TSUSR -- File operation EMT's MACRO V05.05 Wednesday 18-Jan-89 12:12
Table of contents

```
                                        .TITLE  TSUSR  -- File operation EMT's
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
                                ;
                                ;  TSUSR is the TSX module that contains the processing routines for
                                ;  EMTs that perform file operations such as .lookup, .enter, .delete,
                                ;  .rename, etc.
                                ;
                                ;  Copyright (c) 1980, 1981, 1982, 1983, 1984,1985.
                                ;  S&H Computer Systems, Inc.  Nashville, TN
                                ;  All rights reserved.
                                ;
000000                                  .CSECT  TSUSR
000000  103112                  TSUSR:  .RAD50  /USR/
                                ;
                                ;  Macro calls
                                ;
                                        .MCALL  .READW,.WRITW,.WAIT
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  TSUSR,GETUSR,USRTOP,USRINI
                                        .GLOBL  LOOKUP,ENTER,CLOSE,DELETE,RENAME
                                        .GLOBL  DSTAT,FREUSR,USRUCA,ALCEMT,CLRDIR
                                        .GLOBL  SFTIME,SFDATE,SFPROT,GFINFO
                                        .GLOBL  MOUNT,DISMNT,FRESPD,FETCH,CPYMNT
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  EXCJOB,KMONCE
                                        .GLOBL  DVSTAT,CHNADR,FILSPC,EMTBLK,ASNTBL,LDDEMT,LDIEMT
                                        .GLOBL  LDAEMT,ASNEND,CORUSR,S$QUSR,MAXLD,KPAR6,WLDNAM
                                        .GLOBL  AT$LOG,AT$SIZ,AT$DEV,AT$FIL,AT$EXT,AT$$SZ
                                        .GLOBL  PO$ALC,PRIVCO,PO$BYP,PO$NFR,PO$NFW
                                        .GLOBL  CHKABT,UREGO,NUMDEV,LSTFDT,LSTFDD
                                        .GLOBL  TK3SVL,FD$TIM,SYTIMH,SYTIML,FC$CDX
                                        .GLOBL  QNSPNX,JCDB,DVFLAG,DX$RAL,DX$NMT
                                        .GLOBL  PNAME,CHNNUM,SYINDX,SYUNIT,RUNFLG,AF$BYA
                                        .GLOBL  C.CSW,C.SBLK,C.LENG,C.USED,C.NUMQ,C.DEVQ
                                        .GLOBL  CS$NMX,CS$ENT,SERFLG,$KINIT,NLCHN
                                        .GLOBL  CS$OPN,CHKSD,CS$SPL,CSHCLN,CSHALC
                                        .GLOBL  LSW2,RESDEV,OKFILE,OF$DEV,LSW6,OKFAND,OKFNND
                                        .GLOBL  OF$UNT,OT$RON,OF$FLG,OF$FIL,OF$$SZ
                                        .GLOBL  OKFEND,LSW5,CS$RON,ERRSPC,CS$EOF,CS$ERR
                                        .GLOBL  DS$NRD,CL$LIX,PRIVCO,PO$SYS,DX$NRD
                                        .GLOBL  DS$DIR,SDCLOS,SFCLS,VMXFIL,$UCLRN,LSW7
                                        .GLOBL  FD$STA,FD$NAM,FD$LEN,FD$JOB,FD$CHN
                                        .GLOBL  FD$DAT,FD$OPT,CPLEMT,EMTCAD
