R4,#PRMBUF         ;RESTORED ALL OF PARAM STRING?
003064  101402                          BLOS    6$                 ;BR IF YES
003066  012544                          MOV     (R5)+,-(R4)        ;POP STRING OFF STACK
003070  000773                          BR      7$
003072  012567  0000000         6$:     MOV     (R5)+,CURPRM       ;POP POINTER INTO STRING
                                ;  Restore IND status flags
003076  012567  0000000                 MOV     (R5)+,CFIND        ;RESTORE IND STATUS FLAGS
                                ;  RESET COMMAND FILE CONTROL FLAGS
003102  012704  0000000                 MOV     #CFLFL4,R4         ;GET MIST CONTROL FLAGS
003106  040461  0000000                 BIC     R4,LSW4(R1)        ;CLEAR THOSE FLAGS
```

```
003112  005104                          COM     R4                  ;MASK ALL BUT THOSE FLAGS
003114  040415                          BIC     R4,(R5)
003116  052561  0000000                 BIS     (R5)+,LSW4(R1)      ;SET DESIRED COMBINATION
                                ;  Restore command file attribute flags
003122  012567  0000000                 MOV     (R5)+,AFCF          ;Restore attribute flags
                                ;  Restore command file privilege flags
003126  012700  0000000                 MOV     #PRIVFO,RO          ;Point to privilege vector
003132  012520                  11$:    MOV     (R5)+,(RO)+         ;Pop a privilege flag
003134  020027  000000C                 CMP     RO,#PRIVFO+<2*PVNPW>;Popped all?
003140  103774                          BLO     11$                 ;Loop if not
003142  004767  175200                  CALL    RSTPRV              ;Reset some flags
                                ;  RESTORE BUFFER POINTER INFORMATION
003146  012500                  8$:     MOV     (R5)+,RO            ;POINTER INTO BUFFER
003150  004767  000444                  CALL    CFSTRT              ;Set command file buffer pointer
003154  012567  0000000                 MOV     (R5)+,CFBLK         ;CURRENT FILE BLOCK NUMBER
                                ;  NOW REOPEN THE INDIRECT COMMAND FILE
003160                                  .REOPEN #XAREA,#CFCHAN,R5
003176  062705  000012                  ADD     #10.,R5             ;POP FILE NAME INFO OFF STACK
                                ;  REREAD CURRENT BUFFER
003202                                  .READW  #XAREA,#CFCHAN,#CFBUF,#256.,CFBLK
                                ;  FINISHED RESTORING FILE
003242  010567  0000000                 MOV     R5,CFSP             ;SAVE UPDATED STACK POINTER
003246  105367  0000000                 DECB    CFNEST              ;SAY ONE LESS FILE ON STACK
003252  052761  0000000 0000000         BIS     #$CFOPN,LSW4(R1);SAY CHANNEL IS OPEN
003260  012605                          MOV     (SP)+,R5
003262  012604                          MOV     (SP)+,R4
003264  012603                          MOV     (SP)+,R3
003266  012601                  9$:     MOV     (SP)+,R1
003270  000207                          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.
                                         ;
 003272  010146                          ABRTCF: MOV     R1,-(SP)
 003274  116701  000000G                         MOVB    CORUSR,R1         ;GET USER INDEX #
 003300  016700  000000G                         MOV     CFSPND,R0         ;Is there a suspended command file?
 003304  001404                                  BEQ     2$                ;Br if not
 003306  004767  000306                          CALL    CFSTRT            ;Restart command file input
 003312  005067  000000G                         CLR     CFSPND            ;No more suspended command file
                                         ;
                                         ;  Abort all nested command files
                                         ;
 003316  105767  000000G                 2$:     TSTB    CFNEST            ;Any nested command files?
 003322  001003                                  BNE     3$                ;Br if yes
 003324  005767  000000G                         TST     CFPNT             ;IS AN INDIRECT FILE OPEN?
 003330  001407                                  BEQ     4$                ;BR IF NOT
 003332  132767  000000G 000000G         3$:     BITB    #IN$ACT,INDSTA    ;Have we reached the level of IND?
 003340  001003                                  BNE     4$                ;Br if yes -- Leave IND in control
 003342  004767  177322                          CALL    POPCF             ;CLOSE IT
 003346  000763                                  BR      2$
                                         ;
                                         ;  Reset misc command file related values
                                         ;
 003350  042761  000000G 000000G         4$:     BIC     #$CFABT,LSW6(R1);SAY ALL COMMAND FILES HAVE BEEN ABORTED
 003356  042761  000000G 000000G                 BIC     #$NTGCC,LSW9(R1);Clear saved ctrl-C flag
 003364  042761  000000G 000000G                 BIC     #$CFDCC,LSW4(R1)
 003372  105067  000000G                         CLRB    UERSEV            ;CLEAR USER ERROR FLAG
                                         ;
                                         ;  Reset command file privileges from set privileges
                                         ;
 003376  004767  174744                          CALL    RSTPRV            ;Reset current privileges
                                         ;
                                         ;  Finished
                                         ;
 003402  012601                                  MOV     (SP)+,R1
 003404  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.
                                 ;
 003406  010146                  INDABT: MOV     R1,-(SP)
 003410  116701  000000G                 MOVB    CORUSR,R1       ;Get user index #
                                 ;
                                 ;  Close command files
                                 ;
 003414  016700  000000G                 MOV     CFSPND,R0       ;Is there a suspended command file?
 003420  001404                          BEQ     2$              ;Br if not
 003422  004767  000172                  CALL    CFSTRT          ;Restart suspended command file
 003426  005067  000000G                 CLR     CFSPND          ;No more suspended command file
 003432  005767  000000G         2$:     TST     CFPNT           ;Is an indirect file open?
 003436  001403                          BEQ     1$              ;Br if not
 003440  004767  177224                  CALL    POPCF           ;Close it
 003444  000772                          BR      2$
 003446  042761  000000C 000000G 1$:     BIC     #$CFKIL!$CFABT,LSW6(R1);Say command files have been aborted
 003454  105067  000000G                 CLRB    UERSEV          ;Clear user error flag
                                 ;
                                 ;  Stop IND
                                 ;
 003460  105067  000000G                 CLRB    INDSTA          ;Say IND is finished
 003464  042761  000000G 000000G         BIC     #$INDRN,LSW5(R1);Say IND is not running now
 003472  042761  000000G 000000G         BIC     #$NTGCC,LSW9(R1);Clear saved ctrl-C flag
 003500  042761  000000G 000000G         BIC     #$CFDCC,LSW4(R1)
                                 ;
                                 ;  Reset privileges
                                 ;
 003506  004767  174634                  CALL    RSTPRV          ;Reset privileges
                                 ;
                                 ;  Eliminate IND global PLAS region
                                 ;
 003512  012767  000000C 000000C         MOV     #<RS.GBL!RS.PVT>,INDRDB+R.GSTS ;Set status flags in RDB
 003520                                  .CRRG   #XAREA,#INDRDB  ;Try to attach to IND region
 003540  103413                          BCS     3$              ;Br if cannot attach
 003542  012767  000000G 000000C         MOV     #RS.EGR,INDRDB+R.GSTS ;Set eliminate-global-region flag
 003550                                  .ELRG   #XAREA,#INDRDB  ;Eliminate the region
                                 ;
                                 ;  Finished
                                 ;
 003570  012601                  3$:     MOV     (SP)+,R1
 003572  000207                          RETURN
```

```
                                           .SBTTL  CFSTOP -- Stop input from a command file
                                   ;------------------------------------------------------------------
                                   ;  Zero CFPNT to say input is not coming from a command file.
                                   ;
    003574  010046                 CFSTOP: MOV     RO,-(SP)
                                   ;
                                   ;  Say input not coming from a command file
                                   ;
    003576  005067  000000G                CLR     CFPNT           ;Clear command file buffer pointer
                                   ;
                                   ;  Reset status flags in RMON cell
                                   ;
    003602  016700  000000G                MOV     CXTRMN,RO       ;Get virtual address of RMON
--> 003606  042760  000000G 000000G        BIC     #CFACFL,R$CFST(RO) ;Say input not from CF
                                   ;
                                   ;  Finished
                                   ;
    003614  012600                         MOV     (SP)+,RO
    003616  000207                         RETURN

                                           .SBTTL  CFSTRT -- Start input from a command file
                                   ;------------------------------------------------------------------
                                   ;  Store a buffer pointer into CFPNT to start command file input.
                                   ;
                                   ;  Inputs:
                                   ;   RO = Value to be stored into CFPNT
                                   ;
    003620  010046                 CFSTRT: MOV     RO,-(SP)
                                   ;
                                   ;  Store buffer pointer into CFPNT
                                   ;
    003622  010067  000000G                MOV     RO,CFPNT        ;Set CF buffer pointer
    003626  001405                         BEQ     9$              ;Br if not starting command file
                                   ;
                                   ;  Set status flags in RMON cell
                                   ;
    003630  016700  000000G                MOV     CXTRMN,RO       ;Get RMON address
--> 003634  052760  000000G 000000G        BIS     #CFACFL,R$CFST(RO);Set command-file-active flags
                    110400     370 ;
                                   ;  Finished
                                   ;
    003642  012600                 9$:     MOV     (SP)+,RO
    003644  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.
                                ;
003646  010246                  CFSQEZ: MOV     R2,-(SP)
003650  010346                          MOV     R3,-(SP)
003652  010446                          MOV     R4,-(SP)
                                ;
                                ;  Determine if there are any pending commands in the command file buffer
                                ;
003654  012703  001000G                 MOV     #CFBUF+512.,R3  ;POINT PAST TOP OF BUFFER
003660  016704  000000G                 MOV     CFPNT,R4        ;GET POINTER TO PENDING COMMANDS
003664  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.
                                ;
003666  010302                          MOV     R3,R2           ;GET POINTER TO TOP OF BUFFER
003670  020204                  1$:     CMP     R2,R4           ;HAVE WE MOVED ALL PENDING COMMANDS?
003672  101404                          BLOS    6$              ;BR IF YES
003674  114200                          MOVB    -(R2),R0        ;GET NEXT CHAR
003676  001774                          BEQ     1$              ;AND SKIP NULLS
003700  110043                          MOVB    R0,-(R3)        ;PACK INTO TOP OF BUFFER
003702  000772                          BR      1$
003704  010367  000000G         6$:     MOV     R3,CFPNT        ;SAVE NEW COMMAND FILE POINTER
                                ;
                                ;  Null fill buffer from base up to start of pending commands
                                ;
003710  010300                  4$:     MOV     R3,R0           ;SAVE POINTER PAST TOP OF FREE AREA
003712  012704  000000G                 MOV     #CFBUF,R4       ;POINT TO BASE OF BUFFER
003716  020304                  3$:     CMP     R3,R4           ;HAVE WE FILLED TO BASE?
003720  101402                          BLOS    5$              ;BR IF YES
003722  105043                          CLRB    -(R3)           ;NULL FILL BUFFER
003724  000774                          BR      3$
                                ;
                                ;  Finished
                                ;
003726  012604                  5$:     MOV     (SP)+,R4
003730  012603                          MOV     (SP)+,R3
003732  012602                          MOV     (SP)+,R2
003734  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.
                                           ;
 003736  010246                            LOGCHK: MOV     R2,-(SP)
 003740  010346                                    MOV     R3,-(SP)
 003742  010446                                    MOV     R4,-(SP)
 003744  010546                                    MOV     R5,-(SP)
                                           ;
                                           ;  See if the log file is currently open
                                           ;
 003746  032767  0000000 0000000G                 BIT     #LF$OPN,LOGFLG  ;Is the log file currently open?
 003754  001445                                    BEQ     9$              ;Br if not
                                           ;
                                           ;  The log file is open.
                                           ;  Convert device name into device # and unit #
                                           ;
 003756  004767  005634                            CALL    CHKDEV          ;Cvt name to dev # and unit #
 003762  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.
                                           ;
 003764  020467  0000000                           CMP     R4,LDDEVX       ;Is this a logical disk?
 003770  001406                                    BEQ     1$              ;Br if yes
                                           ;
                                           ;  This is a physical disk
                                           ;
 003772  000300                                    SWAB    R0              ;Get unit # to high-order byte
 003774  050004                                    BIS     R0,R4           ;Combine device and unit #
 003776  020467  0000000                           CMP     R4,LOGDVU       ;Same device as log file?
 004002  001032                                    BNE     9$              ;Br if not
 004004  000421                                    BR      8$              ;Br if yes
                                           ;
                                           ;  This is a logical disk
                                           ;
 004006  006300                            1$:     ASL     R0              ;Convert unit # to word table index
 004010  016004  0000000                           MOV     LDPDEV(R0),R4   ;Get physical dev # and unit #
 004014  016002  0000000                           MOV     LDBASE(R0),R2   ;Get base block # of logical disk
 004020  010203                                    MOV     R2,R3
 004022  066003  0000000                           ADD     LDSIZE(R0),R3   ;Get top block # of logical disk
 004026  020467  0000000                           CMP     R4,LOGDVU       ;Is log file on same physical disk?
 004032  001016                                    BNE     9$              ;Br if not
 004034  026702  0000000                           CMP     LOGBAS,R2       ;Is log file on log disk within this disk?
 004040  103413                                    BLO     9$              ;Br if not
 004042  026703  0000000                           CMP     LOGBAS,R3
 004046  103010                                    BHIS    9$
                                           ;
                                           ;  Log file is on the specified device
                                           ;  Pring a warning message and close the log file.
                                           ;
 004050                          8$:     FWARN   #TM$CLG         ;Say we are closing the log file
 004064  004767  000012                  CALL    LOGCLS          ;Close the log file
                                 ;
                                 ;  Finished
                                 ;
 004070  012605                  9$:     MOV     (SP)+,R5
 004072  012604                          MOV     (SP)+,R4
 004074  012603                          MOV     (SP)+,R3
 004076  012602                          MOV     (SP)+,R2
 004100  000207                          RETURN
```

```
                                        .SBTTL  LOGCLS -- Close the log file
                                ;------------------------------------------------------------------------
                                ;  LOGCLS is called to close the log file for the current job.
                                ;
 004102  010246                 LOGCLS: MOV     R2,-(SP)
 004104  032767  000000G 000000G        BIT     #LF$OPN,LOGFLG  ;Is the log file open?
 004112  001451                         BEQ     9$              ;Br if not
                                ;
                                ;  Log file is open, write last buffer to file
                                ;
 004114  016702  000000G                MOV     LOGPTR,R2       ;Get buffer pointer
 004120  001436                         BEQ     2$              ;Br if file overflow occurred
 004122  032702  000001                 BIT     #1,R2           ;Do we need to put in a null at end?
 004126  001401                         BEQ     1$              ;Br if not
 004130  105022                         CLRB    (R2)+           ;Put null to fill out last word
 004132  162702  000000G        1$:     SUB     #LOGBUF,R2      ;Get # bytes in log buffer
 004136  001427                         BEQ     2$              ;Br if buffer is empty
 004140  006202                         ASR     R2              ;Get # words in buffer
 004142                                 .WRITW  #XAREA,#LOGCHN,#LOGBUF,R2,LOGBLK ;Write block to log file
 004200  103006                         BCC     2$              ;Br if write ok
 004202                                 FERR    #LGOVER         ;Log file overflow
                                ;
                                ;  Close the log file
                                ;
 004216  005067  000000G        2$:     CLR     LOGPTR          ;Say we are no longer logging
 004222  042767  000000G 000000G        BIC     #LF$OPN,LOGFLG  ;Say log file is closed
 004230                                 .CLOSE  #LOGCHN         ;Close log file channel
                                ;
                                ;  Finished
                                ;
 004236  012602                 9$:     MOV     (SP)+,R2
 004240  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.
                              ;
 004242  010446               ACRDEC: MOV     R4,-(SP)
                              ;
                              ;  Skip leading spaces and an optional leading equal sign
                              ;
 004244  004767  010502               CALL    SKPSPC          ;Skip over leading spaces
 004250  121327  000075               CMPB    (R3),#'=        ;Is there an equal sign?
 004254  001003                       BNE     3$              ;Br if not
 004256  005203                       INC     R3              ;Skip past the equal sign
 004260  004767  010466               CALL    SKPSPC          ;Skip spaces following equal sign
                              ;
                              ;  See if number is preceded by a minus sign
                              ;
 004264  105067  173656       3$:     CLRB    NEGFLG          ;Assume number should not be negated
 004270  121327  000055               CMPB    (R3),#'-      . ;Is there a leading minus sign?
 004274  001005                       BNE     5$              ;Br if not
 004276  105267  173644               INCB    NEGFLG          ;Remember to negate value
 004302  005203                       INC     R3              ;Skip past minus sign
 004304  004767  010442               CALL    SKPSPC          ;Skip over spaces
                              ;
                              ;  Make sure first character of number is a decimal digit
                              ;
 004310  111300               5$:     MOVB    (R3),R0         ;Get first character of number
 004312  120027  000060               CMPB    R0,#'0          ;Is 1st character a digit?
 004316  103432                       BLO     4$              ;Br if not
 004320  120027  000071               CMPB    R0,#'9
 004324  101027                       BHI     4$              ;Br if not
                              ;
                              ;  Accrue the number
                              ;
 004326  005001                       CLR     R1              ;ACCRUE VALUE IN R1
 004330  112300               2$:     MOVB    (R3)+,R0        ;GET NEXT CHARACTER
 004332  120027  000056               CMPB    R0,#'.          ;Decimal pointer delimiter?
 004336  001414                       BEQ     6$              ;Br if yes
 004340  010004                       MOV     R0,R4
 004342  162704  000060               SUB     #'0,R4          ;CONVERT DIGIT TO BINARY VALUE
 004346  100407                       BMI     1$              ;BRANCH IF CHAR NOT A DIGIT
 004350  020427  000011               CMP     R4,#9.          ;IS IT A DIGIT?
 004354  003004                       BGT     1$              ;BRANCH IF NOT
                              ;
                              ;  Multiply previous value by 10 and add new digit.
                              ;
 004356  070127  000012               MUL     #10.,R1         ;MULTIPLY BY 10.
 004362  060401                       ADD     R4,R1           ;ADD IN NEW DIGIT VALUE
 004364  000761                       BR      2$
```

```
                                 ;
                                 ;  Hit delimiter.
                                 ;
   004366  005303                1$:     DEC     R3              ;POINT TO DELIMITER
   004370  105767  173552        6$:     TSTB    NEGFLG          ;Should we negate the value
   004374  001401                        BEQ     9$              ;Br if not
   004376  005401                        NEG     R1              ;Negate the value
   004400  012604                9$:     MOV     (SP)+,R4
   004402  000207                        RETURN
                                 ;
                                 ;  Error -- Expected number does not start with a digit
                                 ;
   004404                        4$:     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.
                                  ;
 004414  010446                   ACROCT: MOV     R4,-(SP)
                                  ;
                                  ;  Skip leading spaces and an optional leading equal sign
                                  ;
 004416  004767  010330                   CALL    SKPSPC          ;Skip over leading spaces
 004422  121327  000075                   CMPB    (R3),#'=        ;Is there an equal sign?
 004426  001003                           BNE     3$              ;Br if not
 004430  005203                           INC     R3              ;Skip past the equal sign
 004432  004767  010314                   CALL    SKPSPC          ;Skip spaces following equal sign
                                  ;
                                  ;  See if number is preceded by a minus sign
                                  ;
 004436  105067  173504           3$:     CLRB    NEGFLG          ;Assume number should not be negated
 004442  121327  000055                   CMPB    (R3),#'-        ;Is there a leading minus sign?
 004446  001005                           BNE     5$              ;Br if not
 004450  105267  173472                   INCB    NEGFLG          ;Remember to negate value
 004454  005203                           INC     R3              ;Skip past minus sign
 004456  004767  010270                   CALL    SKPSPC          ;Skip over spaces
                                  ;
                                  ;  Make sure first character of number is an octal digit
                                  ;
 004462  111300                   5$:     MOVB    (R3),R0         ;Get first character of number
 004464  120027  000060                   CMPB    R0,#'0          ;Is 1st character a digit?
 004470  103432                           BLO     4$              ;Br if not
 004472  120027  000067                   CMPB    R0,#'7
 004476  101027                           BHI     4$              ;Br if not
                                  ;
                                  ;  Accrue the number
                                  ;
 004500  005001                           CLR     R1              ;ACCRUE VALUE IN R1
 004502  112300                   2$:     MOVB    (R3)+,R0        ;GET NEXT CHAR
 004504  010004                           MOV     R0,R4
 004506  162704  000060                   SUB     #'0,R4          ;CONVERT ASCII CHAR TO VALUE
 004512  100407                           BMI     1$              ;BRANCH IF NOT DIGIT
 004514  020427  000007                   CMP     R4,#7           ;LEGAL OCTAL DIGIT?
 004520  003004                           BGT     1$              ;BRANCH IF NOT
 004522  072127  000003                   ASH     #3,R1           ;MULTIPLY PREVIOUS VALUE BY 8
 004526  060401                           ADD     R4,R1           ;ADD IN NEW DIGIT
 004530  000764                           BR      2$
                                  ;  HIT DELIMITER
 004532  020427  000011           1$:     CMP     R4,#9.          ;Is this a decimal digit?
 004536  101407                           BLOS    4$              ;Error -- He specified a decimal value
 004540  005303                           DEC     R3              ;MAKE R3 POINT TO DELIMITER
 004542  105767  173400                   TSTB    NEGFLG          ;Should we negate the value?
```

```
004546  001401                    BEQ     9$              ;Br if not
004550  005401                    NEG     R1              ;Negate the value
004552  012604           9$:      MOV     (SP)+,R4
004554  000207                    RETURN
                         ;
                         ;  Error -- Invalid octal value
                         ;
004556                   4$:      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.
                                ;
 004566  010146                 ACRSPD: MOV     R1,-(SP)
                                ;
                                ;  Accrue decimal speed value
                                ;
 004570  004767  177446                 CALL    ACRDEC          ;Accrue decimal speed value
                                ;
                                ;  If speed was 134.5, skip over ".5"
                                ;
 004574  020127  000206                 CMP     R1,#134.        ;Decimal value for speed = 134?
 004600  001006                         BNE     2$              ;Br if not
 004602  122327  000056                 CMPB    (R3)+,#'.       ;Skip period
 004606  001013                         BNE     8$
 004610  122327  000065                 CMPB    (R3)+,#'5       ;Skip 5
 004614  001010                         BNE     8$
                                ;
                                ;  Convert speed to speed code
                                ;
 004616  012705  000036         2$:     MOV     #2*15.,R5       ;Get index to highest speed value
 004622  020165  000000'        3$:     CMP     R1,SPDVAL(R5)   ;Search for specified speed
 004626  001407                         BEQ     4$              ;Br if found it
 004630  162705  000002                 SUB     #2,R5           ;Try next table entry
 004634  002372                         BGE     3$              ;Loop if more to check
                                ;
                                ;  Invalid speed value
                                ;
 004636                         8$:     FABORT  #EM$ISV         ;Invalid speed value
                                ;
                                ;  Finished
                                ;
 004646  006205                 4$:     ASR     R5              ;Get 1x speed code
 004650  012601                         MOV     (SP)+,R1
 004652  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.
                                ;
004654  010046                  OCTPRT: MOV     R0,-(SP)
004656  010246                          MOV     R2,-(SP)
004660  012700  000030                  MOV     #30,R0
004664  000261                          SEC                             ;SET STOPPER BIT
004666  006102                  1$:     ROL     R2                      ;MOVE 1ST BIT TO R0
004670  106100                          ROLB    R0
004672                                  .TTYOUT                         ;PRINT THE ASCII CHAR
004676  012700  000206                  MOV     #206,R0                 ;SHIFTED 60 + REPEAT BIT
004702  006302                  2$:     ASL     R2                      ;SHIFT OFF 1ST BIT
004704  001403                          BEQ     3$                      ;BRANCH IF FINISHED
004706  106100                          ROLB    R0                      ;MOVE BIT TO R0
004710  103774                          BCS     2$                      ;DO LOOP 2 TIMES
004712  000765                          BR      1$                      ;NOW GO GET 3RD BIT
004714  012602                  3$:     MOV     (SP)+,R2
004716  012600                          MOV     (SP)+,R0
004720  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.
                            ;
 004722  010146             OCTFIX: MOV     R1,-(SP)
 004724  010346                     MOV     R3,-(SP)
 004726  010501                     MOV     R5,R1           ;Get value to be printed
 004730  020327  000006             CMP     R3,#6.          ;Are we printing a full 6 digits?
 004734  002404                     BLT     3$              ;Br if not
 004736  005000                     CLR     R0              ;Shift 1st bit into R0
 004740  073027  000001             ASHC    #1,R0
 004744  000411                     BR      2$              ;Enter conversion loop
 004746  012700  000020     3$:     MOV     #16.,R0         ;Determine # bits to shift to left
 004752  160300                     SUB     R3,R0           ; justify the data value in R1
 004754  160300                     SUB     R3,R0
 004756  160300                     SUB     R3,R0
 004760  072100                     ASH     R0,R1           ;Left justify data value in R1
 004762  005000             1$:     CLR     R0              ;Shift an octal digit into R0
 004764  073027  000003             ASHC    #3,R0
 004770  062700  000060     2$:     ADD     #'0,R0          ;Convert to ASCII character
 004774                             .TTYOUT                 ;Print the digit
 005000  077310                     SOB     R3,1$           ;Loop if more digits to print
 005002  012603                     MOV     (SP)+,R3
 005004  012601                     MOV     (SP)+,R1
 005006  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 BLKO.
                                     ;   RO = Number of characters in string (not counting null at end)
                                     ;   R3 = Points past end of string
                                     ;
 005010  010146                      ACRTXT: MOV     R1,-(SP)
 005012  010546                              MOV     R5,-(SP)
                                     ;
                                     ;  Skip up to start of string
                                     ;
 005014  004767  007732                      CALL    SKPSPC          ;Skip over any spaces
 005020  121327  000075                      CMPB    (R3),#'=        ;Was equal sign specified before string?
 005024  001003                              BNE     1$              ;Br if not
 005026  005203                              INC     R3              ;Skip past equal sign
 005030  004767  007716                      CALL    SKPSPC
                                     ;
                                     ;  See what the string delimiter is
                                     ;
 005034  111305                      1$:     MOVB    (R3),R5         ;Get string delimiter
 005036  120527  000047                      CMPB    R5,#47          ;Single quote?
 005042  001431                              BEQ     6$              ;Br if yes
 005044  120527  000042                      CMPB    R5,#42          ;Double quote?
 005050  001426                              BEQ     6$              ;Br if yes
                                     ;
                                     ;  String is not quoted.
                                     ;  Begin loop to get characters from the string.
                                     ;
 005052  012701  000000G             2$:     MOV     #BLKO,R1        ;Point to buffer where we store result
                                     ;
                                     ;  Get next char from input string
                                     ;
 005056  112300                      4$:     MOVB    (R3)+,RO        ;Get next char from string
 005060  001414                              BEQ     8$              ;Br if reached end of string
                                     ;
                                     ;  See if this is a control character sequence of the form ^char
                                     ;
 005062  120027  000136                      CMPB    RO,#'^          ;Start of control char sequence?
 005066  001007                              BNE     3$              ;Br if not
 005070  112300                              MOVB    (R3)+,RO        ;Get next char from string
 005072  001422                              BEQ     10$             ;Br if delimiter missing
 005074  120027  000136                      CMPB    RO,#'^          ;Make "^^" = "^"
 005100  001402                              BEQ     3$
 005102  042700  177740                      BIC     #^C<37>,RO      ;Convert char to control character
                                     ;
                                     ;  Store character into result buffer
                                     ;
 005106  110021                      3$:     MOVB    RO,(R1)+        ;Store char into result buffer
 005110  000762                              BR      4$              ;Go get next char
                                     ;
```

```
                                        ;  Reached end of string
                                        ;
   005112  005303                       8$:     DEC     R3              ;Point back to null at end of input string
   005114  105011                               CLRB    (R1)            ;Store null at end of string
   005116  162701  000000G                      SUB     #BLKO,R1        ;Determine length of string
   005122  010100                               MOV     R1,R0           ;Return in R0
   005124  000402                               BR      9$
                                        ;
                                        ;  Accrue a quoted string
                                        ;
   005126  004767  000016               6$:     CALL    ACRSTR          ;Accrue quoted string
                                        ;
                                        ;  Finished
                                        ;
   005132  012605                       9$:     MOV     (SP)+,R5
   005134  012601                               MOV     (SP)+,R1
   005136  000207                               RETURN
                                        ;
                                        ;  Invalid string
                                        ;
   005140                               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 BLKO.
                                ;   R0 = Number of characters in string (not counting null at end)
                                ;   R3 = Points past end of string
                                ;
005150  010146                  ACRSTR: MOV     R1,-(SP)
005152  010546                          MOV     R5,-(SP)
                                ;
                                ;  Skip up to start of string
                                ;
005154  004767  007572                  CALL    SKPSPC          ;Skip over any spaces
005160  121327  000075                  CMPB    (R3),#'=        ;Was equal sign specified before string?
005164  001003                          BNE     1$              ;Br if not
005166  005203                          INC     R3              ;Skip past equal sign
005170  004767  007556                  CALL    SKPSPC
                                ;
                                ;  See what the string delimiter is
                                ;
005174  112305                  1$:     MOVB    (R3)+,R5        ;Get string delimiter
005176  120527  000047                  CMPB    R5,#47          ;Single quote?
005202  001403                          BEQ     2$              ;Br if yes
005204  120527  000042                  CMPB    R5,#42          ;Double quote?
005210  001031                          BNE     10$             ;Br if invalid delimiter
                                ;
                                ;  Begin loop to get characters from the string
                                ;
005212  012701  000000G         2$:     MOV     #BLKO,R1        ;Point to buffer where we store result
                                ;
                                ;  Get next char from input string
                                ;
005216  112300                  4$:     MOVB    (R3)+,R0        ;Get next char from string
005220  001425                          BEQ     10$             ;Br if missing delimiter
005222  120005                          CMPB    R0,R5           ;Is this the end delimiter?
005224  001414                          BEQ     9$              ;Br if yes
                                ;
                                ;  See if this is a control character sequence of the form ^char
                                ;
005226  120027  000136                  CMPB    R0,#'^          ;Start of control char sequence?
005232  001007                          BNE     3$              ;Br if not
005234  112300                          MOVB    (R3)+,R0        ;Get next char from string
005236  001416                          BEQ     10$             ;Br if delimiter missing
005240  120027  000136                  CMPB    R0,#'^          ;Make "^^" = "^"
005244  001402                          BEQ     3$
005246  042700  177740                  BIC     #^C<37>,R0      ;Convert char to control character
                                ;
                                ;  Store character into result buffer
                                ;
005252  110021                  3$:     MOVB    R0,(R1)+        ;Store char into result buffer
005254  000760                          BR      4$              ;Go get next char
```

```
                                        ;
                                        ;  Finished
                                        ;
   005256  105011                       9$:     CLRB    (R1)            ;Store null at end of string
   005260  162701  000000G                      SUB     #BLK0,R1        ;Determine length of string
   005264  010100                               MOV     R1,R0           ;Return in R0
   005266  012605                               MOV     (SP)+,R5
   005270  012601                               MOV     (SP)+,R1
   005272  000207                               RETURN
                                        ;
                                        ;  Invalid string
                                        ;
   005274                               10$:    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.
                                ;
005304  010046                  GTRD50: MOV     R0,-(SP)
005306  010146                          MOV     R1,-(SP)
                                ;  SKIP LEADING BLANKS.
005310  121327  000040          2$:     CMPB    (R3),#'         ;IS THIS CHAR ABLANK?
005314  001002                          BNE     1$              ;BRANCH IF NOT
005316  005203                          INC     R3              ;KEEP SKIPPING
005320  000773                          BR      2$
005322  012746  022000          1$:     MOV     #22000,-(SP)    ;TELL US WHEN TO STOP
005326  005001                  8$:     CLR     R1              ;FORM VALUE IN R1
005330  111300                  7$:     MOVB    (R3),R0         ;GET NEXT CHAR
                                ;  CHECK FOR WILDCARD CHARACTER ('*')
005332  120027  000052                  CMPB    R0,#'*          ;WILDCARD?
005336  001003                          BNE     10$             ;BR IF NOT
005340  012700  000035                  MOV     #35,R0          ;RETURN CODE FOR *
005344  000434                          BR      5$
                                ;  CHECK FOR DIGIT
005346  120027  000060          10$:    CMPB    R0,#'0
005352  103406                          BLO     3$              ;BR IF DELIMITER
005354  120027  000071                  CMPB    R0,#'9
005360  101005                          BHI     4$              ;BR IF NOT DIGIT
005362  062700  177756                  ADD     #<36-'0>,R0     ;CONVERT DIGIT TO RAD50 VAL
005366  000423                          BR      5$
                                ;  HIT DELIMITER
005370  005000                  3$:     CLR     R0              ;CONVERT TO SPACE
005372  000422                          BR      6$
                                ;  CHECK FOR ALPHA CHARACTER
005374  120027  000141          4$:     CMPB    R0,#141         ;IS THIS A LOWER CASE LETTER?
005400  103406                          BLO     12$             ;BR IF NOT
005402  120027  000172                  CMPB    R0,#172         ;LOWER CASE Z
005406  101370                          BHI     3$              ;BR IF MUST BE DELIMITER
005410  162700  000040                  SUB     #40,R0          ;CONVERT LOWER CASE TO UPPER CASE
005414  000406                          BR      13$
005416  120027  000101          12$:    CMPB    R0,#'A
005422  103762                          BLO     3$              ;BRANCH IF DELIMITER
005424  120027  000132                  CMPB    R0,#'Z
005430  101357                          BHI     3$              ;BRANCH IF DELIMITER
005432  062700  177700          13$:    ADD     #<1-'A>,R0      ;CONVERT TO RAD50 VAL
005436  005203                  5$:     INC     R3              ;POINT TO NEXT CHAR
                                ;  MULTIPLY PREVIOUS VALUE BY 50 AND ADD NEW VALUE
005440  070127  000050          6$:     MUL     #50,R1          ;MULTIPLY PREVIOUS VALUE BY 50
005444  060001                          ADD     R0,R1           ;ADD NEW VALUE
005446  006316                          ASL     @SP             ;TEST FOR END OF 3 CHARS
005450  103327                          BCC     7$              ;BRANCH IF NOT END OF 3
005452  005716                          TST     @SP             ;FINISHED 6 CHARACTERS?
005454  001403                          BEQ     9$              ;BRANCH IF YES
005456  010167  0000000                 MOV     R1,R50BUF       ;SAVE 1ST 3 CHARS
005462  000721                          BR      8$
005464  010167  0000020         9$:     MOV     R1,<R50BUF+2>   ;SAVE 2ND 3 CHARS
```

```
   005470  005726                            TST     (SP)+           ;CLEAN OFF STACK
                                     ;   SKIP ANY CHARACTERS IN NAME AFTER SIXTH
   005472  112300                    11$:    MOVB    (R3)+,R0        ;GET NEXT CHAR
   005474  004767  007060                    CALL    CHKDLM          ;SEE IF IT IS A DELIMITER
   005500  103374                            BCC     11$             ;LOOP IF NOT
   005502  005303                            DEC     R3              ;POINT TO DELIMITER
   005504  012601                            MOV     (SP)+,R1
   005506  012600                            MOV     (SP)+,R0
   005510  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.
                          ;
005512  010146            PRTPCT: MOV     R1,-(SP)
005514  010446                    MOV     R4,-(SP)
005516  010546                    MOV     R5,-(SP)
                          ;
                          ;  Get dividend and multiply by 100.
                          ;
005520  012104                    MOV     (R1)+,R4        ;GET HIGH-ORDER VALUE
005522  012105                    MOV     (R1)+,R5        ;GET LOW-ORDER VALUE
005524  012700  000144            MOV     #100.,R0        ;SET MULTIPLIER
005530  004767  000334            CALL    MUL32           ;MULTIPLY BY 100.
                          ;
                          ;  Divide to compute percentage
                          ;
005534  004767  000232            CALL    DIV32           ;DIVIDE TO COMPUTE PERCENTAGE
                          ;
                          ;  16-bit quotient is now in R5.
                          ;  Print the value.
                          ;
005540  004767  000376            CALL    PRTDEC          ;PRINT THE VALUE
                          ;
                          ;  Print percent sign
                          ;
005544  012700  000045            MOV     #'%,R0          ;GET PERCENT SIGN
005550                            .TTYOUT                 ;PRINT IT
                          ;
                          ;  Finished
                          ;
005554  012605                    MOV     (SP)+,R5
005556  012604                    MOV     (SP)+,R4
005560  012601                    MOV     (SP)+,R1
005562  000207                    RETURN
```

```
                                        .SBTTL  PRTR50 -- Print a RAD50 value
                                ;---------------------------------------------------------------------------
                                ;  PRTR50 is called to print a Rad-50 value.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to be printed.
                                ;
  005564  010146                PRTR50: MOV     R1,-(SP)
  005566  010246                        MOV     R2,-(SP)
                                ;
                                ;  Convert value to ascii string and stack the characters.
                                ;
  005570  012702  000003                MOV     #3.,R2          ;CONVERT 3 CHARS
  005574  005046                        CLR     -(SP)           ;PUT NULL ON STACK TO SIGNAL END
  005576  010001                        MOV     R0,R1           ;GET VALUE TO BE CONVERTED
  005600  005000                1$:     CLR     R0              ;CLEAR HIGH-ORDER VALUE
  005602  071027  000050                DIV     #50,R0          ;DIVIDE R0-R1 BY 50
  005606  116146  000000G               MOVB    R50CHR(R1),-(SP);CONVERT REMAINDER TO ASCII CHARACTER & STACK
  005612  010001                        MOV     R0,R1           ;GET QUOTIENT
  005614  077207                        SOB     R2,1$           ;LOOP IF MORE CHARS TO CONVERT
                                ;
                                ;  Finished conversion.  Print the result.
                                ;
  005616  012600                2$:     MOV     (SP)+,R0        ;GET CHAR TO PRINT
  005620  001403                        BEQ     3$              ;BR IF HIT END
  005622                                .TTYOUT                 ;PRINT THE ASCII CHARACTER
  005626  000773                        BR      2$              ;KEEP GOING
                                ;
                                ;  Finished
                                ;
  005630  012602                3$:     MOV     (SP)+,R2
  005632  012601                        MOV     (SP)+,R1
  005634  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.
                               ;
 005636  010146                PRTFNM: MOV     R1,-(SP)
 005640  010246                        MOV     R2,-(SP)
 005642  010446                        MOV     R4,-(SP)
 005644  010546                        MOV     R5,-(SP)
 005646  010005                        MOV     R0,R5           ;Get pointer to file name
 005650  012704  000002                MOV     #2,R4           ;Convert and print 2 words
                               ;
                               ;  Convert RAD50 name to ascii string and stack the characters
                               ;
 005654  005046                4$:     CLR     -(SP)           ;Put null on stack to mark the end
 005656  012702  000003                MOV     #3,R2           ;Convert 3 characters from this word
 005662  012501                        MOV     (R5)+,R1        ;Get RAD50 value to be converted
 005664  005000                1$:     CLR     R0              ;Clear high-order word for divide
 005666  071027  000050                DIV     #50,R0          ;Divide R0-R1 by 50
 005672  116146  000000G               MOVB    R50CHR(R1),-(SP);Stack next char of file name
 005676  010001                        MOV     R0,R1           ;Get quotient
 005700  077207                        SOB     R2,1$           ;Loop if more chars to convert
                               ;
                               ;  Finished converting a word.  Move characters to buffer.
                               ;
 005702  012600                2$:     MOV     (SP)+,R0        ;Get next character of name
 005704  001405                        BEQ     5$              ;Br if hit end of word
 005706  120027  000040                CMPB    R0,#40          ;Is this character a space?
 005712  001773                        BEQ     2$              ;Don't print spaces
 005714  110023                        MOVB    R0,(R3)+        ;Move character to buffer
 005716  000771                        BR      2$
 005720  077423                5$:     SOB     R4,4$           ;Loop if 2nd word to convert
                               ;
                               ;  Finished
                               ;
 005722  012605                3$:     MOV     (SP)+,R5
 005724  012604                        MOV     (SP)+,R4
 005726  012602                        MOV     (SP)+,R2
 005730  012601                        MOV     (SP)+,R1
 005732  000207                        RETURN
```

```
   005734  010246                    .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.
                             ;
   005734  010246            DIVIDE: MOV     R2,-(SP)
   005736  005000                    CLR     R0                      ;INITIALIZE REMAINDER
   005740  012702  000037            MOV     #31.,R2                 ;GET SHIFT COUNT
   005744  006305            1$:     ASL     R5                      ;SHIFT BIT OUT OF LOW ORDER
   005746  006104                    ROL     R4                      ;SHIFT THROUGH HIGH ORDER
   005750  006100                    ROL     R0                      ;INTO R0
   005752  020003                    CMP     R0,R3                   ;GOT  ENOUGH TO SUBTRACT YET?
   005754  103402                    BLO     2$                      ;BR IF NOT
   005756  160300                    SUB     R3,R0                   ;SUBTRACT DIVISOR
   005760  005205                    INC     R5                      ;INCREASE QUOTIENT
   005762  005302            2$:     DEC     R2                      ;COUNT # BITS SHIFTED
   005764  100367                    BPL     1$                      ;BR IF MORE TO DO
   005766  012602                    MOV     (SP)+,R2
   005770  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
                               ;
005772  010246                 DIV32:  MOV     R2,-(SP)
005774  010346                         MOV     R3,-(SP)
005776  005002                         CLR     R2              ;INITIALIZE REMAINDER (R2-R3)
006000  005003                         CLR     R3
006002  012700  000040                 MOV     #32.,R0         ;GET SHIFT COUNT
006006  073427  000001         1$:     ASHC    #1,R4           ;SHIFT BIT OUT OF DIVIDEND
006012  006103                         ROL     R3              ;AND INTO REMAINDER
006014  006102                         ROL     R2
006016  020267  000000G                CMP     R2,DIVSOR       ;GOT ENOUGH TO SUBTRACT YET?
006022  103412                         BLO     2$              ;BR IF NOT
006024  101003                         BHI     3$              ;BR IF YES
006026  020367  000002G                CMP     R3,DIVSOR+2     ;CHECK LOW-ORDER PART
006032  103406                         BLO     2$              ;BR IF NOT ENOUGH YET
006034  166703  000002G        3$:     SUB     DIVSOR+2,R3     ;SUBTRACT LOW-ORDER DIVISOR
006040  005602                         SBC     R2              ;PROPOGATE BORROW
006042  166702  000000G                SUB     DIVSOR,R2       ;SUBTRACT HIGH-ORDER DIVISOR
006046  005205                         INC     R5              ;INCREASE QUOTIENT
006050  077022                 2$:     SOB     R0,1$           ;DO FULL DIVIDE
006052  010267  000000G                MOV     R2,REMNDR       ;STORE HIGH-ORDER REMAINDER
006056  010367  000002G                MOV     R3,REMNDR+2     ;STORE LOW-ORDER REMAINDER
006062  012603                         MOV     (SP)+,R3
006064  012602                         MOV     (SP)+,R2
006066  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.
                                ;
 006070  010046                 MUL32:  MOV     R0,-(SP)
 006072  010246                         MOV     R2,-(SP)
 006074  010346                         MOV     R3,-(SP)
 006076  010402                         MOV     R4,R2           ;COPY VALUE TO BE MULTIPLIED
 006100  010503                         MOV     R5,R3
 006102  005004                         CLR     R4              ;FORM PRODUCT IN R4-R5
 006104  005005                         CLR     R5
 006106  000241                 1$:     CLC
 006110  006000                         ROR     R0              ;SHOULD WE ADD IN THIS TIME?
 006112  103003                         BCC     2$              ;BR IF NOT
 006114  060305                         ADD     R3,R5           ;ADD LOW-ORDER PART
 006116  005504                         ADC     R4              ;PROPOGATE CARRY
 006120  060204                         ADD     R2,R4           ;ADD HIGH-ORDER PART
 006122  073227  000001         2$:     ASHC    #1,R2           ;SHIFT MULTIPLICAN VALUE
 006126  005700                         TST     R0              ;MORE TO MULTIPLY?
 006130  001366                         BNE     1$              ;LOOP IF YES
 006132  012603                         MOV     (SP)+,R3
 006134  012602                         MOV     (SP)+,R2
 006136  012600                         MOV     (SP)+,R0
 006140  000207                         RETURN
```