                                        .GLOBL  FS$TEN,FS$EMP,FS$PRM,FS$EOS,VNFCSH
                                        .GLOBL  ALCTBL,ALCEND,AD$DVU,AD$JOB,AD$FLG,AD$$SZ
                                        .GLOBL  DH$NSG,DH$NXT,DH$HIS,DH$NEB,DH$BLK
                                        .GLOBL  DH$$SZ,DH$$BS,LDFLAG,LD$RON
                                        .GLOBL  EMTXIT,SETERR,URO,SYSCHN
                                        .GLOBL  SYSDAT,GETCHA,LSW,$DILUP,LDSIZE
                                        .GLOBL  FC$$SZ,BADEMT,LNPRIM,CLTOTL
                                        .GLOBL  IOWAIT,GETCXT,FRECXT
```

```
                        .GLOBL  UACHKW, VALADB, VALADW
                        .GLOBL  HANSIZ, HANENT, DEVSIZ, EMTPS
                        .GLOBL  SYSCHN, LSTSL, CHNSIZ
                        .GLOBL  CURCP, R$CHN, R$XCHN, FS$PRO
                        .GLOBL  $NOIN, LSW3, LDDEVX, CLDEVX, C1DEVX
                        .GLOBL  ABORT, EMTADR, USREMT, CSHHD, FC$LNK
                        .GLOBL  FD$$SZ, FC$SBL, CSHDEV, CSHDVN, Q.JNUM
                        .GLOBL  Q.CSW, Q.FUNC, Q.UNIT, Q.JOB, Q.DEVX
                        .GLOBL  VPAR6, Q.BUFF, Q.PAR, Q.WCNT, Q.COMP, Q.BLKN
                        .GLOBL  Q.CHAN, SPUSR, S$QSPD, Q.UCSW, Q.ICSW
                        .GLOBL  DF$LOK, DF$ENT, DF$DEL, DF$CLS, Q.PA6
                        .GLOBL  OVRHC, INTPRI, PR7, PSW, LDPDEV, CLCLOS
                        .GLOBL  USRJOB, SPDJOB, SPSIZE, SPFLDV, SPFLNM, GETQ, QIO
                        .GLOBL  $DUPRN, UMODE, EMTPS, ODTBAS
                        .GLOBL  CD$DVU, CD$BAS, CD$$SZ, LDBASE, CD$TOP
                        .GLOBL  CD$JOB, CD$$UB, CD$NAM, LDNAME, NSPLDV, CLZERR
```

```
                                ;
                                ;  Macro definitions.
                                ;
                                        .MACRO  DISABL          ;Disable interrupt processing
                                        BIS     #PR7,@#PSW
                                        .ENDM   DISABLE

                                        .MACRO  ENABL           ;Enable interrupt processing
                                        BIC     INTPRI,@#PSW
                                        .ENDM   ENABL

                                ;
                                ;  Macro definition for calling global routines residing in mapped system regions.
                                ;
                                        .MACRO  OCALL   ENTADD
                                        .IF     B,ENTADD
                                                .ERROR  ;OCALL SPECIFIED WITH NO ENTRY ADDRESS
                                                .MEXIT
                                        .ENDC
                                        CALL    OVRHC           ;CALL THE OVERLAY HANDLER
                                        .WORD   ENTADD          ;SPECIFY THE ENTRY POINT
                                        .ENDM
                                ;
                                ;  Assembly parameters.