```
                                         .SBTTL  PRTDEC -- Print a decimal value
                                 ;-----------------------------------------------------------------
                                 ;  PRTDEC IS CALLED TO PRINT THE DECIMAL VALUE IN R5.
                                 ;  ALL REGISTERS ARE PRESERVED.
                                 ;
 006142  010046                  PRTDEC: MOV     R0,-(SP)
 006144  010246                          MOV     R2,-(SP)
 006146  010346                          MOV     R3,-(SP)
 006150  010446                          MOV     R4,-(SP)
 006152  010546                          MOV     R5,-(SP)
 006154  012702  000000G                 MOV     #PBUFND,R2      ;POINT TO END OF PRINT BUFFER
 006160  005004                  1$:     CLR     R4              ;CLEAR HIGH-ORDER FOR DIVIDE
 006162  071427  000012                  DIV     #10.,R4         ;DIVIDE R4-R5 BY 10.
 006166  062705  000060                  ADD     #'0,R5          ;CONVERT REMAINDER TO ASCII DIGIT
 006172  110542                          MOVB    R5,-(R2)        ;STORE THE CHARACTER
 006174  010405                          MOV     R4,R5           ;ANYTHING LEFT TO CONVERT
 006176  001370                          BNE     1$              ;BR IF YES
                                 ;  VALUE IS CONVERTED, PRINT IT.
 006200                                  .PRINT  R2
 006204  012605                          MOV     (SP)+,R5
 006206  012604                          MOV     (SP)+,R4
 006210  012603                          MOV     (SP)+,R3
 006212  012602                          MOV     (SP)+,R2
 006214  012600                          MOV     (SP)+,R0
 006216  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.
                                ;
 006220  010346                 PRTLN:  MOV     R3,-(SP)
 006222  010546                         MOV     R5,-(SP)
 006224  006205                         ASR     R5              ;Convert job index # to job #
 006226  012703  000002                 MOV     #2,R3           ;Print value in 2 character field
 006232  004767  000006                 CALL    PRTFIX          ;Print it
 006236  012605                         MOV     (SP)+,R5
 006240  012603                         MOV     (SP)+,R3
 006242  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.
                                ;
006244  010046                  PRTFIX: MOV     R0,-(SP)
006246  010246                          MOV     R2,-(SP)
006250  010346                          MOV     R3,-(SP)
006252  010446                          MOV     R4,-(SP)
006254  010546                          MOV     R5,-(SP)
006256  012702  000000G                 MOV     #PBUFND,R2      ;POINT TO END OF CONVERSION BUFFER
006262  005004                  1$:     CLR     R4              ;CLEAR HIGH-ORDER FOR DIVIDE
006264  071427  000012                  DIV     #10.,R4         ;DIVIDE R4-R5 BY 10.
006270  062705  000060                  ADD     #'0,R5          ;CONVERT REMAINDER TO ASCII DIGIT
006274  110542                          MOVB    R5,-(R2)        ;MOVE TO PRINT BUFFER
006276  005303                          DEC     R3              ;COUNT DOWN # OF COLUMNS USED
006300  010405                          MOV     R4,R5           ;GET REMAINING QUOTIENT
006302  001367                          BNE     1$              ;BR IF MORE TO CONVERT
006304  005703                          TST     R3              ;DO WE NEED TO PUT IN PADDING SPACES?
006306  003403                          BLE     2$              ;BR IF NOT
006310  112742  000040          3$:     MOVB    #' ,-(R2)       ;INSERT LEADING SPACES
006314  077303                          SOB     R3,3$
006316                          2$:     .PRINT  R2              ;PRINT THE FINAL RESULT
006322  012605                          MOV     (SP)+,R5
006324  012604                          MOV     (SP)+,R4
006326  012603                          MOV     (SP)+,R3
006330  012602                          MOV     (SP)+,R2
006332  012600                          MOV     (SP)+,R0
006334  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.
                                      ;
 006336  020527  000011               PRTDC2: CMP     R5,#9.          ;DOES VALUE USE 1 OR MORE DIGITS?
 006342  101006                               BHI     1$              ;BR IF USES MORE THAN 1 DIGIT
 006344  010046                               MOV     R0,-(SP)
 006346                                       .TTYOUT #'0             ;PRINT LEADING ZERO
 006356  012600                               MOV     (SP)+,R0
 006360  004767  177556               1$:     CALL    PRTDEC          ;NOW PRINT VALUE
 006364  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.
                                      ;
 006366  020527  000143               PRTDC3: CMP     R5,#99.         ;Will value print with 3 digits?
 006372  101004                               BHI     1$              ;Br if yes
 006374                                       .TTYOUT #40             ;Print a blank if not
 006404  020527  000011               1$:     CMP     R5,#9.          ;Will value print with 2 digits?
 006410  101004                               BHI     2$              ;Br if yes
 006412                                       .TTYOUT #40             ;Print second blank if not
 006422  004767  177514               2$:     CALL    PRTDEC          ;Print actual value
 006426  000207                               RETURN

                                              .SBTTL  PRTSPC -- Print specified number of spaces
                                      ;----------------------------------------------------------------------
                                      ;  Print the specified number of spaces.
                                      ;
                                      ;  Inputs:
                                      ;   R3 = Number of spaces to print
                                      ;
 006430  010346                       PRTSPC: MOV     R3,-(SP)
 006432  001405                               BEQ     9$
 006434                               1$:     .TTYOUT #40             ;Print a space
 006444  077305                               SOB     R3,1$           ;Loop if more to print
 006446  012603                       9$:     MOV     (SP)+,R3
 006450  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.
                                     ;
006452  010246                       PRTTTP: MOV     R2,-(SP)
006454  010446                               MOV     R4,-(SP)
006456  016104  0000000                      MOV     LTRMTP(R1),R4    ;Get terminal type flags
006462  032761  0000000 0000000              BIT     #$KINIT,LSW(R1)  ;Has line initializion been done yet?
006470  001002                               BNE     1$               ;Br if yes
006472  016104  0000000                      MOV     ITRMTP(R1),R4    ;Get sysgen terminal type code
006476  012702  000022               1$:     MOV     #NTRMTP,R2       ;Get # terminal types
006502  020462  006544'              2$:     CMP     R4,TTFTBL(R2)    ;Is this the terminal type?
006506  001407                               BEQ     3$               ;Br if yes
006510  162702  000002                       SUB     #2,R2            ;More to check?
006514  002372                               BGE     2$               ;Br if yes
006516                               4$:     .PRINT  #TTNXXX          ;Unknown terminal type
006524  000404                               BR      9$
006526  016202  006570'              3$:     MOV     TTNTBL(R2),R2    ;Get pointer to terminal name string
006532                                       .PRINT  R2               ;Print terminal name
                                     ;
                                     ;  Finished
                                     ;
006536  012604                       9$:     MOV     (SP)+,R4
006540  012602                               MOV     (SP)+,R2
006542  000207                               RETURN

                                     ;
                                     ;  Terminal type flag table
                                     ;
006544  0000000                      TTFTBL: .WORD   VT52             ;VT52
006546  0000000                              .WORD   VT100            ;VT100
006550  0000000                              .WORD   HAZEL            ;HAZELTINE
006552  0000000                              .WORD   ADM3A            ;ADM3A
006554  0000000                              .WORD   LA36             ;LA36
006556  0000000                              .WORD   LA120            ;LA120
006560  0000000                              .WORD   DIABLO           ;DIABLO
006562  0000000                              .WORD   QUME             ;QUME
006564  0000000                              .WORD   VT2007           ;VT200 -- 7 bit control codes
006566  0000000                              .WORD   VT2008           ;VT200 -- 8 bit control codes
        000022                       NTRMTP  =       <.-TTFTBL>-2     ;Highest terminal type index
                                     ;
                                     ;  Terminal type name pointer table
                                     ;
006570  006625'                      TTNTBL: .WORD   TTNV52           ;VT52
006572  006636'                              .WORD   TTNV10           ;VT100
006574  006660'                              .WORD   TTNHZL           ;HAZELTINE
006576  006671'                              .WORD   TTNADM           ;ADM3A
006600  006702'                              .WORD   TTNL36           ;LA36
006602  006713'                              .WORD   TTNL12           ;LA120
006604  006724'                              .WORD   TTNDIA           ;DIABLO
006606  006735'                              .WORD   TTNQUM           ;QUME
006610  006647'                              .WORD   TTNV20           ;VT200
006612  006647'                              .WORD   TTNV20           ;VT200
```

```
     58                                         ;
     59                                         ;   Terminal name strings
     60                                         ;
     61                                                  .NLIST  BEX
     62 006614     165      156      153   TTNXXX: .ASCII  /unknown /<200>
     63 006625     126      124      065   TTNV52: .ASCII  /VT52    /<200>
     64 006636     126      124      061   TTNV10: .ASCII  /VT100   /<200>
     65 006647     126      124      062   TTNV20: .ASCII  /VT200   /<200>
     66 006660     110      141      172   TTNHZL: .ASCII  /Hazeltne/<200>
     67 006671     101      104      115   TTNADM: .ASCII  /ADM3A   /<200>
     68 006702     114      101      063   TTNL36: .ASCII  /LA36    /<200>
     69 006713     114      101      061   TTNL12: .ASCII  /LA120   /<200>
     70 006724     104      151      141   TTNDIA: .ASCII  /Diablo  /<200>
     71 006735     121      165      155   TTNQUM: .ASCII  /Qume    /<200>
     72                                                  .EVEN
     73                                                  .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.
                               ;
 006746  010446                EDTFIL: MOV     R4,-(SP)
                               ;
                               ;  Edit device name
                               ;
 006750  012400                        MOV     (R4)+,R0        ;GET DEVICE NAME
 006752  001404                        BEQ     1$              ;BR IF NO DEVICE SPECIFIED
 006754  004767  000046                CALL    EDTR50          ;EDIT INTO BUFFER
 006760  112723  000072                MOVB    #':,(R3)+       ;TERMINATE DEV NAME WITH COLON
                               ;
                               ;  Get file name
                               ;
 006764  012400                1$:     MOV     (R4)+,R0        ;GET 1ST 3 CHARS OF FILE NAME
 006766  001414                        BEQ     3$              ;BR IF NO FILE NAME
 006770  004767  000032                CALL    EDTR50          ;EDIT IN 1ST 3 CHARS
 006774  012400                        MOV     (R4)+,R0        ;GET 2ND 3 CHARS
 006776  001402                        BEQ     2$              ;BR IF ALL BLANK
 007000  004767  000022                CALL    EDTR50          ;EDIT INTO BUFFER
                               ;
                               ;  Put in extension
                               ;
 007004  012400                2$:     MOV     (R4)+,R0        ;GET EXTENSION
 007006  001404                        BEQ     3$              ;BR IF NO EXTENSION
 007010  112723  000056                MOVB    #'.,(R3)+       ;PUT PERIOD AT END OF FILE NAME
 007014  004767  000006                CALL    EDTR50          ;EDIT IN EXTENSION
                               ;
                               ;  Finished
                               ;
 007020  105013                3$:     CLRB    (R3)            ;PUT IN NULL AT END OF STRING
 007022  012604                        MOV     (SP)+,R4
 007024  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.
                                ;
007026  010146                  EDTR50: MOV     R1,-(SP)
007030  010246                          MOV     R2,-(SP)
                                ;
                                ;  See if value is wildcard ("*")
                                ;
007032  020027  000000G                 CMP     R0,#WLDNAM      ;Wildcard?
007036  001003                          BNE     5$              ;Br if not
007040  112723  000052                  MOVB    #'*,(R3)+       ;Store wildcard character
007044  000420                          BR      9$
                                ;
                                ;  Convert value to ascii characters and stack the characters
                                ;
007046  005046                  5$:     CLR     -(SP)           ;PUT NULL ON STACK TO SIGNAL END OF CHARS
007050  012702  000003                  MOV     #3,R2           ;GET # OF CHARS TO CONVERT
007054  010001                          MOV     R0,R1           ;GET VALUE TO CONVERT
007056  005000                  1$:     CLR     R0              ;CLEAR HIGH-ORDER FOR DIVIDE
007060  071027  000050                  DIV     #50,R0          ;DIVIDE R0-R1 BY 50
007064  005701                          TST     R1              ;IS THE CHARACTER A SPACE?
007066  001402                          BEQ     4$              ;BR IF YES
007070  116146  000000G                 MOVB    R50CHR(R1),-(SP);STACK THE CHARACTER
007074  010001                  4$:     MOV     R0,R1           ;GET QUOTIENT
007076  077211                          SOB     R2,1$           ;LOOP IF MORE TO CONVERT
                                ;
                                ;  Move characters from the stack to the result buffer
                                ;
007100  112623                  2$:     MOVB    (SP)+,(R3)+     ;MOVE CHAR TO RESULT
007102  001376                          BNE     2$              ;LOOP IF MORE TO MOVE
007104  005303                          DEC     R3              ;POINT PAST LAST CHAR
                                ;
                                ;  Finished
                                ;
007106  012602                  9$:     MOV     (SP)+,R2
007110  012601                          MOV     (SP)+,R1
007112  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
                                ;
 007114  010446                 PRTUNM: MOV     R4,-(SP)
 007116  010546                         MOV     R5,-(SP)
                                ;
                                ;  If line is not logged on, there is nothing to print
                                ;
 007120  032761  0000000 0000000        BIT     #$KINIT,LSW(R1) ;Has line been initialized?
 007126  001442                         BEQ     10$             ;Br if not
                                ;
                                ;  Did user request PPN display?
                                ;
 007130  105767  171013                 TSTB    SJSPPN          ;SHOW JOBS/PPN?
 007134  001017                         BNE     15$             ;BRANCH IF SO
                                ;
                                ;  See if user name is known
                                ;
 007136  010105                         MOV     R1,R5           ;GET JOB #
 007140  012704  000014                 MOV     #12.,R4         ;EACH JOB HAS A 12. CHAR USER NAME
 007144  070527  000006                 MUL     #6,R5           ;GET POINTER TO USER NAME FOR THIS JOB
 007150  062705  0000000                ADD     #LUNAME,R5
 007154  121527  000040                 CMPB    (R5),#'         ;IS USER NAME BLANK?
 007160  001405                         BEQ     15$             ;BR IF YES -- PRINT PPN INSTEAD
 007162                         16$:    .TTYOUT (R5)+           ;PRINT USER NAME
 007170  077404                         SOB     R4,16$
 007172  000420                         BR      10$
                                ;
                                ;  Don't know user name, print PPN
                                ;
 007174  016105  0000000        15$:    MOV     LPROJ(R1),R5    ;GET PROJECT #
 007200  001415                         BEQ     10$             ;BR IF NOT LOGGED ON
 007202                                 .PRINT  #PPNMSG
 007210  004767  176726         9$:     CALL    PRTDEC          ;PRINT PROJECT NUMBER
 007214  112700  000054                 MOVB    #',,R0
 007220                                 .TTYOUT
 007224  016105  0000000                MOV     LPROG(R1),R5    ;PRINT PROGRAMMER NUMBER
 007230  004767  176706                 CALL    PRTDEC
                                ;
                                ;  Finished
                                ;
 007234  012605                 10$:    MOV     (SP)+,R5
 007236  012604                         MOV     (SP)+,R4
 007240  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.
                                ;
 007242  010046                 PRTTIM: MOV     R0,-(SP)
 007244  010346                         MOV     R3,-(SP)
 007246  010446                         MOV     R4,-(SP)
 007250  010546                         MOV     R5,-(SP)
                                ;
                                ;  PRINT CONNECT TIME
                                ;
 007252                                 .PRINT  #CTMSG          ;PRINT CONNECT TIME HEADER
 007260  016705  0000000                MOV     MINTIM,R5       ;GET CURRENT MINUTE TIMER VALUE
 007264  166105  0000000                SUB     LCONTM(R1),R5   ;CALCULATE CONNECT TIME FOR LINE
 007270  005205                         INC     R5              ;CHARGE A MINIMUM OF 1 MINUTE
 007272  005004                 1$:     CLR     R4              ;CLEAR HIGH-ORDER FOR DIVIDE
 007274  012703  000074                 MOV     #60.,R3         ;SET TO DIVIDE BY 60.
 007300  004767  176430                 CALL    DIVIDE          ;DIVIDE BY 60
 007304  004767  177026                 CALL    PRTDC2          ;PRINT # HOURS CONNECTED
 007310  010005                         MOV     R0,R5           ;GET # MINUTES CONNECTED
 007312                                 .TTYOUT #':             ;PRINT COLON AFTER HOURS
 007322  004767  177010                 CALL    PRTDC2          ;PRINT # MINUTES CONNECTED
 007326                                 .PRINT  #COLOO          ;PRINT ':00' SECONDS
                                ;
                                ;  PRINT CPU TIME
                                ;
 007334                                 .PRINT  #CPUMSG         ;PRINT CPU HEADER MESSAGE
 007342  016104  0000000                MOV     LCPUHI(R1),R4   ;GET HIGH ORDER CPU TIME (CLOCK TICKS)
 007346  016105  0000000                MOV     LCPULO(R1),R5   ;GET LOW ORDER CPU TIME (CLOCK TICKS)
 007352  004767  000020                 CALL    PRTTMV          ;PRINT TIME VALUE
 007356                                 .PRINT  #CRLF           ;END PRINT LINE
 007364  012605                         MOV     (SP)+,R5
 007366  012604                         MOV     (SP)+,R4
 007370  012603                         MOV     (SP)+,R3
 007372  012600                         MOV     (SP)+,R0
 007374  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.
                                ;
 007376  010346                 PRTTMV: MOV     R3,-(SP)
 007400  010446                         MOV     R4,-(SP)
 007402  010546                         MOV     R5,-(SP)
                                ;
                                ;  Convert time to seconds and fractions thereof
                                ;
 007404  066705  0000000                ADD     TK5VAL,R5       ;Round to nearest second
 007410  005504                         ADC     R4              ;Propogate round carry
 007412  016703  0000000                MOV     TK1SEC,R3       ;Get # clock ticks per second
 007416  004767  176312                 CALL    DIVIDE          ;Convert to seconds and fractions
                                ;
                                ;  We now have # seconds in R4-R5.
                                ;  Convert to minutes and seconds.
                                ;
 007422  012703  000074                 MOV     #60.,R3         ;60 seconds per minute
 007426  004767  176302                 CALL    DIVIDE          ;SPLIT INTO MINUTES AND SECONDS
 007432  010046                         MOV     R0,-(SP)        ;SAVE # SECONDS
                                ;
                                ;  Split time into hours and minutes
                                ;
 007434  004767  176274                 CALL    DIVIDE          ;SPLIT INTO # HOURS AND MINUTES
                                ;
                                ;  Print the complete time value
                                ;
 007440  004767  176672                 CALL    PRTDC2          ;PRINT  # HOURS
 007444  010005                         MOV     R0,R5           ;GET # MINUTES
 007446                                 .TTYOUT #':
 007456  004767  176654                 CALL    PRTDC2          ;PRINT # MINUTES
 007462                                 .TTYOUT #':
 007472  012605                         MOV     (SP)+,R5        ;GET # SECONDS
 007474  004767  176636                 CALL    PRTDC2          ;PRINT # SECONDS
                                ;
                                ;  Now convert original clock-tick time into tenths of seconds
                                ;
 007500  011605                         MOV     (SP),R5         ;Recover original low-order value
 007502  016604  000002                 MOV     2(SP),R4        ;Recover original high-order value
                                ;
                                ;  Convert time value to seconds and fractions thereof
                                ;
 007506  016703  0000000                MOV     TK1SEC,R3       ;Get # clock ticks per second
 007512  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.
                                     ;
   007516  010046                            MOV     R0,-(SP)        ;Save fractional remainder
   007520  012700  000012                    MOV     #10.,R0         ;Multiply whole seconds value by 10.
   007524  004767  176340                    CALL    MUL32           ;Get approximate # 1/10
   007530  010467  000000G                   MOV     R4,CPUAH        ;Save high-order part
   007534  010567  000000G                   MOV     R5,CPUAL        ;Save low-order part
                                     ;
                                     ;  Now convert fractional reminder into 1/10 second units
                                     ;
   007540  012605                            MOV     (SP)+,R5        ;Get fractional remainder
   007542  005004                            CLR     R4              ;Clear high-order for divide
   007544  071467  000000G                   DIV     TK1VAL,R4       ;Convert remainder to 1/10 sec units
   007550  060467  000000G                   ADD     R4,CPUAL        ;Add to low-order part
   007554  005567  000000G                   ADC     CPUAH           ;Propogate carry to high-order part
                                     ;
                                     ;  Finished
                                     ;
   007560  012605                            MOV     (SP)+,R5
   007562  012604                            MOV     (SP)+,R4
   007564  012603                            MOV     (SP)+,R3
   007566  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
                         ;
007570  010346           PRTTMD: MOV     R3,-(SP)
007572  010446                   MOV     R4,-(SP)
007574  010546                   MOV     R5,-(SP)
007576  062705  000005           ADD     #5.,R5          ;ROUND TO NEAREST SECOND
007602  005504                   ADC     R4
007604  012703  000012           MOV     #10.,R3         ;GET DIVISOR
007610  004767  176120           CALL    DIVIDE          ;SPLIT INTO SECONDS AND TENTHS
007614  012703  000074           MOV     #60.,R3
007620  004767  176110           CALL    DIVIDE          ;SPLIT INTO MINUTES AND SECONDS
007624  010046                   MOV     R0,-(SP)        ;SAVE # SECONDS
007626  004767  176102           CALL    DIVIDE          ;SPLIT INTO # HOURS AND MINUTES
007632  020527  000030           CMP     R5,#24.         ;MORE THAN 1 DAY?
007636  103415                   BLO     1$              ;BR IF NOT
007640  071427  000030           DIV     #24.,R4         ;GET DAYS AND HOURS WITHIN A DAY
007644  010546                   MOV     R5,-(SP)        ;SAVE HOURS (REMAINDER)
007646  010405                   MOV     R4,R5           ;GET NUMBER OF DAYS
007650  004767  176266           CALL    PRTDEC          ;PRINT NUMBER OF DAYS
007654  010046                   MOV     R0,-(SP)        ;SAVE MINUTES
007656                           .TTYOUT #40             ;PRINT A SPACE FOLLOWING THE DAY VALUE
007666  012600                   MOV     (SP)+,R0        ;RECOVER MINUTES
007670  012605                   MOV     (SP)+,R5        ;GET NUMBER OF HOURS WITHIN THE DAY
007672  004767  176440   1$:     CALL    PRTDC2          ;PRINT  # HOURS
007676  010005                   MOV     R0,R5           ;GET # MINUTES
007700                           .TTYOUT #':
007710  004767  176422           CALL    PRTDC2          ;PRINT # MINUTES
007714                           .TTYOUT #':
007724  012605                   MOV     (SP)+,R5        ;GET # SECONDS
007726  004767  176404           CALL    PRTDC2          ;PRINT # SECONDS
                         ;
                         ;  Finished
                         ;
007732  012605                   MOV     (SP)+,R5
007734  012604                   MOV     (SP)+,R4
007736  012603                   MOV     (SP)+,R3
007740  000207                   RETURN
```

```
                                           .SBTTL  PRTDAT -- Print the current date
                                   ;-------------------------------------------------------------------
                                   ;  PRTDAT IS CALLED TO PRINT THE CURRENT DATE.
                                   ;  ALL REGISTERS ARE PRESERVED.
                                   ;
 007742  010046                    PRTDAT: MOV     R0,-(SP)
 007744  010246                            MOV     R2,-(SP)
 007746  010546                            MOV     R5,-(SP)
 007750                                    .DATE                    ;GET CURRENT DATE
 007756  010046                            MOV     R0,-(SP)         ;SAVE DATE
 007760  006300                            ASL     R0               ;LEFT JUSTIFY VALUE
 007762  012702  000005                    MOV     #5,R2            ;SHIFT OFF 5 BITS
 007766  005005                            CLR     R5               ;INTO R5
 007770  006100                    1$:     ROL     R0               ;SHIFT MONTH VALUE INTO R5
 007772  006105                            ROL     R5
 007774  077203                            SOB     R2,1$            ;LOOP IF MORE BITS TO SHIFT
 007776  005305                            DEC     R5               ;GET MONTH VALUE IN RANGE 0-11
 010000  010546                            MOV     R5,-(SP)         ;SAVE MONTH VALUE
                                   ;  PRINT DAY NUMBER
 010002  012702  000005                    MOV     #5,R2            ;GET 5 BITS OF DAY VALUE
 010006  005005                            CLR     R5
 010010  006100                    2$:     ROL     R0               ;SHIFT BITS INTO R5
 010012  006105                            ROL     R5
 010014  077203                            SOB     R2,2$
 010016  004767  176120                    CALL    PRTDEC           ;PRINT DAY-OF-MONTH
 010022                                    .TTYOUT #'-              ;PUT IN SEPARATOR
                                   ;  PRINT MONTH NAME
 010032  011605                            MOV     (SP),R5          ;GET MONTH INDEX
 010034  006305                            ASL     R5               ;*2
 010036  062605                            ADD     (SP)+,R5         ;*3
 010040  062705  000000G                   ADD     #MONTAB,R5       ;POINT INTO ASCII MONTH TABLE
 010044                                    .TTYOUT (R5)+            ;PRINT NAME OF MONTH
 010052                                    .TTYOUT (R5)+
 010060                                    .TTYOUT (R5)
 010066                                    .TTYOUT #'-              ;PUT IN SEPARATOR
                                   ;  PRINT YEAR
 010076  012605                            MOV     (SP)+,R5         ;GET BACK ORIGINAL DATE VALUE
 010100  042705  177740                    BIC     #<^C37>,R5       ;CLEAR ALL BUT YEAR FIELD
 010104  062705  000110                    ADD     #72.,R5          ;YEAR # IS RELATIVE TO 1972
 010110  004767  176222                    CALL    PRTDC2           ;PRINT YEAR VALUE
 010114  012605                            MOV     (SP)+,R5
 010116  012602                            MOV     (SP)+,R2
 010120  012600                            MOV     (SP)+,R0
 010122  000207                            RETURN
```

```
                                        .SBTTL  PRTTOD -- Print the time of day
                                ;-----------------------------------------------------------
                                ;  PRTTOD IS CALLED TO PRINT THE TIME OF DAY.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
 010124  010046                 PRTTOD: MOV     R0,-(SP)
 010126  010346                         MOV     R3,-(SP)
 010130  010446                         MOV     R4,-(SP)
 010132  010546                         MOV     R5,-(SP)
 010134                                 .GTIM   #XAREA,#CPUAH   ;GET # CLOCK TICKS SINCE 00:00
 010154  016704  000000G                MOV     CPUAH,R4        ;GET HIGH-ORDER VALUE
 010160  016705  000000G                MOV     CPUAL,R5        ;GET LOW-ORDER VALUE
 010164  016703  000000G                MOV     TK1SEC,R3       ;Get # clock ticks per second
 010170  004767  175540                 CALL    DIVIDE          ;Get # seconds past midnight
 010174  012703  000074                 MOV     #60.,R3         ;DIVIDE BY 60 TO SPLIT INTO SEC & MIN
 010200  004767  175530                 CALL    DIVIDE
 010204  010046                         MOV     R0,-(SP)        ;SAVE # SECONDS INTO MINUTE
 010206  004767  175522                 CALL    DIVIDE          ;GET # MINUTES & # HOURS
 010212  004767  176120                 CALL    PRTDC2          ;PRINT HOUR VALUE
 010216  010005                         MOV     R0,R5           ;GET # MINUTE VALUE
 010220                                 .TTYOUT #':             ;PUT IN COLON SEPARATOR
 010230  004767  176102                 CALL    PRTDC2          ;PRINT # MINUTES
 010234                                 .TTYOUT #':             ;ANOTHER COLON
 010244  012605                         MOV     (SP)+,R5        ;GET # SECONDS
 010246  004767  176064                 CALL    PRTDC2          ;PRINT # SECONDS
 010252  012605                         MOV     (SP)+,R5
 010254  012604                         MOV     (SP)+,R4
 010256  012603                         MOV     (SP)+,R3
 010260  012600                         MOV     (SP)+,R0
 010262  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.
                                ;
 010264  010046                 DATTIM: MOV     R0,-(SP)
 010266                                 .DATE                   ;GET CURRENT DATE
 010274  005700                         TST     R0              ;WAS DATE ENTERED BY OPERATOR?
 010276  001412                         BEQ     1$              ;BR IF NOT
 010300  004767  177436                 CALL    PRTDAT          ;PRINT CURRENT DATE
 010304                                 .PRINT  #SPACE2         ;PRINT 2 SPACES
 010312  004767  177606                 CALL    PRTTOD          ;PRINT TIME OF DAY
 010316                                 .PRINT  #CRLF           ;PRINT CR-LF
 010324  012600                 1$:     MOV     (SP)+,R0
 010326  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.
                                ;
010330  010046                  SEARCH: MOV     R0,-(SP)
010332  010146                          MOV     R1,-(SP)
010334  010246                          MOV     R2,-(SP)
010336  010546                          MOV     R5,-(SP)
                                ;
                                ;  Skip leading spaces, tabs, and form-feeds.
                                ;
010340  112300                  14$:    MOVB    (R3)+,R0        ;Get next character
010342  120027  000040                  CMPB    R0,#'           ;Is this a space?
010346  001774                          BEQ     14$             ;Skip leading spaces
010350  120027  000011                  CMPB    R0,#TAB         ;Skip leading tabs
010354  001771                          BEQ     14$
010356  120027  000014                  CMPB    R0,#FF          ;Skip leading form-feeds
010362  001766                          BEQ     14$
010364  005303                          DEC     R3              ;Point to first non-blank character
                                ;
                                ;  Move keyword to a holding buffer and convert lower-case letters
                                ;  to upper case
                                ;
010366  012705  000000G                 MOV     #KEYBUF,R5      ;Point to keyword holding buffer
010372  112300                  15$:    MOVB    (R3)+,R0        ;Get next character from command string
010374  120027  000141                  CMPB    R0,#141         ;Is this a lower-case letter?
010400  103405                          BLO     16$             ;Br if not
010402  120027  000172                  CMPB    R0,#172
010406  101035                          BHI     17$
010410  162700  000040                  SUB     #40,R0          ;Convert lower-case to upper-case
010414  120027  000132          16$:    CMPB    R0,#'Z          ;Is this character a letter?
010420  101020                          BHI     23$             ;Br if not
010422  120027  000101                  CMPB    R0,#'A
010426  103020                          BHIS    18$             ;Br if it is a letter
010430  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
010434  101404                          BLOS    24$             ;BRANCH IF MAYBE                ;**
010436  120027  000072                  CMPB    R0,#':          ;CHAR RANGE : TO @, IS IT : ?   ;**
010442  001412                          BEQ     18$             ;INCLUDE IF SO                  ;**
010444  000416                          BR      17$             ;ELSE DELIMITER                 ;**
```

```
 010446  120027  000060      24$:    CMPB    R0,#'0          ;See if in range 0 - 9          ;**
 010452  103006                      BHIS    18$             ;Br if digit
 010454  120027  000044              CMPB    R0,#'$          ;Allow "$" in command names
 010460  001403                      BEQ     18$
 010462  120027  000137      23$:    CMPB    R0,#'_          ;Also allow underscore
 010466  001005                      BNE     17$             ;Br if character is a delimiter
 010470  020527  177777G     18$:    CMP     R5,#KEYEND-1    ;Have we reached end of keyword buffer?
 010474  103336                      BHIS    15$             ;Br if yes
 010476  110025                      MOVB    R0,(R5)+        ;Move character to buffer
 010500  000734                      BR      15$
                             ;
                             ;  Reached end of keyword
                             ;
 010502  105015              17$:    CLRB    (R5)            ;Put null at end of keyword name
                             ;
                             ;  Skip up to start of next field after keyword
                             ;
 010504  005303                      DEC     R3              ;Point to delimiter
 010506  122327  000040      22$:    CMPB    (R3)+,#'        ;Skip spaces following keyword
 010512  001775                      BEQ     22$
 010514  005303                      DEC     R3              ;Point to 1st non-blank character
 010516  010346                      MOV     R3,-(SP)        ;Save delimiter pointer on stack for later
 010520  105767  000000G             TSTB    KEYBUF          ;Did we have a null keyword?
 010524  001433                      BEQ     21$             ;Br if yes
                             ;
                             ;  The keyword has been converted to upper-case and is stored in KEYBUF.
                             ;  We are now ready to begin comparing the keyword.
                             ;
 010526  012402                      MOV     (R4)+,R2        ;Get # of bytes per table entry
 010530  012405              5$:     MOV     (R4)+,R5        ;Point to asciz name string
 010532  001433                      BEQ     20$             ;Br if end of table hit
 010534  012703  000000G             MOV     #KEYBUF,R3      ;Point to our keyword
 010540  005001                      CLR     R1              ;Say no star seen yet
                             ;
                             ;  Begin to compare command string pointed to by R3 with
                             ;  keyword in table entry pointed to by R5
                             ;
 010542  112300              6$:     MOVB    (R3)+,R0        ;Get a char from the keyword
 010544  001414                      BEQ     1$              ;Br if hit end of keyword
 010546  105715              2$:     TSTB    (R5)            ;Hit end of name in table?
 010550  001410                      BEQ     4$              ;Br if yes -- no match
 010552  121527  000052              CMPB    (R5),#'*        ;Is next char a star?
 010556  001003                      BNE     19$             ;Br if not
 010560  005201                      INC     R1              ;Remember star seen
 010562  005205                      INC     R5              ;Point beyond star
 010564  000770                      BR      2$              ;Continue comparison
 010566  122500              19$:    CMPB    (R5)+,R0        ;Do names match
 010570  001764                      BEQ     6$              ;Yes -- keep checking
                             ;
                             ;  Names do not match
                             ;  Compare keyword with next entry in table
                             ;
 010572  060204              4$:     ADD     R2,R4           ;Point to next table entry
 010574  000755                      BR      5$              ;Go check it
                             ;
                             ;  Reached end of keyword. All chars match so far.
                             ;  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.
                                ;
010576  105715                  1$:     TSTB    (R5)            ;Are we at the end of the table entry also?
010600  001413                          BEQ     7$              ;yes, we have an exact match
010602  121527  000052                  CMPB    (R5),#'*        ;Is next table char "*"?
010606  001410                          BEQ     7$              ;If yes, then this is acceptable abbrev
010610  005701                          TST     R1              ;Have we already seen "*"?
010612  001006                          BNE     7$              ;If yes then we have an acceptable abbrev
                                ;
                                ;  We have an ambiguous keyword
                                ;
010614  010504                  21$:    MOV     R5,R4           ;Make R4 non-zero to signal ambigious case
010616  000261                          SEC                     ;Signal failure on return
010620  000404                          BR      10$             ;Finished
                                ;
                                ;  Cannot find keyword in table
                                ;
010622  005004                  20$:    CLR     R4              ;Signal unrecognizable command
010624  000261                          SEC                     ;Signal failure on return
010626  000401                          BR      10$
                                ;
                                ;  We found keyword in table
                                ;
010630  000241                  7$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
010632  012603                  10$:    MOV     (SP)+,R3        ;Recover command string pointer
010634  012605                          MOV     (SP)+,R5
010636  012602                          MOV     (SP)+,R2
010640  012601                          MOV     (SP)+,R1
010642  012600                          MOV     (SP)+,R0
010644  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.
                                    ;
010646                              FPRINT: .PRINT  #KMFTXT         ;PRINT ERROR MESSAGE HEADER
010654                                      .PRINT  R5              ;NOW PRINT ERROR MESSAGE
010660  004767  000044                      CALL    KMNERR          ;SEE IF WE SHOULD ABORT COMMAND FILES
010664  000207                              RETURN

                                            .SBTTL  PRTWRN -- Print warning message
                                    ;-----------------------------------------------------------------------
                                    ;  PRTWRN is called to print a KMON warning message.
                                    ;
                                    ;  Inputs:
                                    ;   R5 = Pointer to asciz text string.
                                    ;
010666                              PRTWRN::.PRINT  #WRNHED         ;Print warning heading
010674                                      .PRINT  R5              ;Print warning message
010700  152767  000000G 000000G             BISB    #SC$WRN,INDERR  ;Set warning severity for IND
010706  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.
                                    ;
010710  004767  177732              FKILL:  CALL    FPRINT          ;PRINT FATAL ERROR MESSAGE
010714  012706  000000G                     MOV     #KMSTK,SP       ;CLEAN OFF STACK
010720  004767  000004                      CALL    KMNERR          ;SEE IF WE SHOULD ABORT COMMAND FILES
010724  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
                                    ;
010730  010146                      KMNERR: MOV     R1,-(SP)
010732  152767  000000G 000000G             BISB    #SC$SEV,INDERR  ;SET ERROR SEVERITY FOR IND
010740  116701  000000G                     MOVB    CORUSR,R1       ;GET CURRENT JOB INDEX NUMBER
010744  122767  000000G 000000G             CMPB    #SC$SEV,ERRSEV  ;ABORT ON ERRORS?
010752  103410                              BLO     1$              ;BR IF NOT
010754  004767  172312                      CALL    ABRTCF          ;ABORT ALL OPEN COMMAND FILES
010760  032761  000000G 000000G             BIT     #$INDAB,LSW7(R1);SHOULD WE ABORT IND FILES ON ERRORS?
010766  001402                              BEQ     1$              ;BR IF NOT
010770  004767  172412                      CALL    INDABT          ;Abort execution of IND & nested command files
010774  012601                      1$:     MOV     (SP)+,R1
010776  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.
                                          ;
011000  012567  000000G                   ACRFN:  MOV     (R5)+,FILNAM    ;SET DEFAULT DEVICE
011004  011567  000006G                           MOV     (R5),FILNAM+6   ;SET DEFAULT EXTENSION
011010  005067  000010G                           CLR     FILNAM+8.       ;NO FILE SIZE
                                          ;  ACCRUE NEXT FIELD OF NAME
011014  004767  174264                    2$:     CALL    GTRD50          ;ACCRUE NAME IN RAD50 FORMAT
                                          ;  SEE IF THIS IS THE DEVICE OR FILE NAME
011020  121327  000072                            CMPB    (R3),#':        ;WAS THAT THE DEVICE NAME
011024  001011                                    BNE     1$              ;BR IF NOT. MUST BE FILE NAME
                                          ;  WE HAVE JUST ACCRUED THE DEVICE NAME
011026  016767  000000G 000000G                   MOV     R50BUF,FILNAM   ;SET DEVICE NAME
011034  005203                                    INC     R3              ;POINT PAST COLON
011036  111300                                    MOVB    (R3),R0         ;GET 1ST CHAR OF FILE NAME
011040  004767  003514                            CALL    CHKDLM          ;SEE IF IT IS A DELIMITER
011044  103363                                    BCC     2$              ;BR IF NOT DELIMITER
011046  000431                                    BR      3$              ;INVALID FILE NAME
                                          ;  WE JUST GOT THE FILE NAME
011050  016767  000000G 000002G           1$:     MOV     R50BUF,FILNAM+2 ;STORE THE FILE NAME
011056  001425                                    BEQ     3$              ;MUST NOT BE NULL
011060  016767  000002G 000004G                   MOV     R50BUF+2,FILNAM+4
011066  026727  000002G 132500                    CMP     FILNAM+2,#132500;WAS 1ST PART OF NAME "*"?
011074  001003                                    BNE     5$              ;BR IF NOT
011076  012767  132500  000004G                   MOV     #132500,FILNAM+4;IF YES THEN MAKE 2ND PART BE "*" TOO
                                          ;  SEE IF AN EXTENSION WAS SPECIFIED
011104  121327  000056                    5$:     CMPB    (R3),#'.        ;IS EXTENSION PRESENT?
011110  001006                                    BNE     4$              ;BR IF NOT
011112  005203                                    INC     R3              ;SKIP OVER PERIOD
011114  004767  174164                            CALL    GTRD50          ;ACCRUE THE EXTENSION
011120  016767  000000G 000006G                   MOV     R50BUF,FILNAM+6 ;STORE THE EXTENSION
011126  000241                            4$:     CLC                     ;SIGNAL SUCCESS ON RETURN
011130  000207                                    RETURN
                                          ;
                                          ;  ERROR -- INVALID FILE NAME
011132                                    3$:     FERR    #BDFNAM
011146  000261                                    SEC                     ;SIGNAL ERROR ON RETURN
011150  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.
                                ;
011152  010146                  ACRFIL: MOV     R1,-(SP)
011154  010246                          MOV     R2,-(SP)
011156  010446                          MOV     R4,-(SP)
011160  010546                          MOV     R5,-(SP)
                                ;
                                ;  Skip over leading spaces in front of the file spec
                                ;
011162  122327  000040          6$:     CMPB    (R3)+,#'        ;SKIP LEADING SPACES
011166  001775                          BEQ     6$
011170  005303                          DEC     R3              ;POINT TO 1ST NON-BLANK CHAR
                                ;
                                ;  Move file spec to a holding buffer
                                ;
011172  012702  000000G                 MOV     #BLKO,R2        ;POINT TO HOLDING BUFFER
011176  112300                  1$:     MOVB    (R3)+,R0        ;GET NEXT CHAR FROM FILE NAME
011200  001416                          BEQ     2$              ;BR IF HIT END
011202  120027  000057                  CMPB    R0,#'/          ;TERMINATE ON SLASH
011206  001413                          BEQ     2$
011210  120027  000054                  CMPB    R0,#',          ;TERMINATE ON COMMA
011214  001410                          BEQ     2$
011216  120027  000040                  CMPB    R0,#'           ;TERMINATE ON BLANK
011222  001405                          BEQ     2$
011224  120027  000075                  CMPB    R0,#'=          ;TERMINATE ON EQUAL SIGN
011230  001402                          BEQ     2$
011232  110022                          MOVB    R0,(R2)+        ;MOVE CHAR TO HOLDING BUFFER
011234  000760                          BR      1$              ;GO GET NEXT CHAR
                                ;
                                ;  Set up the file-spec as an input or output file.
                                ;
011236  005705                  2$:     TST     R5              ;INPUT OR OUTPUT TYPE FILE?
011240  001407                          BEQ     4$              ;BR IF INPUT FILE
011242  112722  000075                  MOVB    #'=,(R2)+       ;MAKE SPEC LOOK LIKE OUTPUT FILE
011246  162704  000002                  SUB     #2,R4           ;CORRECT DEFAULT EXTENSION POINTER FOR OUTPUT
011252  012705  000000G                 MOV     #KCSIBF,R5      ;CSISPC WILL PUT RESULT HERE
011256  000402                          BR      5$
011260  012705  000036G         4$:     MOV     #KCSIBF+30.,R5  ;CSISPC WILL PUT RESULT HERE
011264  105012                  5$:     CLRB    (R2)            ;PUT IN ASCIZ NULL AT END OF NAME
011266  005303                          DEC     R3              ;POINT BACK TO DELIMITER
                                ;
                                ;  Use .CSISPC to parse the file spec
                                ;
```

```
 011270  010601                          MOV     SP,R1           ;SAVE SP ACROSS .CSISPC
 011272                                  .CSISPC #KCSIBF,R4,#BLKO ;PARSE THE FILE SPEC
 011306  010106                          MOV     R1,SP           ;RESTORE SP (IGNORE SWITCH INFO FROM .CSISPC)
 011310  103410                          BCS     9$              ;BR IF INVALID
                                 ;
                                 ;  Move file spec to result area
                                 ;
 011312  012700  000000G                 MOV     #FILNAM,R0      ;POINT TO RESULT AREA
 011316  012520                          MOV     (R5)+,(R0)+
 011320  012520                          MOV     (R5)+,(R0)+
 011322  012520                          MOV     (R5)+,(R0)+
 011324  012520                          MOV     (R5)+,(R0)+
 011326  011510                          MOV     (R5),(R0)
                                 ;
                                 ;  Finished
                                 ;
 011330  000241                          CLC                     ;SIGNAL SUCCESS ON RETURN
 011332  012605                  9$:     MOV     (SP)+,R5
 011334  012604                          MOV     (SP)+,R4
 011336  012602                          MOV     (SP)+,R2
 011340  012601                          MOV     (SP)+,R1
 011342  000207                          RETURN
```

```
                                        .SBTTL  DMTALL -- Dismount and deallocate all devices
                                ;----------------------------------------------------------------------
                                ;  Dismount and deallocate all devices that are mounted by the current job.
                                ;
 011344  010246                 DMTALL: MOV     R2,-(SP)
 011346  010346                         MOV     R3,-(SP)
 011350  010546                         MOV     R5,-(SP)
                                ;
                                ;  Deallocate all devices allocated by this job
                                ;
 011352  005067  0000000                CLR     ALCDEV          ;Say to deallocate all devices for this job
 011356  012700  0000000                MOV     #DLCEMT,R0      ;Point to EMT argument block
 011362  104375                         EMT     375             ;Deallocate all devices for this job
                                ;
                                ;  Search through mount table looking for devices mounted by our job
                                ;
 011364  016705  0000000                MOV     CSHDEV,R5       ;Point to table of mounted devices
 011370  010500                 1$:     MOV     R5,R0           ;Get address of mount entry
 011372  004767  001070                 CALL    CDGET           ;Read mount entry into CDBUF
 011376  005767  000000C                TST     CDBUF+CD$DVU    ;Is this table entry in use?
 011402  001421                         BEQ     2$              ;Br if not
 011404  004767  000152                 CALL    CDJFLG          ;Get mount-flag for our job
 011410  130312                         BITB    R3,(R2)         ;Is this device mounted by us?
 011412  001415                         BEQ     2$              ;Br if not
                                ;
                                ;  Found a device that is mounted by us,
                                ;  reset mount flag for our job and see if device is mounted
                                ;  by any other jobs.
                                ;
 011414  140312                         BICB    R3,(R2)         ;Reset mount flag for our job
 011416  010500                         MOV     R5,R0           ;Get address where block is to be stored
 011420  004767  001062                 CALL    CDPUT           ;Write updated block back into kernel data
 011424  012703  000000C                MOV     #CDBUF+CD$JOB,R3;Get address of mount-flag table
 011430  012702  0000000                MOV     #CD$$UB,R2      ;Get # bytes in mount-flag table
 011434  105723                 3$:     TSTB    (R3)+           ;Any other jobs using this device?
 011436  001003                         BNE     2$              ;Br if yes
 011440  077203                         SOB     R2,3$
                                ;
                                ;  No other jobs have this device mounted.
                                ;  Free the mount table entry for this device and remove
                                ;  any file entries from cache.
                                ;
 011442  004767  000022                 CALL    DMTSUB          ;Dismount this device and all of its files
                                ;
                                ;  Check next entry in mount table
                                ;
 011446  062705  0000000        2$:     ADD     #CD$$SZ,R5      ;Point to next entry in mount table
 011452  020567  0000000                CMP     R5,CSHDVN       ;Any more entries?
 011456  103744                         BLO     1$              ;Loop if yes
                                ;
                                ;  Finished
                                ;
 011460  012605                         MOV     (SP)+,R5
 011462  012603                         MOV     (SP)+,R3
 011464  012602                         MOV     (SP)+,R2
 011466  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.
                                  ;
 011470  010346                   DMTSUB: MOV     R3,-(SP)
                                  ;
                                  ;  See if this device is a logical disk mounted by us
                                  ;
 011472  016700  000000G                  MOV     R50LD0,R0        ;Get LD0 as initial device name
 011476  005003                           CLR     R3               ;Init LD table index
 011500  026763  000000C 000000G  1$:     CMP     CDBUF+CD$DVU,LDPDEV(R3) ;Is this LD on same physical device?
 011506  001004                           BNE     2$               ;Br if not
 011510  026763  000000C 000000G          CMP     CDBUF+CD$BAS,LDBASE(R3) ;Same starting block numbers?
 011516  001412                           BEQ     3$               ;Br if yes -- Found logical disk that matches
 011520  005200                   2$:     INC     R0               ;Advance logical disk name
 011522  062703  000002                   ADD     #2,R3            ;Advance LD table index
 011526  020327  000016                   CMP     R3,#14.          ;Checked all logical disks?
 011532  101762                           BLOS    1$               ;Br if not
                                  ;
                                  ;  This is not a logical disk.
                                  ;  Get physical device name.
                                  ;
 011534  016700  000000C                  MOV     CDBUF+CD$DVU,R0 ;Get physical device and unit number
 011540  004767  001666                   CALL    CVDVNM           ;Convert to device name
                                  ;
                                  ;  Do dismount EMT
                                  ;
 011544  010067  000000G          3$:     MOV     R0,MNTDEV        ;Set name of device to dismount
 011550  012700  000000G                  MOV     #DMTARG,R0       ;Point to EMT argument block
 011554  104375                           EMT     375              ;Dismount the device
                                  ;
                                  ;  Finished
                                  ;
 011556  012603                           MOV     (SP)+,R3
 011560  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.
                                 ;
 011562  116703  000000G         CDJFLG: MOVB    CORUSR,R3          ;Get current job index number
 011566  006203                          ASR     R3                 ;Convert to index by 1
 011570  005303                          DEC     R3                 ;Make base job number 0
 011572  005002                          CLR     R2                 ;Clear for divide
 011574  071227  000010                  DIV     #8.,R2             ;Divide by 8 jobs per byte
 011600  062702  000000C                 ADD     #CDBUF+CD$JOB,R2;Get address of byte within entry in CDBUF
 011604  012700  000001                  MOV     #1,R0              ;Get a mount flag
 011610  072003                          ASH     R3,R0              ;Position flag according to job number
 011612  010003                          MOV     R0,R3              ;Return flag in R3
 011614  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.
                        ;
011616  010146          CHKDEV: MOV     R1,-(SP)
011620  010246                  MOV     R2,-(SP)
                        ;
                        ;  If this name has been assigned, substitute physical device name for
                        ;  logical device name.
                        ;
011622  010501                  MOV     R5,R1           ;GET LOGICAL DEVICE NAME
011624  010500                  MOV     R5,R0
011626  004767  001740          CALL    ASNSRC          ;SEE IF DEVICE NAME HAS BEEN ASSIGNED
011632  103402                  BCS     11$             ;BR IF NOT ASSIGNED
011634  016201  000004          MOV     4(R2),R1        ;REPLACE LOGICAL DEVICE NAME WITH PHYSICAL
                        ;
                        ;  Get device name and split off unit number.
                        ;
011640  005000          11$:    CLR     R0              ;SET FOR DIVIDE
011642  071027  000050          DIV     #50,R0          ;SPLIT OFF LOW-ORDER RAD50 CHARACTER
011646  012702  177777          MOV     #-1,R2          ;ASSUME NO UNIT NUMBER SPECIFIED
011652  005701                  TST     R1              ;WAS A UNIT NUMBER SPECIFIED?
011654  001406                  BEQ     6$              ;BR IF NOT
011656  162701  000036          SUB     #36,R1          ;CONVERT RAD50 DIGIT TO BINARY VALUE
011662  010102                  MOV     R1,R2           ;GET BINARY VALUE OF UNIT NUMBER
011664  020227  000007          CMP     R2,#7           ;RESTRICT UNIT NUMBER TO RANGE 0-7
011670  101027                  BHI     8$              ;BR IF INVALID UNIT NUMBER
011672  070027  000050  6$:     MUL     #50,R0          ;GET DEVICE NAME WITHOUT UNIT NUMBER
                        ;
                        ;  The rad50 device name less unit number is now in R1.
                        ;  R2 has the binary value of the unit number or -1 if no unit number
                        ;  was specified.
                        ;
                        ;  Translate "SY:" and "DK:" to physical device.
                        ;
011676  020167  000000G         CMP     R1,R50DK        ;IS DEVICE NAME "DK"?
011702  001403                  BEQ     2$              ;BR IF YES
011704  020167  000000G         CMP     R1,R50SY        ;IS DEVICE NAME "SY"?
011710  001007                  BNE     3$              ;BR IF NOT
011712  016704  000000G 2$:     MOV     SYINDX,R4       ;GET SY DEVICE INDEX NUMBER
011716  005702                  TST     R2              ;DID USER SPECIFY A UNIT NUMBER?
011720  002015                  BGE     7$              ;BR IF YES
011722  116702  000001G         MOVB    SYUNIT+1,R2     ;GET SYSTEM DEVICE UNIT NUMBER
011726  000412                  BR      7$
                        ;
                        ;  Look up device name in permanent device name table.
                        ;
011730  016704  000000G 3$:     MOV     NUMDEV,R4       ;GET INDEX NUMBER OF LAST DEVICE
```

```
 011734  020164  0000000         5$:     CMP     R1,PNAME(R4)    ;SEARCH FOR NAME IN TABLE
 011740  001405                          BEQ     7$              ;BR IF FOUND
 011742  162704  000002                  SUB     #2,R4           ;TRY NEXT ENTRY
 011746  002372                          BGE     5$              ;LOOP IF MORE TO CHECK
                                 ;
                                 ;  Invalid device name
                                 ;
 011750  000261                  8$:     SEC                     ;SIGNAL INVALID DEVICE NAME
 011752  000414                          BR      10$             ;RETURN
                                 ;
                                 ;  Found device name.  Translate no unit number into # 0.
                                 ;
 011754  010200                  7$:     MOV     R2,R0           ;GET UNIT NUMBER VALUE
 011756  002001                          BGE     1$              ;BR IF UNIT NUMBER WAS SPECIFIED
 011760  005000                          CLR     R0              ;SAY UNIT # = 0 IF NONE SPECIFIED
                                 ;
                                 ;  If the device is a logical disk (LDn), check to make sure the
                                 ;  particular unit is mapped to a file
                                 ;
 011762  020467  0000000         1$:     CMP     R4,LDDEVX       ;IS THIS A LOGICAL DISK?
 011766  001005                          BNE     9$              ;BR IF NOT
 011770  010002                          MOV     R0,R2           ;GET UNIT NUMBER
 011772  006302                          ASL     R2              ;CONVERT TO WORD TABLE INDEX
 011774  005762  0000000                 TST     LDPDEV(R2)      ;IS UNIT MAPPED TO A FILE?
 012000  001763                          BEQ     8$              ;BR IF NOT
 012002  000241                  9$:     CLC                     ;SIGNAL GOOD RETURN
                                 ;
                                 ;  Finished -- Return
                                 ;
 012004  012602                  10$:    MOV     (SP)+,R2
 012006  012601                          MOV     (SP)+,R1
 012010  000207                          RETURN
```

```
                                        .SBTTL  CHKMNT -- See if device is mounted
                                ;------------------------------------------------------------------
                                ;  CHKMNT is called to determine if a specified device is mounted
                                ;  by any users.
                                ;
                                ;  Inputs:
                                ;   R5 = Rad50 device-unit name.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Device is mounted.
                                ;   C-flag set ==> Device is not mounted.
                                ;   CDBUF contains mount table entry for device.
                                ;
 012012  010346                 CHKMNT: MOV     R3,-(SP)
 012014  010446                         MOV     R4,-(SP)
 012016  010546                         MOV     R5,-(SP)
                                ;
                                ;  Convert device name into device index number and unit number
                                ;
 012020  004767  177572                 CALL    CHKDEV          ;Convert device name to device # and unit #
 012024  103437                         BCS     9$              ;Br if invalid device name
                                ;
                                ;  If this device is a logical disk, get base block # and physical dev info
                                ;
 012026  020467  000000G                CMP     R4,LDDEVX       ;Is this a logical disk?
 012032  001006                         BNE     1$              ;Br if not
 012034  006300                         ASL     R0              ;Convert unit # to word table index
 012036  016004  000000G                MOV     LDPDEV(R0),R4   ;Get physical disk device # and unit #
 012042  016003  000000G                MOV     LDBASE(R0),R3   ;Get base block # of logical disk file
 012046  000403                         BR      2$
 012050  000300                 1$:     SWAB    R0              ;Put unit # in high-order byte
 012052  050004                         BIS     R0,R4           ;Combine unit # and device #
 012054  005003                         CLR     R3              ;Base block # = 0 if not logical disk
                                ;
                                ;  Search mount table for this device
                                ;
 012056  016705  000000G        2$:     MOV     CSHDEV,R5       ;Point to start of mount table
 012062  010500                 3$:     MOV     R5,R0           ;Get addr of mount table entry
 012064  004767  000376                 CALL    CDGET           ;Copy mount table entry into CDBUF
 012070  020467  000000C                CMP     R4,CDBUF+CD$DVU ;Is entry for this device?
 012074  001003                         BNE     4$              ;Br if not
 012076  020367  000000C                CMP     R3,CDBUF+CD$BAS ;Check base block numbers too
 012102  001407                         BEQ     5$              ;Br if found entry for this device
 012104  062705  000000G        4$:     ADD     #CD$$SZ,R5      ;Point to next mount table entry
 012110  020567  000000G                CMP     R5,CSHDVN       ;Hit end of table?
 012114  103762                         BLO     3$              ;Loop if not
 012116  000261                 6$:     SEC                     ;Device is not mounted at all
 012120  000401                         BR      9$
                                ;
                                ;  We found entry for this device in mount table.
                                ;
 012122  000241                 5$:     CLC                     ;Signal that device is mounted
                                ;
                                ;  Finished
                                ;
 012124  012605                 9$:     MOV     (SP)+,R5
 012126  012604                         MOV     (SP)+,R4
```

```
     58 012130  012603                         MOV     (SP)+,R3
     59 012132  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.
                               ;
012134  010246                 CHKMTX: MOV     R2,-(SP)
012136  010346                         MOV     R3,-(SP)
012140  010446                         MOV     R4,-(SP)
012142  010546                         MOV     R5,-(SP)
                               ;
                               ;  Get mount flag for current job
                               ;
012144  004767  177412                 CALL    CDJFLG          ;Get mount flag for our job
012150  111204                         MOVB    (R2),R4         ;Save mount flags for byte with our mount flag
012152  140312                         BICB    R3,(R2)         ;Clear mount flag for our job
                               ;
                               ;  See if any other jobs have this device mounted
                               ;
012154  012705  000000C                MOV     #CDBUF+CD$JOB,R5;Point to table with mount flags
012160  012700  000000G                MOV     #CD$$UB,R0      ;Get # bytes in table
012164  105725                 7$:     TSTB    (R5)+           ;Any other mount flags set?
012166  001004                         BNE     8$              ;Br if yes
012170  077003                         SOB     R0,7$
012172  110412                         MOVB    R4,(R2)         ;Reset mount flag for our job
012174  000241                 6$:     CLC                     ;Signal that device is not mounted by others
012176  000402                         BR      9$
012200  110412                 8$:     MOVB    R4,(R2)         ;Reset mount flag for our job
012202  000261                         SEC                     ;Signal that device is mounted by others
                               ;
                               ;  Finished
                               ;
012204  012605                 9$:     MOV     (SP)+,R5
012206  012604                         MOV     (SP)+,R4
012210  012603                         MOV     (SP)+,R3
012212  012602                         MOV     (SP)+,R2
012214  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)
                                ;
 012216                         CKCLUS:
                                ;
                                ;  See if CL unit is in use by SET HOST
                                ;
 012216  005760  0000000                TST     CL$XLN(R0)      ;Any job using with SET HOST?
 012222  001404                         BEQ     1$              ;Br if not
 012224  016000  0000000                MOV     CL$XLN(R0),R0   ;Get number of job
 012230  006200                         ASR     R0              ;Convert index # to job #
 012232  000420                         BR      9$
                                ;
                                ;  See if any job has CL unit open or allocated
                                ;
 012234  006200                 1$:     ASR     R0              ;Convert to unit number
 012236  020027  000007                 CMP     R0,#7.          ;Is this a CL or C1 unit?
 012242  101405                         BLOS    2$              ;Br if CL
 012244  162700  000010                 SUB     #8.,R0          ;Remove C1 unit bias
 012250  066700  165644                 ADD     R50C10,R0       ;Form Rad50 device name
 012254  000402                         BR      3$
 012256  066700  165630         2$:     ADD     R50CL0,R0       ;Form rad50 device name
 012262  010067  0000000        3$:     MOV     R0,ALCDEV       ;Set device name for EMT
 012266  012700  0000000                MOV     #TALEMT,R0      ;Point to check-allocation EMT
 012272  104375                         EMT     375             ;See if this unit is allocated or in use
                                ;
                                ;  Finished
                                ;
 012274  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
                                ;
012276  010046                  CHKALC: MOV     R0,-(SP)
012300  010446                          MOV     R4,-(SP)
012302  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set name of device in EMT argument block
                                ;
012304  010067  0000000                 MOV     R0,ALCDEV       ;Set name of device for EMT
                                ;
                                ;  Don't do allocation test for LD device
                                ;
012310  010005                          MOV     R0,R5           ;Get name of device
012312  004767  177300                  CALL    CHKDEV          ;Convert to device index number
012316  120467  0000000                 CMPB    R4,LDDEVX       ;Is this a LD device?
012322  001455                          BEQ     9$              ;Br if yes -- allocation ok
                                ;
                                ;  Execute EMT that will test for allocation conflict
                                ;
012324  012700  0000000                 MOV     #TALEMT,R0      ;Point to EMT argument block
012330  104375                          EMT     375             ;Do allocation test
012332  010005                          MOV     R0,R5           ;Save # of any job that is using device
012334  103017                          BCC     15$             ;Br if allocation is ok
                                ;
                                ;  An error occurred on the test allocation.
                                ;  Print error message.
                                ;
012336  123727  0000000 000002          CMPB    @#ERRLOC,#2     ;Invalid device?
012344  001444                          BEQ     9$              ;Br if yes
012346  123727  0000000 000001          CMPB    @#ERRLOC,#1     ;Device already allocated by someone else?
012354  001007                          BNE     15$             ;Br if not
012356                                  FERR    #EM$DAA         ;Device allocated by another job
012372  000422                          BR      17$
                                ;
                                ;  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
                                ;
012374  010500                  15$:    MOV     R5,R0           ;Get # of job using device
012376  001427                          BEQ     9$              ;Br if no job using device
012400  120467  0000000                 CMPB    R4,CLDEVX       ;Is device a CL unit?
012404  001424                          BEQ     9$              ;Br if family member wants CL unit
012406  006300                          ASL     R0              ;Convert to job index number
012410  120067  0000000                 CMPB    R0,CORUSR       ;In use by our job?
012414  001420                          BEQ     9$              ;Br if yes
012416  005727  0000000                 TST     #KUSECK         ;Do we want to consider this as an error?
012422  001415                          BEQ     9$              ;Br if not
012424                                  FERR    #EM$DIU         ;Device is in use by another user
012440  004767  173476          17$:    CALL    PRTDEC          ;Print the number of the job that has the dev
012444                                  .PRINT  #CRLF           ;Terminate the print line
```

```
 012452  000167  000000G                JMP     RDCMD           ;Abort the command
                                ;
                                ;   We can access the device
                                ;
 012456  012605                 9$:     MOV     (SP)+,R5
 012460  012604                         MOV     (SP)+,R4
 012462  012600                         MOV     (SP)+,R0
 012464  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.
                                ;
 012466  010067  000000G        CDGET:  MOV     R0,CDGADR         ;Set address of block to get
 012472  012700  000000G                MOV     #CDGEMT,R0        ;Get address of EMT arg block
 012476  104375                         EMT     375               ;Get copy of block
 012500  012700  000000G                MOV     #CDBUF,R0         ;Return pointer to CDBUF in R0
 012504  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.
                                ;
 012506  010067  000000G        CDPUT:  MOV     R0,CDPADR         ;Set destination address
 012512  012700  000000G                MOV     #CDPEMT,R0        ;Point to EMT argument block
 012516  104375                         EMT     375               ;Store the block
 012520  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.
                                ;
012522  010246                  LDCLEN: MOV     R2,-(SP)
012524  010546                          MOV     R5,-(SP)
                                ;
                                ;  Perform LD CLEAN operation (reset logical disk information)
                                ;
012526  016705  000000G                 MOV     R50LDO,R5       ;GET NAME OF FIRST LOGICAL DISK (LDO)
012532  005002                          CLR     R2              ;GET INDEX TO 1ST LOGICAL DISK (LDO)
                                ;
                                ;  See if this logical disk is in use
                                ;
012534  010200                  1$:     MOV     R2,RO           ;GET LD INDEX NUMBER
012536  006300                          ASL     RO              ;* 4 WORDS PER ENTRY
012540  006300                          ASL     RO
012542  005760  000000G                 TST     LDNAME(RO)      ;IS THIS LOGICAL DISK IN USE?
012546  001423                          BEQ     2$              ;BR IF NOT
                                ;
                                ;  Dismount the logical disk
                                ;
012550  010567  000000G                 MOV     R5,MNTDEV       ;SET DEVICE NAME FOR DISMOUNT
012554  012700  000000G                 MOV     #DMTARG,RO      ;DISMOUNT THE DEVICE
012560  105267  000000G                 INCB    SERFLG          ;DO .SERR
012564  104375                          EMT     375
012566  105067  000000G                 CLRB    SERFLG          ;DO .HERR
                                ;
                                ;  Now reinitialize information about the logical disk
                                ;
012572  004767  000042                  CALL    LDMNT           ;CHECK IT
                                ;
                                ;  Now remount the logical disk
                                ;
012576  005762  000000G                 TST     LDPDEV(R2)      ;IS THE LOGICAL DISK THERE?
012602  001405                          BEQ     2$              ;BR IF NOT
012604  010567  000000G                 MOV     R5,MNTDEV       ;SET DEVICE NAME FOR THE MOUNT
012610  012700  000000G                 MOV     #MNTARG,RO      ;MOUNT THE DEVICE
012614  104375                          EMT     375
                                ;
                                ;  Check next logical disk
                                ;
012616  005205                  2$:     INC     R5              ;ADVANCE LDn NAME
012620  062702  000002                  ADD     #2,R2           ;ADVANCE LOGICAL DISK INDEX NUMBER
012624  020227  000016                  CMP     R2,#14.         ;HAVE WE CHECKED ALL LOGICAL DISKS?
012630  101741                          BLOS    1$              ;BR IF NOT
                                ;
                                ;  Finished
                                ;
012632  012605                          MOV     (SP)+,R5
012634  012602                          MOV     (SP)+,R2
012636  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.
                                  ;
012640  010446                    LDMNT:  MOV     R4,-(SP)
012642  010546                            MOV     R5,-(SP)
                                  ;
                                  ;  Remove any entry for this logical disk from access control table
                                  ;
012644  004767  000450                    CALL    DLLDAC          ;REMOVE LD ENTRY FROM ACCESS CONTROL TABLE
                                  ;
                                  ;  Do lookup on file
                                  ;
012650  010205                            MOV     R2,R5           ;GET UNIT #
012652  006305                            ASL     R5              ;*4 WORDS PER ENTRY
012654  006305                            ASL     R5
012656  062705  000000G                   ADD     #LDNAME,R5      ;POINT TO FILE SPEC IN LDNAME TABLE
012662  005715                            TST     (R5)            ;IS THERE A FILE SPEC FOR THIS DISK?
012664  001475                            BEQ     9$              ;BR IF NOT
012666  112767  000001  000000G           MOVB    #1,SERFLG       ;DO .SERR TO AVOID ABORT FOR ILLEGAL DEVICE
012674                                    .LOOKUP #XAREA,#1,R5    ;LOOKUP THE FILE
012712  112767  000000  000000G           MOVB    #0,SERFLG       ;DO .HERR (DON'T CLEAR C-FLAG)
012720  103457                            BCS     9$              ;BR IF CAN'T FIND THE FILE
012722  010062  000000G                   MOV     R0,LDSIZE(R2)   ;SAVE THE SIZE OF THE FILE
                                  ;
                                  ;  Do .SAVESTATUS to get information about the file
                                  ;
012726  012705  000000G                   MOV     #BLK0,R5        ;POINT TO AREA FOR SAVESTATUS DATA
012732                                    .SAVEST #XAREA,#1,R5    ;SAVE FILE STATUS
012750  016500  000000G                   MOV     C.CSW(R5),R0    ;GET CSW
012754  042700  177701                    BIC     #^C76,R0        ;EXTRACT DEVICE UNIT #
012760  110062  000000G                   MOVB    R0,LDPDEV(R2)   ;SAVE PHYSICAL DEVICE INDEX NUMBER
012764  116504  000000G                   MOVB    C.DEVQ(R5),R4   ;GET PHYSICAL UNIT NUMBER
012770  042704  177770                    BIC     #^C7,R4
012774  110462  000001G                   MOVB    R4,LDPDEV+1(R2) ;SET PHYS UNIT # FOR LOGICAL DISK
013000  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.
                                  ;
013006  032765  000000G 000000G           BIT     #CS$RON,C.CSW(R5);DO WE HAVE READ-ONLY ACCESS TO FILE?
013014  001403                            BEQ     2$              ;BR IF NOT
013016  052762  000000G 000000G           BIS     #LD$RON,LDFLAG(R2);SET NO-WRITE FLAG FOR THIS LD
                                  ;
                                  ;  See if the logical disk file is itself within a logical disk
                                  ;  (i.e., this is a nested logical disk)
                                  ;
013024  020067  000000G           2$:     CMP     R0,LDDEVX       ;IS FILE ON A LOGICAL DISK?
```

```
013030  001007                           BNE     1$              ;BR IF NOT
013032  006304                           ASL     R4              ;CVT UNIT # TO LOG DISK TABLE INDEX
013034  016462  000000G 000000G          MOV     LDPDEV(R4),LDPDEV(R2) ;GET REAL PHYSICAL DEVICE & UNIT #
013042  066462  000000G 000000G          ADD     LDBASE(R4),LDBASE(R2) ;BIAS BASE BLOCK # BY LOG DISK BASE
                                 ;
                                 ;  Make entry for the LD in the access control table.
                                 ;
013050  004767  000100           1$:     CALL    ADLDAC          ;MAKE ENTRY IN ACCESS CONTROL TABLE
                                 ;
                                 ;  Success
                                 ;
013054  000241                           CLC                     ;SIGNAL SUCCESS ON RETURN
013056  000403                           BR      10$
                                 ;
                                 ;  Error -- Could not find the file
                                 ;
013060  005062  000000G          9$:     CLR     LDPDEV(R2)      ;SAY LOGICAL DISK IS NOT ACTIVE
013064  000261                           SEC                     ;SIGNAL FAILURE ON RETURN
013066  012605                   10$:    MOV     (SP)+,R5
013070  012604                           MOV     (SP)+,R4
013072  000207                           RETURN
```

```
      1                                                .SBTTL  CKLDAC -- Check if LD is in access control table
      2                                         ;-----------------------------------------------------------------------
      3                                         ;  CKLDAC is called to determine if a certain logical disk is in the
      4                                         ;  device/file access control table.
      5                                         ;
      6                                         ;  Inputs:
      7                                         ;   R2 = Logical disk index number (2 * unit #)
      8                                         ;
      9                                         ;  Outputs:
     10                                         ;   C-flag cleared ==> Found logical disk entry in access control table.
     11                                         ;   C-flag set     ==> Logical disk entry is not in access table.
     12                                         ;   R0 = Pointer to access control entry.
     13                                         ;
     14 013074  010246                          CKLDAC: MOV     R2,-(SP)        ;SAVE ORIGINAL LD INDEX NUMBER
     15 013076  006202                                  ASR     R2              ;CONVERT INDEX # TO UNIT #
     16 013100  005767  000000G                         TST     RESDEV          ;ARE THERE ANY ENTRIES IN ACCESS TABLE?
     17 013104  001416                                  BEQ     4$              ;BR IF NOT
     18                                         ;
     19                                         ;  There are entries in the access control table.
     20                                         ;  Search for entry matching our logical disk.
     21                                         ;
     22 013106  012700  000000G                         MOV     #OKFILE,R0      ;POINT TO START OF ACCESS TABLE
     23 013112  126067  000000G 000000G 1$:             CMPB    OF$DEV(R0),LDDEVX ;IS THIS ENTRY FOR A LOGICAL DISK?
     24 013120  001003                                  BNE     2$              ;BR IF NOT
     25 013122  120260  000000G                         CMPB    R2,OF$UNT(R0)   ;IS ENTRY FOR SPECIFIED UNIT?
     26 013126  001407                                  BEQ     3$              ;BR IF YES -- FOUND ENTRY
     27 013130  062700  000000G         2$:             ADD     #OF$$SZ,R0      ;POINT TO NEXT ACCESS ENTRY
     28 013134  020027  000000G                         CMP     R0,#OKFEND      ;CHECKED ALL ENTRIES?
     29 013140  103764                                  BLO     1$              ;BR IF NOT
     30                                         ;
     31                                         ;  LD entry is not in access table
     32                                         ;
     33 013142  000261                          4$:     SEC                     ;SIGNAL FAILURE ON RETURN
     34 013144  000401                                  BR      9$
     35                                         ;
     36                                         ;  Found LD entry in table
     37                                         ;
     38 013146  000241                          3$:     CLC                     ;SIGNAL SUCCESS ON RETURN
     39 013150  012602                          9$:     MOV     (SP)+,R2        ;RECOVER LD INDEX NUMBER
     40 013152  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)
                                        ;
 013154  005767  000000G                ADLDAC: TST     RESDEV          ;ANY ENTRIES IN ACCESS TABLE?
 013160  001456                                 BEQ     9$              ;BR IF NOT
                                        ;
                                        ;  There are entries in the access control table
                                        ;
 013162  010246                                 MOV     R2,-(SP)
 013164  010546                                 MOV     R5,-(SP)
 013166  006202                                 ASR     R2              ;CONVERT LD INDEX # TO UNIT #
                                        ;
                                        ;  Find a free entry in the access table
                                        ;
 013170  012705  000000G                        MOV     #OKFILE,R5      ;POINT TO START OF ACCESS TABLE
 013174  020527  000000G                1$:     CMP     R5,#OKFEND      ;REACHED END OF TABLE?
 013200  103036                                 BHIS    3$              ;BR IF YES -- TABLE OVERFLOW
 013202  005765  000000G                        TST     OF$FIL(R5)      ;IS THIS ENTRY FREE?
 013206  001403                                 BEQ     2$              ;BR IF FREE
 013210  062705  000000G                        ADD     #OF$$SZ,R5      ;POINT TO NEXT ENTRY
 013214  000767                                 BR      1$              ;GO CHECK IT
                                        ;
                                        ;  We found a free entry.  Add entry for LD.
                                        ;
 013216  116765  000000G 000000G        2$:     MOVB    LDDEVX,OF$DEV(R5);SET LOGICAL DISK DEVICE INDEX NUMBER
 013224  110265  000000G                        MOVB    R2,OF$UNT(R5)   ;SET LOGICAL DISK UNIT #
 013230  012700  000000G                        MOV     #WLDNAM,R0      ;SET FILE NAME TO WILDCARDS
 013234  010065  000000G                        MOV     R0,OF$FIL(R5)
 013240  010065  000002G                        MOV     R0,OF$FIL+2(R5)
 013244  010065  000004G                        MOV     R0,OF$FIL+4(R5)
 013250  105065  000000G                        CLRB    OF$FLG(R5)      ;INITIALLY CLEAR ALL CONTROL FLAGS
 013254  006302                                 ASL     R2              ;CVT UNIT # TO LD INDEX #
 013256  032762  000000G 000000G                BIT     #LD$RON,LDFLAG(R2);IS LOGICAL DISK WRITE PROTECTED?
 013264  001412                                 BEQ     4$              ;BR IF NOT
 013266  152765  000000G 000000G                BISB    #OT$RON,OF$FLG(R5);SET READ-ONLY FLAG IN ACCESS TABLE
 013274  000406                                 BR      4$
                                        ;
                                        ;  Error -- Access table overflow
                                        ;
 013276                                 3$:     FERR    #TBLOVF         ;TABLE OVERFLOW
                                        ;
                                        ;  Finished
                                        ;
 013312  012605                         4$:     MOV     (SP)+,R5
 013314  012602                                 MOV     (SP)+,R2
 013316  000207                         9$:     RETURN
```

```
                                .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)
                        ;
013320  004767  177550  DLLDAC: CALL    CKLDAC          ;IS THERE AN ENTRY FOR THIS LD IN TABLE?
013324  103406                  BCS     1$              ;BR IF NOT
                        ;
                        ;  We have found an entry in the access table for this LD.
                        ;  R0 = Pointer to the entry.
                        ;
013326  105060  000000G         CLRB    OF$DEV(R0)      ;MARK ENTRY AS FREE
013332  105060  000000G         CLRB    OF$UNT(R0)
013336  005060  000000G         CLR     OF$FIL(R0)
                        ;
                        ;  Finished
                        ;
013342  000207          1$:     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.
                                ;
013344  010246                  DOASGN: MOV     R2,-(SP)
                                ;
                                ;  Determine if the "physical" device name is actually a logical name
                                ;
013346  004767  000220          1$:     CALL    ASNSRC          ;SEE IF THIS IS ACTUALLY A LOGICAL NAME
013352  103402                          BCS     2$              ;BR IF NOT
013354  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
                                ;
013360  010046                  2$:     MOV     R0,-(SP)        ;SAVE PHYSICAL DEVICE NAME
013362  010500                          MOV     R5,R0           ;GET LOGICAL NAME
013364  004767  000202                  CALL    ASNSRC          ;LOOK IT UP
013370  103010                          BCC     3$              ;BR IF FOUND ENTRY -- REUSE THE ENTRY
                                ;
                                ;  Add a new entry to the assign table
                                ;
013372  005000                          CLR     R0              ;SEARCH FOR A FREE ENTRY IN THE ASSIGN TABLE
013374  004767  000172                  CALL    ASNSRC          ;LOOK FOR FREE ENTRY
013400  103004                          BCC     3$              ;BR IF FOUND ONE
013402                                  FABORT  #ASNOVF         ;ASSIGN TABLE OVERFLOW
                                ;
                                ;  Move information into assign table
                                ;
013412  010522                  3$:     MOV     R5,(R2)+        ;LOGICAL NAME
013414  005022                          CLR     (R2)+           ;NO FILE SIZE
013416  012622                          MOV     (SP)+,(R2)+     ;PHYSICAL DEVICE NAME
013420  005022                          CLR     (R2)+           ;NO FILE NAME
013422  005022                          CLR     (R2)+
013424  005022                          CLR     (R2)+           ;NO EXTENSION
                                ;
                                ;  Finished
                                ;
013426  012602                          MOV     (SP)+,R2
013430  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                                 ;   RO = Device number (low-order byte), unit number (high-order byte)
      8                                 ;
      9                                 ;  Outputs:
     10                                 ;   RO = RAD50 device name.
     11                                 ;
     12 013432  010346                  CVDVNM: MOV     R3,-(SP)
     13 013434  010003                          MOV     RO,R3           ;Copy device # and unit #
     14 013436  000303                          SWAB    R3              ;Put unit # in low-order byte
     15 013440  042703  177770                  BIC     #^C7,R3         ;Clear all but unit number in R3
     16 013444  042700  177400                  BIC     #^C377,RO       ;Clear all but device number in RO
     17 013450  016000  000000G                 MOV     PNAME(RO),RO    ;Get base device name
     18 013454  062703  000036                  ADD     #30.,R3         ;Convert unit number to RAD50 character
     19 013460  060300                          ADD     R3,RO           ;Combine unit number with device name
     20 013462  012603                          MOV     (SP)+,R3
     21 013464  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)
                                        ;
 013466                                 CHKCLU:
                                        ;
                                        ;  See if this is a CL unit
                                        ;
 013466  020067  164416                         CMP     R0,R50CL        ;Is name "CL"?
 013472  001002                                 BNE     1$              ;Br if not
 013474  005000                                 CLR     R0              ;Translate to unit 0
 013476  000431                                 BR      7$
 013500  020067  164406                 1$:     CMP     R0,R50CL0       ;Is unit in the range CL0 to CL7?
 013504  103406                                 BLO     2$              ;Br if not
 013506  020067  164402                         CMP     R0,R50CL7
 013512  101003                                 BHI     2$              ;Br if not
 013514  166700  164372                         SUB     R50CL0,R0       ;Get unit number
 013520  000420                                 BR      7$
                                        ;
                                        ;  See if this is a C1 unit
                                        ;
 013522  020067  164370                 2$:     CMP     R0,R50C1        ;Is unit name "C1"?
 013526  001003                                 BNE     3$              ;Br if not
 013530  012700  000010                         MOV     #8.,R0          ;C1 = unit 8
 013534  000412                                 BR      7$
 013536  020067  164356                 3$:     CMP     R0,R50C10       ;Is unit in the range C10 to C17?
 013542  103411                                 BLO     8$              ;Br if not
 013544  020067  164352                         CMP     R0,R50C17
 013550  101006                                 BHI     8$              ;Br if not
 013552  166700  164342                         SUB     R50C10,R0       ;Get C1 unit number
 013556  062700  000010                         ADD     #8.,R0          ;Add C1 unit bias
                                        ;
                                        ;  This is a CL or C1 unit
                                        ;
 013562  000241                         7$:     CLC                     ;Signal success on return
 013564  000401                                 BR      9$
                                        ;
                                        ;  This is not a CL or C1 unit
                                        ;
 013566  000261                         8$:     SEC                     ;Signal failure on return
                                        ;
                                        ;  Finished
                                        ;
 013570  000207                         9$:     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.
                                  ;
013572  010146                    ASNSRC: MOV     R1,-(SP)
013574  012701  000000G                   MOV     #MAXASN,R1      ;GET # ASSIGN BLOCKS
013600  012702  000000G                   MOV     #ASNTBL,R2      ;POINT TO ASSIGN TABLE
013604  020062  000000G           1$:     CMP     R0,AT$LOG(R2)   ;COMPARE LOGICAL NAMES
013610  001405                            BEQ     2$              ;BR IF WE FOUND BLOCK WE ARE LOOKING FOR
013612  062702  000000G                   ADD     #AT$$SZ,R2      ;POINT TO NEXT ASSIGN BLOCK
013616  077106                            SOB     R1,1$           ;LOOP IF MORE BLOCKS TO CHECK
013620  000261                            SEC                     ;SIGNAL FAILURE
013622  000401                            BR      3$
013624  000241                    2$:     CLC                     ;SIGNAL SUCCESS
013626  012601                    3$:     MOV     (SP)+,R1
013630  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.
                                ;
013632  010246                  LOGASN: MOV     R2,-(SP)
013634  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if device name is in our assign table
                                ;
013636  011500                          MOV     (R5),R0         ;Get logical device name
013640  004767  177726                  CALL    ASNSRC          ;See if name is in assign table
013644  103421                          BCS     1$              ;Br if name is not in assign table
                                ;
                                ;  Found logical device name in the assign table.
                                ;  Translate to physical device.
                                ;
013646  010504                          MOV     R5,R4           ;Get pointer to file spec buffer
013650  016224  000000G                 MOV     AT$DEV(R2),(R4)+;Put in physical device name
013654  016200  000000G                 MOV     AT$FIL(R2),R0   ;Was file name assigned?
013660  001413                          BEQ     1$              ;Br if not
013662  010024                          MOV     R0,(R4)+        ;Translate file name
013664  016224  000002G                 MOV     AT$FIL+2(R2),(R4)+
013670  016200  000000G                 MOV     AT$EXT(R2),R0   ;Was file extension specified?
013674  001405                          BEQ     1$              ;Br if not
013676  010024                          MOV     R0,(R4)+        ;Translate file extension
013700  016200  000000G                 MOV     AT$SIZ(R2),R0   ;Was file size specified?
013704  001401                          BEQ     1$              ;Br if not
013706  010014                          MOV     R0,(R4)         ;Translate file size
                                ;
                                ;  Translate "DK" and "SY" to physical device names
                                ;
013710  021567  000000G         1$:     CMP     (R5),R50SY      ;Is device name "SY"?
013714  001403                          BEQ     2$              ;Br if yes
013716  021567  000000G                 CMP     (R5),R50DK      ;Is device name "DK"?
013722  001002                          BNE     3$              ;Br if not
013724  016715  000000G         2$:     MOV     SYNAME,(R5)     ;Translate to physical device
                                ;
                                ;  Finished
                                ;
013730  012604                  3$:     MOV     (SP)+,R4
013732  012602                          MOV     (SP)+,R2
013734  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
                                ;
 013736  010346                 FORCE0: MOV     R3,-(SP)
 013740  010446                         MOV     R4,-(SP)
 013742  010546                         MOV     R5,-(SP)
 013744  011305                         MOV     (R3),R5         ;MOVE CURRENT DEV NAME TO R5
 013746  005004                         CLR     R4              ;SET UP FOR DIVIDE
 013750  071427  000050                 DIV     #50,R4          ;SEPARATE INTO NAME AND UNIT
 013754  005705                         TST     R5              ;WAS 3RD CHAR BLANK?
 013756  001012                         BNE     9$              ;RETURN IF NOT
 013760  010405                         MOV     R4,R5           ;GET HIGH 2 CHARS
 013762  005004                         CLR     R4              ;SET UP FOR ANOTHER DIVIDE
 013764  071427  000050                 DIV     #50,R4          ;SEPARATE 1 & 2 CHARS
 013770  005704                         TST     R4              ;WAS CHAR 1 BLANK?
 013772  001404                         BEQ     9$              ;EMPTY OR INVALID DEV NAME!
 013774  005705                         TST     R5              ;WAS CHAR 2 BLANK?
 013776  001402                         BEQ     9$              ;1-CHAR DEV NAME SHOULD BE INVALID???
 014000  062713  000036                 ADD     #^R  0,(R3)     ;FORCE TO UNIT 0
 014004  012605                 9$:     MOV     (SP)+,R5
 014006  012604                         MOV     (SP)+,R4
 014010  012603                         MOV     (SP)+,R3
 014012  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.
                                ;
014014  010246                  DEADEV: MOV     R2,-(SP)
014016  010346                          MOV     R3,-(SP)
014020  012702  000000G                 MOV     #ASNTBL,R2      ;Point to assign table
014024  012703  000000G                 MOV     #MAXASN,R3      ;Get # assign table entries
014030  020062  000000G         1$:     CMP     R0,AT$DEV(R2)   ;Is this entry for specified phys device?
014034  001004                          BNE     2$              ;Br if not
014036  005062  000000G                 CLR     AT$LOG(R2)      ;Clear logical device name
014042  005062  000000G                 CLR     AT$DEV(R2)      ;Clear physical device name
014046  062702  000000G         2$:     ADD     #AT$$SZ,R2      ;Point to next assign entry
014052  077312                          SOB     R3,1$           ;Loop if more to check
014054  012603                          MOV     (SP)+,R3
014056  012602                          MOV     (SP)+,R2
014060  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.
                                   ;
014062  010246                     INSSRC: MOV     R2,-(SP)
014064  010346                             MOV     R3,-(SP)
014066  010446                             MOV     R4,-(SP)
014070  010546                             MOV     R5,-(SP)
                                   ;
                                   ;  Copy file specification to INSSPC and perform any assigns
                                   ;
014072  012702  000134'                    MOV     #INSSPC,R2      ;Point to result buffer
014076  012703  000004                     MOV     #4,R3           ;Get # words to move
014102  012022                     10$:    MOV     (R0)+,(R2)+     ;Copy the file spec
014104  077302                             SOB     R3,10$
014106  012705  000134'                    MOV     #INSSPC,R5      ;Point to name
014112  004767  177514                     CALL    LOGASN          ;Perform full assignment
                                   ;
                                   ;  Check next entry in INSTALL table
                                   ;
014116  016705  000000G                    MOV     INSTBL,R5       ;Point to 1st entry in install table
014122  010567  000000G            1$:     MOV     R5,INGADR       ;Set address of entry to get
014126  012700  000000G                    MOV     #INGEMT,R0      ;Point to EMT arg block
014132  104375                             EMT     375             ;Get the install entry
014134  012702  000000C                    MOV     #IIBUF+II$NAM,R2;Point to entry we just got
014140  012703  000134'                    MOV     #INSSPC,R3      ;Point to target name
014144  012700  000004                     MOV     #4,R0           ;# words to compare
014150  021227  000000G            3$:     CMP     (R2),#WLDNAM    ;Wildcard in install entry?
014154  001402                             BEQ     7$              ;Br if yes
014156  021213                             CMP     (R2),(R3)       ;Compare file specs
014160  001003                             BNE     2$              ;Br if they don't match
014162  022223                     7$:     CMP     (R2)+,(R3)+     ;Advance both pointers
014164  077007                             SOB     R0,3$           ;Loop if more to compare
014166  000407                             BR      4$              ;We found the entry
014170  062705  000000G            2$:     ADD     #II$$SZ,R5      ;Point to next install entry
014174  020567  000000G                    CMP     R5,INSTBN       ;Checked all entries?
014200  103750                             BLO     1$              ;Loop if more to check
                                   ;
                                   ;  Cannot find entry for this program
                                   ;
014202  000261                     6$:     SEC                     ;Signal failure on return
014204  000415                             BR      9$
                                   ;
                                   ;  Found entry for program
                                   ;
014206  026727  000000C 000000G    4$:     CMP     IIBUF+II$NAM,#WLDNAM ;Is install device wild ("*")?
014214  001410                             BEQ     8$              ;Br if yes
014216  016705  000000G                    MOV     RUNDEV,R5       ;Get device spec for program
```

```
 014222  004767  175370                 CALL    CHKDEV          ;Convert device name to dev index number
 014226  103403                         BCS     8$              ;Br if invalid device
 014230  020467  000000G                CMP     R4,LDDEVX       ;Is program on a logical disk?
 014234  001762                         BEQ     6$              ;Br if yes -- Cannot be installed
 014236  000241                 8$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
 014240  012605                 9$:     MOV     (SP)+,R5
 014242  012604                         MOV     (SP)+,R4
 014244  012603                         MOV     (SP)+,R3
 014246  012602                         MOV     (SP)+,R2
 014250  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.
                                    ;
 014252  010146                     LSTSPL: MOV     R1,-(SP)
 014254  010246                             MOV     R2,-(SP)
 014256  010346                             MOV     R3,-(SP)
 014260  010446                             MOV     R4,-(SP)
 014262  010546                             MOV     R5,-(SP)
 014264  016102  000000G                    MOV     SDFHD(R1),R2     ;GET ADDRESS OF FIRST SFCB FOR THIS DEVICE
 014270  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)
                                    ;
 014272  010267  163546             1$:     MOV     R2,PEKADR        ;Set address of block in kernel
 014276  012767  000000G 163542             MOV     #SFCBSZ,PEKSIZ   ;Set size of data to get
 014304  012700  000040'                    MOV     #PEKEMT,RO       ;Point to EMT argument block
 014310  104375                             EMT     375              ;Move SFCB from kernel to our buffer
 014312  012702  000000G                    MOV     #BLKO,R2         ;Point to buffer with SFCB we got
                                    ;
                                    ;  See if 1st write has been done to this spool file.
                                    ;
 014316  132762  000000G 000000G            BITB    #SF$1ST,SFFLAG(R2);HAS 1ST WRITE BEEN DONE?
 014324  001504                             BEQ     5$               ;BR IF NOT
                                    ;
                                    ;  Print the file ID number
                                    ;
 014326  016205  000000G                    MOV     SFID(R2),R5      ;Get spool file ID number
 014332  012703  000004                     MOV     #4.,R3           ;Print 4 digits
 014336  004767  171702                     CALL    PRTFIX           ;Print ID value
 014342                                     .PRINT  #SPACE2          ;Print 2 spaces
                                    ;
                                    ;  Print device name.
                                    ;
 014350  016100  000000G                    MOV     SDNAME(R1),RO    ;GET NAME OF SPOOLED DEVICE (RAD50)
 014354  004767  171204                     CALL    PRTR50           ;PRINT THE DEVICE NAME
                                    ;
                                    ;  Print a star if this file is being printed now.
                                    ;
 014360  012700  000040                     MOV     #' ,RO           ;ASSUME FILE NOT BEING PRINTED
 014364  132762  000000G 000000G            BITB    #SF$BSY,SFFLAG(R2);IS FILE BEING PRINTED NOW?
 014372  001402                             BEQ     4$               ;BR IF NOT
 014374  012700  000052                     MOV     #'*,RO           ;PRINT *
 014400                             4$:     .TTYOUT
 014404                                     .PRINT  #SPACE2
                                    ;
                                    ;  Print user number
                                    ;
 014412  116205  000000G                    MOVB    SFUSER(R2),R5    ;GET USER INDEX #
 014416  006205                          ASR     R5                 ;CONVERT TO SEQUENTIAL NUMBER
 014420  012703  000002                  MOV     #2.,R3             ;PRINT 2 CHARS
 014424  004767  171614                  CALL    PRTFIX             ;PRINT VALUE
 014430                                  .PRINT  #SPACE2
                                 ;
                                 ;  Print file name
                                 ;
 014436  016200  000000G                 MOV     SFFILE(R2),R0      ;GET 1ST 3 CHARS OF FILE NAME (RAD50)
 014442  004767  171116                  CALL    PRTR50             ;PRINT THEM
 014446  016200  000002G                 MOV     SFFILE+2(R2),R0    ;PRINT 2ND 3 CHARS
 014452  004767  171106                  CALL    PRTR50
 014456                                  .PRINT  #SPACE2
                                 ;
                                 ;  Print form name
                                 ;
 014464  010203                          MOV     R2,R3
 014466  062703  000000G                 ADD     #SFFORM,R3         ;POINT TO CELL WITH FORM NAME
 014472  012704  000006                  MOV     #6.,R4             ;PRINT 6 CHARS
 014476  112300                  3$:     MOVB    (R3)+,R0           ;GET NEXT CHAR OF NAME
 014500                                  .TTYOUT                    ;PRINT IT
 014504  077404                          SOB     R4,3$
 014506                                  .PRINT  #SPACE2
                                 ;
                                 ;  Print number of blocks in spool file
                                 ;
 014514  016205  000000G                 MOV     SFNMBL(R2),R5      ;GET # BLOCKS IN SPOOL FILE
 014520  012703  000004                  MOV     #4.,R3             ;PRINT 4 CHARS
 014524  004767  171514                  CALL    PRTFIX             ;PRINT VALUE
                                 ;
                                 ;  end of print line
                                 ;
 014530                                  .PRINT  #CRLF
                                 ;
                                 ;  See if there are more spool files for this device.
                                 ;
 014536  016202  000000G         5$:     MOV     SFQLNK(R2),R2      ;CHAIN TO NEXT SFCB
 014542  001253                          BNE     1$                 ;BR IF MORE TO PRINT
                                 ;
                                 ;  Finished
                                 ;
 014544  012605                  9$:     MOV     (SP)+,R5
 014546  012604                          MOV     (SP)+,R4
 014550  012603                          MOV     (SP)+,R3
 014552  012602                          MOV     (SP)+,R2
 014554  012601                          MOV     (SP)+,R1
 014556  000207                          RETURN
```

```
                                          .SBTTL  CHKDLM -- See if char is a delimiter
                                  ;-------------------------------------------------------------------
                                  ;  CHKDLM IS CALLED TO SEE IF THE CHARACTER IN RO 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.
                                  ;
 014560  120027  000060           CHKDLM: CMPB    RO,#'0          ;IS CHAR A DIGIT?
 014564  103422                           BLO     1$              ;BR IF NOT
 014566  120027  000071                   CMPB    RO,#'9
 014572  101421                           BLOS    2$              ;BR IF DIGIT
 014574  120027  000141                   CMPB    RO,#141         ;IS THIS A LOWER CASE LETTER?
 014600  103406                           BLO     3$              ;BR IF NOT
 014602  120027  000172                   CMPB    RO,#172         ;LOWER CASE Z
 014606  101011                           BHI     1$              ;BR IF DELIMITER
 014610  162700  000040                   SUB     #40,RO          ;CONVERT LOWER-CASE TO UPPER CASE
 014614  000410                           BR      2$
 014616  120027  000101           3$:     CMPB    RO,#'A          ;IS CHAR A LETTER?
 014622  103403                           BLO     1$              ;BR IF NOT
 014624  120027  000132                   CMPB    RO,#'Z
 014630  101402                           BLOS    2$              ;BR IF LETTER
                                  ;  CHARACTER IS A DELIMITER
 014632  000261                   1$:     SEC                     ;SIGNAL DELIMITER
 014634  000207                           RETURN
                                  ;  CHARACTER IS ALPHANUMERIC
 014636  000241                   2$:     CLC
 014640  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.
                                    ;
 014642  010346                     CVTTAB: MOV     R3,-(SP)
 014644  112300                     1$:     MOVB    (R3)+,R0        ;Get next character from command
 014646  001412                             BEQ     4$              ;Br if end of command
 014650  120027  000011                     CMPB    R0,#TAB         ;Is this a tab character
 014654  001403                             BEQ     2$              ;Br if yes
 014656  120027  000014                     CMPB    R0,#FF          ;Is this a form-feed character?
 014662  001370                             BNE     1$              ;Br if not
 014664  112763  000040  177777     2$:     MOVB    #' ,-1(R3)      ;Replace control character with space
 014672  000764                             BR      1$
                                    ;
                                    ;  Finished converting tabs and Form-feeds to spaces.
                                    ;  Now skip over leading spaces.
                                    ;
 014674  012603                     4$:     MOV     (SP)+,R3        ;Get pointer to start of command
 014676  122327  000040             3$:     CMPB    (R3)+,#'        ;Skip leading spaces
 014702  001775                             BEQ     3$
 014704  005303                             DEC     R3              ;Point to 1st non-blank character
                                    ;
                                    ;  Finished
                                    ;
 014706  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.
                                ;
 014710  010246                 CVTUC:  MOV     R2,-(SP)
 014712  010302                         MOV     R3,R2           ;Get pointer to start of string
 014714  112200                 1$:     MOVB    (R2)+,R0        ;Get next char from string
 014716  001413                         BEQ     9$              ;Br if hit end of string
 014720  120027  000141                 CMPB    R0,#141         ;Is this a lower case letter?
 014724  103773                         BLO     1$              ;Br if not
 014726  120027  000172                 CMPB    R0,#172
 014732  101370                         BHI     1$              ;Br if not
 014734  162700  000040                 SUB     #40,R0          ;Convert letter to upper case
 014740  110062  177777                 MOVB    R0,-1(R2)       ;Store converted char back into string
 014744  000763                         BR      1$
                                ;
                                ;  Finished
                                ;
 014746  012602                 9$:     MOV     (SP)+,R2
 014750  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.
                                  ;
 014752  122327  000040           SKPSPC: CMPB    (R3)+,#'        ;IS NEXT CHARACTER A SPACE?
 014756  001775                           BEQ     SKPSPC          ;BR IF YES -- SKIP IT
 014760  005303                           DEC     R3              ;BACKUP POINTER TO FIRST NON-BLANK CHAR
 014762  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
                                  ;

 014764  010046                   SKPDLM: MOV     R0,-(SP)        ;Save register
 014766  112300                           MOVB    (R3)+,R0        ;Get next command character
 014770  001413                           BEQ     2$              ;Br if end of command hit
 014772  120027  000040                   CMPB    R0,#BLANK       ;Check for space
 014776  001403                           BEQ     1$              ;Br if space found
 015000  120027  000054                   CMPB    R0,#COMMA       ;Check for comma
 015004  001007                           BNE     3$              ;Character not blank or comma
                                  ;
                                  ;  Found legal delimiter - skip multiple spaces and commas.
                                  ;
 015006  004767  177740           1$:     CALL    SKPSPC          ;Skip multiple spaces
 015012  122327  000054                   CMPB    (R3)+,#COMMA    ;Next character a comma (following spaces?)
 015016  001773                           BEQ     1$              ;Br if comma found
 015020  000241                   2$:     CLC                     ;Flag legal delimiter found
 015022  000401                           BR      10$             ;Finished
                                  ;
                                  ;  Did not find a legal separator.
                                  ;
 015024  000261                   3$:     SEC                     ;Flag no legal delimiter found
                                  ;  Use no instructions which alter the c-bit before the RETURN.
 015026  005303                   10$:    DEC     R3              ;Back up to last not blank or comma character
 015030  012600                           MOV     (SP)+,R0        ;Restore register
 015032  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.
                                   ;
 015034  112300                    GETKCH: MOVB    (R3)+,R0        ;GET NEXT CHARACTER
 015036  120027  000141                    CMPB    R0,#141         ;IS IT A LOWER CASE LETTER?
 015042  103405                            BLO     1$              ;BR IF DEFINITELY NOT
 015044  120027  000172                    CMPB    R0,#172         ;CHECK UPPER RANGE
 015050  101002                            BHI     1$              ;BR IF NOT LOWER CASE
 015052  162700  000040                    SUB     #40,R0          ;CONVERT LOWER-CASE TO UPPER-CASE
 015056  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.
                                ;
015060  010246                  DELSPC: MOV     R2,-(SP)
015062  010346                          MOV     R3,-(SP)
015064  010302                          MOV     R3,R2
015066  112300                  1$:     MOVB    (R3)+,R0        ;GET NEXT CHAR FROM COMMAND LINE
015070  001405                          BEQ     2$              ;BR IF END OF COMMAND HIT
015072  120027  000040                  CMPB    R0,#'           ;IS THIS CHAR A SPACE?
015076  001773                          BEQ     1$              ;IF YES THEN SKIP OVER IT
015100  110022                          MOVB    R0,(R2)+        ;MOVE CHAR INTO NEW COMMAND LINE
015102  000771                          BR      1$
015104  105012                  2$:     CLRB    (R2)            ;PUT IN ASCIZ NULL AT END
015106  012603                          MOV     (SP)+,R3
015110  012602                          MOV     (SP)+,R2
015112  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.
                                ;
015114  004767  177632          CHKEQ:  CALL    SKPSPC          ;Skip over any spaces
015120  112300                          MOVB    (R3)+,R0        ;Get next command character
015122  120027  000075                  CMPB    R0,#'=          ;Is character equal sign?
015126  001407                          BEQ     1$              ;Br if yes
015130  120027  000072                  CMPB    R0,#':          ;Is it colon?
015134  001404                          BEQ     1$              ;Br if yes
015136                                  FABORT  #EM$CSE         ;Command syntax error
015146  004767  177600          1$:     CALL    SKPSPC          ;Skip over any spaces
                                ;
                                ;  Finished
                                ;
015152  000207                          RETURN
```

```
                                       .SBTTL  CKPRIV -- Check for OPER privilege
                               ;------------------------------------------------------------
                               ;  Determine if the current user has OPER privilege.
                               ;
 015154  032767  000000G 000000G CKPRIV: BIT     #PO$OPR,PRIVCO  ;Does user have OPER privilege?
 015162  001004                          BNE     9$              ;Br if yes
 015164                                  FABORT  #EM$OPR         ;Operator privilege required
 015174  000207                  9$:     RETURN

                                         .SBTTL  CKSYPV -- Check for SYSPRV privilege
                               ;----------------------------------------------------------------
                               ;  Check for SYSPRV privilege.
                               ;
 015176  032767  000000G 000000G CKSYPV: BIT     #PO$SYS,PRIVCO  ;Does user have SYSPRV privilege?
 015204  001004                          BNE     9$              ;Br if yes
 015206                                  FABORT  #EM$SPR
 015216  000207                  9$:     RETURN

                                         .SBTTL  CKTERM -- Check for TERMINAL privilege
                               ;--------------------------------------------------------------------
                               ;  Check to see if the user has TERMINAL privilege.
                               ;
 015220  032767  000000G 000000G CKTERM: BIT     #P2$TRM,PRIVC2  ;Does user have TERMINAL privilege?
 015226  001004                          BNE     9$              ;Br if yes
 015230                                  FABORT  #EM$TPR         ;Terminal privilege required
 015240  000207                  9$:     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.
                               ;
 015242  010346                  PRGALL: MOV     R3,-(SP)
 015244  012703  177777G                 MOV     #NUCHN-1,R3     ;GET # OF LAST CHANNEL TO PURGE
 015250  020327  000017          2$:     CMP     R3,#17          ;Don't purge channel 17
 015254  001404                          BEQ     3$
 015256                                  .PURGE  R3              ;PURGE ALL OF USER'S CHANNELS
 015266  005303                  3$:     DEC     R3
 015270  002367                          BGE     2$
 015272                                  .PURGE  #RUNCHN         ;Purge channel used to start SAV files
 015300  012603                          MOV     (SP)+,R3
 015302  000207                          RETURN
         000001                          .END
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 11869 Words  ( 47 Pages)
Size of core pool: 17920 Words  ( 70 Pages)
Operating  system: RT-11

Elapsed time: 00:02:18.25
DK:TSKMN3,LP:TSKMN3=DK:TSKMN3.MAC/C/N:SYM
```

```
$1STLG   1-46
$CARUP   1-59
$CCLRN   1-60
$CFABT   1-74     22-28     23-19
$CFALL   1-81     21-24
$CFCCL   1-81     21-9
$CFDCC   1-81     22-30     23-27
$CFKIL   1-66     23-17
$CFOPN   1-87     20-9      20-77     21-16     21-24     21-81
$CFSOT   1-79     21-24
$CLTST   1-70
$CTRLC   1-73
$CTRLO   1-29
$CTRLS   1-65
$DEAD    1-124
$DEFER   1-92
$DETCH   1-63
$DIBOL   1-46
$DILUP   1-77
$DISCN   1-64
$DOOFF   1-83
$DUPRN   1-78
$ECHO    1-80
$EMTTR   1-69
$FORM    1-79
$FORMO   1-81
$HITTY   1-35
$INCOR   1-96
$INDAB   1-67     56-47
$INDDF   1-123
$INDRN   1-123     8-16     23-25
$INIT    1-124
$INKMN   1-73
$KED     1-96
$KINIT   1-31     46-12     49-14
$LC      1-80
$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    1-80
$PHONE   1-124
$PRGLK   1-61
$QTSET   1-101    20-17
$QUIET   1-93     20-21
$SCCA    1-24      8-39      8-42
$SCOPE   1-80
$SNWTT   1-121
$SPLJB   1-65
$SUCF    1-36     21-26
$TAB     1-79
$TAPE    1-119
$TECO    1-101
$TTGAG   1-87
```

```
$UCLCF    1-89
$UCLRN    1-89
$VNOTT    1-28
$VTESC    1-56
$WILD     1-102
...V1    15-53     15-57     15-61     15-68     19-36     19-36     19-36     19-41     19-41     19-44     19-44     19-48
         19-48     19-48     19-52     19-52     19-53     19-53     20-32     20-32     21-19     21-19     21-74     21-74
         21-77     21-77     23-36     23-39     27-19     27-19     27-27     27-27     31-13     32-25     36-33     37-26
         42-19     44-29     45-10     45-26     45-29     45-42     46-20     46-23     49-30     49-38     49-41     50-14
         50-23     50-25     50-29     50-33     51-40     51-42     52-27     52-32     52-34     53-26     53-32     53-33
         53-34     53-35     54-10     54-21     54-23     54-43     54-45     56-6      56-7      56-18     56-19     58-59
         66-57     69-30     69-30     69-30     69-38     69-38     81-39     81-52     81-53     81-61     81-69     81-77
         81-79     81-87     87-37     87-40
...V2    19-36     19-36     19-36#    19-36#    19-41     19-41     19-41#    19-41#    19-44     19-44#    19-48     19-48
         19-48#    19-48#    19-52     19-52     19-52#    19-52#    19-53     19-53     19-53#    19-53#    20-32     20-32
         20-32#    20-32#    21-19     21-19#    21-74     21-74     21-74#    21-74#    21-77     21-77     21-77#    21-77#
         27-19     27-19     27-19#    27-19#    27-27     27-27#    69-30     69-30     69-30#    69-30#    69-38     69-38
         69-38#    69-38#    87-37     87-37#    87-40     87-40#
ABRTAD    1-28
ABRTCD    1-28
ABRTCF    1-131    22-8#     56-46
ACRDEC    1-139    28-14#    30-16
ACRFIL    1-138    58-17#
ACRFN     1-131    57-14#
ACROCT    1-141    29-14#
ACRPRV    1-46      5-14#    13-22
ACRSPD    1-140    30-12#
ACRSTR    1-27     33-68     34-14#
ACRTXT    1-26     33-14#
ADLDAC   69-65     71-10#
ADM3A     1-115    46-37
ADM3FL    1-116
ADM3NO    1-117
AF$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    1-154
ALDEX     1-153     1-154
ALFN      1-165
AMBOPT    1-139    12-50
ASDEX     1-136
ASKLNM    1-135
ASNEND    1-78
ASNHD1    1-148
ASNHD2    1-149
ASNOVF    1-161    73-29
ASNSRC    1-169    62-22     73-13     73-21     73-27     76-13#    77-22
ASNTBL    1-77     76-15     79-12
AT$$SZ    1-75     76-18     79-18
AT$DEV    1-75     77-29     79-14     79-17*
AT$EXT    1-75     77-34
AT$FIL    1-75     77-30     77-33
AT$LOG    1-75     76-16     79-16*
AT$SIZ    1-75     77-37
AUTHFN    1-158
```

```
Cross reference table (CREF V05.04)


BADBOT   1-167     19-50
BADCMD   1-135
BADPMT   1-142
BADPRI   1-142
BADSAV   1-135
BASMAP   1-110
BDFNAM   1-138     57-46
BDLGOP   1-143
BDLIN    1-156
BELL     1-176#
BLANK    1-175#    15-61     85-33
BLKO     1-134      4-30     19-41     19-43     33-36     33-62     34-35     34-62     58-30     58-59     69-37     81-27
BLKWDS   1-180#
BOTCSR   1-72      19-43*
BOTDEV   1-72      19-6      19-10     19-28*    19-48
BOTEMT   1-167     19-58
BOTHAN   4-54#     19-12*    19-24*    19-34*    19-36
BOTUNI   1-72      19-13*    19-29*
C.CSW    1-84      69-39     69-50
C.DEVQ   1-84      69-42
C.SBLK   1-84      69-45
CALUCL   1-147
CCLNAM   1-162
CCLSAV   1-68
CCSPRV   1-43       7-98
CD$$SZ   1-85      59-47     63-44
CD$$UB   1-85      59-34     64-26
CD$BAS   1-85      60-18     63-42
CD$DVU   1-85      59-20     60-16     60-28     63-40
CD$JOB   1-85      59-33     61-18     64-25
CD$NAM   1-85
CD$TOP   1-101
CDBUF    1-45      59-20     59-33     60-16     60-18     60-28     61-18     63-40     63-42     64-25     67-16
CDGADR   1-45      67-13*
CDGEMT   1-45      67-14
CDGET    1-45      59-19     63-39     67-13#
CDJFLG  59-22      61-13#    64-19
CDPADR   1-45      67-28*
CDPEMT   1-45      67-29
CDPUT    1-45      59-32     67-28#
CF$IND   1-125
CF$QUT   1-125
CF2DEP   1-167     20-98
CFABLV   1-126
CFACFL   1-26      24-14     24-38
CFBLK    1-93      20-34     20-85*    21-72*    21-77
CFBUF    1-68      20-83     21-77     25-22     25-41
CFCHAN   1-92      20-32     20-32     21-19     21-19     21-74     21-74     21-77     21-77
CFEND    1-68
CFHOLD   1-100     21-15*
CFIND    1-83      20-14*    20-17*    20-46     20-47*    21-14     21-54*
CFLFL4   1-88      20-18     21-56
CFNEST   1-86      20-78*    21-10     21-21     21-80*    22-17
CFOVFL  20-29      20-57     20-70     20-98#
CFPNT    1-93       8-14     20-15     20-35     21-12     22-19     23-15     24-9*     24-32*    25-23     25-36*
CFSEND   1-87      20-28     20-56     20-69
```

```
TSKMN3 -- TSKMON Subroutines    MACRO V05.04  Monday 21-Dec-87 11:41 Page S-4
Cross reference table (CREF V05.04)


CFSP       1-87      20-27      20-76*     21-35      21-79*
CFSPND     1-99      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      1-125
CHAIN      1-73
CHKALC     1-30      66-10#
CHKCLU     1-26      75-14#
CHKDEV     1-162     26-24      62-14#     63-20      66-21      80-58
CHKDLM     1-152     35-61      57-26      82-9#
CHKEQ      1-43      13-18      86-29#
CHKMNT     1-137     63-14#
CHKMTX     1-137     64-12#
CHKTTD     1-156     18-11#
CINDAT     1-109
CINFLG     1-28
CKACOJ     1-39      16-13#
CKCLUS     1-32      65-11#
CKLDAC    70-14#     72-9
CKPRIV     1-139     87-5#
CKSYPV     1-141     87-14#
CKTERM     1-139     87-23#
CL$XLN     1-27      65-15      65-17
CLDEVX     1-92      66-48
CLRPRV     1-43       5-19       6-8#
CMDBUF     1-130      1-147
CMDCCL     1-162
COAD       1-153      1-154
COAF       1-166
COAL       1-153      1-154
COLOO      1-147     50-25
COMMA      1-179#    15-53      85-35      85-41
CONFIG     1-82
CORUSR     1-29       8-13       9-9       16-17      16-36      20-8       21-8       22-9       23-7       56-43      61-13      66-51
CPUAH      1-147     51-63*     51-72*     54-10      54-11
CPUAL      1-147     51-64*     51-71*     54-12
CPUMSG     1-168     50-29
CR         1-174#
CRLF       1-142     15-57      50-33      54-45      66-57      81-89
CS$RON     1-55      69-50
CSHDEV     1-77      59-17      63-37
CSHDVN     1-77      59-48      63-45
CSHHD      1-129
CSIMS1     1-141
CSIMS2     1-139
CSIMS4     1-161
CTMSG      1-168     50-14
CTRLTT     1-71
CURMTX     1-152
CURPRM     1-98      20-49      20-86*     21-52*
CVDVNM     1-149     60-29      74-12#
CVTTAB     1-130     83-14#
CVTUC      1-126     84-8#
CW$50H     1-82
```

```
CXTBAS   1-76
CXTPAG   1-54
CXTRMN   1-26      24-13     24-37
CXTWDS   1-76
DATTIM   1-33      54-38#
DCCRD    1-105
DCCWR    1-105
DCTRD    1-105
DCTWR    1-105
DEADEV   1-137     79-10#
DELSPC   1-151     86-11#
DETARG   1-166
DETHD    1-166
DETTXT   1-145
DEVHD1   1-148
DEVIDL   1-153      1-153     1-154
DEVUNT   1-140
DFJMEM   1-31
DIABFL   1-93
DIABLO   1-114     46-40
DIABNO   1-94
DIV32   36-23      40-14#
DIVIDE   1-146     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   1-159     40-22     40-25     40-27     40-29
DJABMS   1-157
DKSAV    1-130
DLCEMT   1-50      59-12
DLLDAC  69-19      72-7#
DLMSG    1-157
DLTXT    1-144
DMTALL   1-157     59-5#
DMTARG   1-136     60-34     68-25
DMTSUB   1-162     59-43     60-10#
DOASGN   1-59      73-9#
DOSTOP   1-158     19-6#
DZTXT    1-164
EDIT     1-46
EDTFIL   1-150     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$TPR   1-38      87-25
ERRLOC   1-30      66-35     66-37
ERRSEV   1-102     56-44
FC$CDX   1-129
FC$LNK   1-129
FD$NAM   1-129
```

Cross reference table (CREF V05.04)

```
FF        1-178#    55-31     83-19
FILNAM    1-131      1-132    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     1-131      5-46     12-50     12-51     19-50     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    1-169     78-8#
FPRINT    1-131     16-58     27-21     56-6#     56-29     57-46     66-39     66-55     71-46
FSTDL     1-63
GAGMSG    1-156
GENTOP    1-72
GETKCH    1-50      85-66#
GRT1      1-111
GTRD50    1-136     12-21     35-10#    57-18     57-40
H.CSR     1-157     19-43
HANBSY    1-141
HANCHN    1-181#
HANENT    1-51
HANIDX    1-140
HANSIZ    1-51
HAZEL     1-52      46-36
HAZLFL    1-52
HAZLNO    1-52
HIMAP     1-110
HIPRI     1-142
HNBUF     1-140
HUPARG    1-161
II$$SZ    1-25      80-44
II$NAM    1-25      80-34     80-55
IIBUF     1-25      80-34     80-55
ILLCMD    1-137
ILSW2     1-58
IN$ACT    1-123      8-18     20-12     22-21
IN$CMD    1-123
IN$CNT    1-123      8-18
INDABT    1-126     23-6#     56-49
INDACT    1-133
INDCOL   15-16*     15-58     15-84#
INDERR    1-74      56-20*    56-42*
INDRDB    4-34#     23-35*    23-36     23-38*    23-39
INDSAV    1-123
INDSTA    1-74       8-18     20-12     20-14     20-17     20-47     20-93*    21-14*    22-21     23-24*
INFOMT    1-137
INGADR    1-25      80-31*
INGEMT    1-25      80-32
INSSPC    4-55#     80-21     80-25     80-35
INSSRC    1-169     80-14#
INSTBL    1-25      80-30
INSTBN    1-25      80-45
INVDAT    1-166
INVDEV    1-165
INVEC     1-124
INVLDM    1-161
INVLDN    1-138
INVOPT    1-131      1-137     1-152    12-51
INVTIM    1-157
IOABFL    1-29
```

```
ITRMTP   1-125    46-14
JCXPGS   1-110
JCXSMS   1-164
JSTKND   1-71
JSWLOC   1-30
K52      1-46
KBMSG    1-146
KBTX     1-150
KCSIBF   1-135    58-50     58-52     58-59
KED      1-46
KEYBUF   1-168    55-38     55-80     55-89
KEYEND   1-168    55-64
KILEMT   1-158
KL3CLR   1-61
KL4CLR   1-95
KMFTXT   1-168    56-6
KMNBAS   1-120
KMNCHN   1-68
KMNERR  56-8      56-31     56-41#
KMNHI    1-53
KMNNAM   1-162
KMNPGS   1-54
KMNSTK   1-54
KMNSTR   1-54
KMNTOP   1-54
KMPRMT   1-106
KMSTK    1-169    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    1-114    46-39
LA12FL   1-115
LA12NO   1-115
LA36     1-114    46-38
LA36FL   1-94
LA36NO   1-94
LACTIV   1-55
LAFSIZ   1-58
LCBIT    1-114
LCOL     1-57      1-101
LCONTM   1-65     50-16
LCPUHI   1-65     50-30
LCPULO   1-65     50-31
LD$RON   1-90     69-52     71-39
LDBASE   1-91     26-45     60-18     63-29     69-45*    69-61     69-61*
LDCLEN   1-33     68-6#
LDDEVX   1-92     26-30     62-77     63-25     66-22     69-57     70-23     71-31     80-60
```

```
TSKMN3 -- TSKMON Subroutines    MACRO V05.04  Monday 21-Dec-87 11:41 Page S-8
Cross reference table (CREF V05.04)


LDFLAG   1-91     69-52*   71-39
LDMNT    1-34     68-32    69-14#
LDNAM    1-135
LDNAME   1-91     68-19    69-26
LDOPHD   1-143
LDPDEV   1-91     26-44    60-16    62-81    63-28    68-36    69-41*   69-44*   69-60    69-60*   69-74*
LDSIZE   1-91     26-47    69-33*
LF       1-173#
LF$OPN   1-128    26-18    27-6     27-26
LF$WRT   1-128
LFWLIM   1-58
LGOVER   1-167    27-21
LICTXT   1-111
LINBUF   1-53
LINCNT   1-55
LINCUR   1-58
LINFRE   1-157
LINNXT   1-53
LINPNT   1-55
LJSW     1-71
LMXLN    1-124
LMXNUM   1-122
LMXPRM   1-125
LNBLKS   1-76
LNMAP    1-82
LNPRIM   1-82     16-37    16-37
LNSBLK   1-77
LNSPAC   1-86
LOCKTX   1-146
LOGASN   1-138    77-16#   80-26
LOGBAS   1-59     26-50    26-52
LOGBLK   1-127    27-19
LOGBUF   1-127    27-16    27-19
LOGCHK   1-59     26-11#
LOGCHN   1-127    27-19    27-19    27-27    27-27
LOGCLS   1-142    26-59    27-5#
LOGDVU   1-59     26-37    26-48
LOGFLG   1-127    26-18    27-6     27-26*
LOGPTR   1-127    27-11    27-25*
LOMAP    1-110
LOTBUF   1-56
LOTNXT   1-56
LOTPNT   1-56
LOTSIZ   1-57
LOTSPC   1-57
LPARNT   1-35     16-39
LPRG1    1-112
LPRG2    1-112
LPROG    1-64     16-51    16-51    49-42
LPROJ    1-64     16-44    16-44    49-36
LRBFIL   1-83
LRDTIM   1-55
LSCCA    1-79
LSECPT   1-69
LSTACT   1-53
LSTATE   1-108
```

Cross reference table (CREF V05.04)

```
LSTDL      1-63      16-24
LSTMX      1-122
LSTPL      1-103     16-22
LSTPRM     1-99      20-63      20-89      21-41
LSTSL      1-108
LSTSPL     1-28      81-9#
LSUCF      1-60
LSW        1-29      46-12      49-14
LSW11      1-24       8-46*      8-49*
LSW2       1-73       9-10*      9-21*     20-19
LSW2S      1-42       9-14*      9-22*
LSW3       1-78      21-25*
LSW4       1-95      20-9       20-18*     20-21*     20-44      20-77*     21-9*      21-16      21-24*     21-57*     21-60*     21-81*
          22-30*     23-77*
LSW5       1-61       8-16       8-39*      8-42*     23-25*
LSW6       1-121     22-28*     23-19*
LSW7       1-125     56-47
LSW9       1-52      21-26*     22-29*     23-26*
LTRMTP     1-115     46-11
LTSCMD     1-86
LUNAME     1-64      49-27
LWINDO     1-24
MAXASN     1-74      76-14      79-13
MAXAVL     1-140
MAXMEM     1-30
MAXMTX     1-152
MAXSEC     1-69
MDT        1-52
MINTIM     1-69      50-15
MISSEQ     1-141
MNTARG     1-161     68-39
MNTDEV     1-136     60-33*     68-24*     68-38*
MNTFUL     1-138
MNTTXT     1-163
MONAR1     1-151
MONAR2     1-151
MONHD      1-151
MONTAB     1-168     53-31
MONVEC     1-126
MSGBUF     1-156
MSGEND     1-156
MTOPHD     1-137
MUL32      1-166     36-19      41-14#     51-62
MXCSR      1-122
MXDTR      1-122
MXJADR     1-32
MXJMEM     1-31
MXPRMT     1-106
MXVEC      1-124
NAMTOP     1-111
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    1-144
NEGFLG   4-59#    28-26*   28-29*   28-62    29-26*   29-29*   29-57
NEWJSW   1-71
NFRESB   1-97
NINTXT   1-144
NLINES   1-84
NOCCL    1-143
NOCIN    1-135
NODAT    1-149
NOFIL    1-147     1-155
NOFLAG   1-137
NOFRDL   1-166
NOIND    1-141
NOKMON   1-162
NOLDMT   1-149
NONEMS   1-149
NOOPTT   1-155
NOPMGN   1-151
NOPRG    1-131     1-135
NOSTRT   1-134
NOTAVL   1-150
NOTON    1-156
NOTXT    1-144
NOUDC    1-148
NSWPMS   1-152
NTRMTP  46-15     46-44#
NUCHN    1-83     87-34
NUMDEV   1-117    62-57
NUMON    1-58
OCTFIX   1-144    32-7#
OCTPRT   1-164    31-7#
OF$$SZ   1-119    70-27    71-26
OF$DEV   1-118    70-23    71-31*   72-15*
OF$FIL   1-118    71-24    71-34*   71-35*   71-36*   72-17*
OF$FLG   1-118    71-37*   71-41*
OF$UNT   1-118    70-25    71-32*   72-16*
OFFEMT   1-158
OKFEND   1-70     70-28    71-22
OKFILE   1-70     70-22    71-21
OPTLST   1-47     10-10#
OT$RON   1-119    71-41
OTHRON   1-160
OTRMNT   1-162
OVRCOR   1-134
PO$$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$WRL   1-39     3-55     3-56     16-31
PASLIN   1-102
PAUMSG   1-130
PBFEND   1-87    20-50    20-91*   21-47*
PBUFND   1-167   42-11    44-17
PEKADR   1-27     4-28#   81-23*
PEKEMT   1-27     4-26#   81-25
PEKSIZ   1-27     4-29#   81-24*
PF$IOW   1-121
PF$SYS   1-121
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   1-151
PNAME    1-117    1-140   19-23    19-34    62-58    74-17
POPCF    1-101   21-7#    22-23    23-17
PPNMSG   1-168   49-38
PRGALL   1-34    87-30#
PRGSIZ   1-53
PRGTOP   1-53
```

```
PRIVCO    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
PRIVFO    1-44      8-21      8-29     20-37     20-39     21-64     21-66
PRIVSO    1-44      7-13      8-20
PRMBUF    1-99     20-53     20-91     21-48
PRMEND    1-99
PRMPNT    1-98     20-66     20-73     20-87     21-36
PRTBUF    1-149
PRTDAT    1-165    53-6#     54-42
PRTDC2    1-146    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    1-146    45-24#
PRTDEC    1-140    36-28     42-6#     45-12     45-30     49-39     49-43     52-25     53-25     66-56
PRTFIX    1-143    43-14     44-12#    81-38     81-60     81-85
PRTFNM    1-149    38-14#
PRTLN     1-145    43-10#
PRTPCT    1-159    36-10#
PRTR50    1-165    37-8#     81-44     81-66     81-68
PRTSPC    1-143    45-40#
PRTTIM    1-165    50-7#
PRTTMD    1-150    52-9#
PRTTMV    1-147    50-32     51-14#
PRTTOD    1-165    54-6#     54-44
PRTTTP    1-144    46-9#
PRTUNM    1-145    49-9#
PRTWRN    1-132    26-58     56-18#
PRVHD     3-11#     5-33     15-38
PRVLST    1-43     15-16#
PRVOPT    1-27     13-14#
PUSHCF    1-132    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      1-114    46-41
QUMEFL    1-116
QUMENO    1-116
R$CFST    1-26     24-14*    24-38*
R.GSTS    1-47     23-35*    23-38*
R50BUF    1-136    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    1-167    37-18     38-28     48-33
R50CL     4-48#    75-18
R50CLO    4-49#    65-29     75-22     75-26
R50CL7    4-50#    75-24
R50DIR    1-132
R50DK     1-165    62-45     77-45
R50DSK    1-138
R50DUP    1-133
R50IND    1-132
R50K52    1-133
R50KED    1-133
R50KEX    1-133
R50LD     1-138
R50LDO    1-136    60-14     68-11
R50LD7    1-138
```

```
R50LOG   1-142
R50MON   1-166
R50NO    1-139     12-22
R50PIP   1-133
R50SY    1-132     62-47     77-43
R50TT    1-161     18-11
R50TT0   4-46#     18-13
R50TT7   4-47#     18-15
R50VIR   1-134
RDB      1-107
RDBEND   1-107
RDCMD    1-130     56-32     66-58
RDERM    1-134
REMNDR   1-167     40-32#    40-33#
RESDEV   1-119     70-16     71-10
RNMS     1-145
RONTXT   1-150
RS.CRR   1-47
RS.EGR   1-47      23-38
RS.GBL   1-47      23-35
RS.PVT   1-47      23-35
RSR      1-122
RSTPRV   1-43       8-6#     21-28     21-68     22-35     23-31
RT$$SZ   1-107
RT$NAM   1-107
RUNCHN   1-50      87-40     87-40
RUNDEV   1-76      80-57
RUNEMT   1-134
RUNFLG   1-37       8-38#     8-40      8-47
RUNHD    1-130
RUNMS    1-166
S$INWT   1-67
S$IOFN   1-62
S$IOWT   1-112
S$MSWT   1-68
S$OTFN   1-62
S$OTLO   1-62
S$OTWT   1-67
S$QUSR   1-112
S$SFWT   1-67       1-112
S$SPCB   1-113
S$SPDB   1-113
S$SPND   1-61
S$TMWT   1-67
S$TTFN   1-62
S$TWFN   1-62
SC$SEV   1-70      56-42     56-44
SC$WRN   1-69      56-20
SCHAIN   1-92
SCNOPS   1-49      11-10#
SD$BAK   1-95
SD$DEL   1-88
SD$FLK   1-89
SD$HLD   1-98
SD$SNG   1-97
SD$WFM   1-89
```

```
SDBLK    1-90
SDBU     1-95
SDBUF1   1-90
SDCB     1-103    17-11
SDCBND   1-103    17-12
SDCBSZ   1-108    17-16
SDFHD    1-100    17-14    81-14
SDFLAG   1-89
SDFORM   1-89
SDNAME   1-108     1-152   81-43
SDSFCB   1-88
SDSKIP   1-95
SEARCH   1-130    12-32    55-19#
SERFLG   1-29     68-26*   68-28*   69-29*   69-31*
SETHD    1-139
SETWRD   1-46      5-34    10-21    11-31    12-13#
SF$1ST   1-98     81-31
SF$BSY   1-97     81-49
SF$HLD   1-98
SFCBSZ   1-48     81-24
SFFILE   1-113    81-65    81-67
SFFLAG   1-100    81-31    81-49
SFFORM   1-97     81-74
SFID     1-50     81-36
SFNMBL   1-97     81-83
SFQLNK   1-100    81-73
SFUSER   1-113    81-57
SH$$SZ   1-48
SH$FLG   1-48
SH$NAM   1-48
SH$RTN   1-48
SH$VAL   1-48
SHMTH1   1-148
SHMTH2   1-148
SHOHD    1-163
SHTMSG   1-158
SIZEMT   1-161
SIZVAL   1-135
SJSPPN   1-169     4-60#   49-19
SKPDLM   1-70     85-30#
SKPSPC   1-70      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   1-145    54-43    81-39    81-53    81-61    81-69    81-79
SPACE3   1-145
SPACE5   1-146
SPACE6   1-149
SPACTV   1-153     1-154
SPCF     1-155
SPDVAL   4-7#     30-30
SPFLK    1-155
SPFUL    1-155
SPGEMT   1-155
SPLACT   1-158    17-10#
```

```
SPLCHN   1-66
SPLHD    1-152
SPLHLA   1-143
SPLND    1-90
SPLPND   1-160
SPSNG    1-153     1-155
SPUBUF   1-32
SPWFM    1-153     1-154
SRTSMS   1-160
SRTTXT   1-163
START    1-167    19-52    19-53
STLGHD   1-142
STPASK   1-160
STPFLG   1-66
SUBARG   1-150
SUBTXT   1-163
SUCS     1-111
SUM1     1-159
SUM2     1-159
SUM3     1-159
SUM4     1-159
SUM5     1-160
SUM6     1-160
SUM7     1-160
SUMS     1-111
SUPCOD   1-111
SWPTX    1-146
SXBPNT   1-32
SYHD1    1-145
SYHD2    1-145
SYINDX   1-117    19-22    19-33    62-49
SYNAME   1-118    77-47
SYSAV    1-130
SYSDAT   1-109
SYTIMH   1-109
SYTIML   1-109
SYUNIT   1-117    19-25    62-52
TAB      1-177#   55-29    83-17
TALEMT   1-60     65-31    66-27
TBLOVF   1-139    71-46
TECO     1-46
TK1SEC   1-57     51-22    51-53    54-13
TK1VAL   1-109    51-70
TK5VAL   1-96     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      1-159
TMTOTL   1-33      1-159
TMUSRH   1-33
TOTMMS   1-163
TOTON    1-66
TOTXT    1-142
```

Cross reference table (CREF V05.04)

```
TRGRET    1-111
TRMSTR    1-132
TSKMN3    1-5#      1-29
TSR       1-122
TSXLN     1-111
TSXSIT    1-111
TSXSMS    1-164
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$MDC    1-129
UC$NDC    1-129
UCHAN     1-81
UCLBLK    1-128
UCLDAT    1-128
UCLNAM    1-86
UERSEV    1-102    22-31#    23-20#
UFORM     1-73
UFPTRP    1-88
UHIMEM    1-76
UMSSMS    1-163
UMSYTP    1-63
UPTMMS    1-158
USPLCH    1-66
USRMMS    1-164
USRSTK    1-31
USTART    1-72
UTRPAD    1-30
VCORTM    1-106
VHIPCT    1-104
VIMAGE    1-71
VQUAN1    1-104
VQUAN2    1-104
VQUN1A    1-104
VT100     1-114    46-35
VT10FL    1-116
VT10NO    1-116
VT2007    1-113    46-42
VT2008    1-113    46-43
VT52      1-114    46-34
VT52FL    1-115
VT52NO    1-94
WILDFL    1-35
WLDNAM    1-35     48-19     71-33     80-37     80-55
WRNHED    1-133    56-18
XAREA     1-131    19-36     19-41     19-48     19-52     19-53     20-32     21-74     21-77     23-36     23-39     27-19
         54-10     69-30     69-38
```

```
YESTXT   1-144
ZCLR     1-122
```

```
TSKMN3 -- TSKMON Subroutines     MACRO V05.04  Monday 21-Dec-87 11:41 Page M-1
Cross reference table (CREF V05.04)


...CM0   58-59     58-59     58-59
...CM1   19-36     19-41     19-48     19-52     19-53     20-32     21-74     21-77     27-19     69-30     69-38
...CM2   19-36     19-36     19-41     19-41     19-41     19-41     19-48     19-48     19-52     19-52     19-52     19-52
         19-53     19-53     19-53     19-53     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   19-44     21-19     27-27     87-37     87-40
...CM5   15-53     15-57     15-61     15-68     19-36     19-41     19-48     19-52     19-53     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   19-41     19-52     19-53     21-77     27-19
.CLOSE    1-17#    19-44     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#    19-36     19-48     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#    19-41     19-52     19-53     21-77
.REOPE    1-15#    21-74
.SAVES    1-15#    20-32     69-38
.SERR     1-19#    19-35
.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     19-50     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
```