                                ;
000300                          USRUCA  =       300             ;USR-User communication area
                                ;
                                ;  Data areas
                                ;
000002                          USRBUF: .BLKW   512.            ;Buffer to hold a directory segment
002002  177777                  CURSEG: .WORD   -1              ;Number of segment currently in USRBUF
002004  000000                  DIRHIS: .WORD   0               ;Highest segment # used in directory
002006  000000                  DIRSIZ: .WORD   0               ;Number of bytes per directory entry
002010  000000                  DIRNSG: .WORD   0               ;Number of directory segments available
002012  000000                  FILBLK: .WORD   0               ;Block number of start of file
002014  000000                  FILDVU: .WORD   0               ;Dev # in low byte, unit # in high byte
002016  075250                  R50SY:  .RAD50  /SY/
002020  015270                  R50DK:  .RAD50  /DK/
002022  075273                  R50SYS: .RAD50  /SYS/
002024  100020                  R50TSX: .RAD50  /TSX/
002026  177777                  USRCNT: .WORD   -1              ;# of times we have claimed usr
002030  000000                  L1SIZ:  .WORD   0               ;Size of largest free slot found
002032  000000                  L1OFF:  .WORD   0               ;Address of entry in dir segment buffer
002034  000000                  L1SEG:  .WORD   0               ;# of directory segment with largest slot
002036  000000                  L2SIZ:  .WORD   0               ;Size of second largest free slot found
002040  000000                  L2OFF:  .WORD   0               ;Address of entry in dir segment buffer
002042  000000                  L2SEG:  .WORD   0               ;# of dir segment with 2nd largest slot
002044  000000                  ASNSIZ: .WORD   0               ;Size specified for file with ASSIGN
                                ;
                                ;  Byte data
                                ;
002046  000                     CKADEV: .BYTE   0
002047  000                     CKAUNT: .BYTE   0
002050  000                     CNSLXD: .BYTE   0               ;CONSOL routine reorganized segment flag
                                ;
                                ;  Internal USR error checks.
                                ;  (Numbers correspond to KMON error table offsets)
```

```
                                ;
177762                  UERR1   =       -16     ;Can't find tentative file entry for file on close
177761                  UERR2   =       -17     ;File length in chan block not = to len in file entry
177760                  UERR3   =       -20     ;Highest block # written > file length
177757                  UERR4   =       -21     ;Empty file entry doesn't follow tentative file entry
177756                  UERR5   =       -22     ;Not tentative file entry during close
177755                  UERR6   =       -23     ;Illegal or uninitialized directory
                                .EVEN
```

```
                        .SBTTL  USRINI -- Initialization for USR module
                ;-----------------------------------------------------------------------
                ;  USRINI is called during system initialization to perform data table
                ;  allocation for the TSUSR module.
                ;
002052  010246  USRINI: MOV     R2,-(SP)
002054  010346         MOV     R3,-(SP)
                ;
                ;  Allocate tables within this overlay segment above top of code
                ;
002056  012702  015466'      MOV     #USRTOP,R2      ;Point above top of code in this segment
                ;
                ;  Allocate file directory cache table
                ;
002062  010237  000000G      MOV     R2,CSHHD        ;Set pointer to 1st free directory cache entry
002066  013700  000000G      MOV     VNFCSH,R0       ;Get # cache entries to allocate
002072  010203         1$:     MOV     R2,R3           ;Get address of current entry
002074  062703  000000G      ADD     #FC$$SZ,R3      ;Get address of next entry
002100  020327  140000       CMP     R3,#140000      ;Will current entry fit in overlay region?
002104  101403               BLOS    2$              ;Br if yes
002106  162703  000000G      SUB     #FC$$SZ,R3      ;No -- Backup pointer to current entry
002112  000406               BR      3$
002114  010362  000000G  2$: MOV     R3,FC$LNK(R2)   ;Make current entry point to next one
002120  005062  000000G      CLR     FC$CDX(R2)      ;Say entry is currently free
002124  010302               MOV     R3,R2           ;Make next entry be current one
002126  077017               SOB     R0,1$           ;Loop if more to allocate
002130  162703  000000G  3$: SUB     #FC$$SZ,R3      ;Point to last entry
002134  005063  000000G      CLR     FC$LNK(R3)      ;Make its forward link zero
                ;
                ;  Finished
                ;
002140  012603               MOV     (SP)+,R3
002142  012602               MOV     (SP)+,R2
002144  000207               RETURN
```

```
                                        .SBTTL  .LOOKUP
                                ;-------------------------------------------------------------------------------
                                ;  Perform a file lookup.
                                ;
002146  004737  007376'         LOOKUP: CALL    USRCOM          ;DO COMMON PARAMETER SETUP
002152  005037  000000G                 CLR     URO             ;FOR NOW, SAY FILE SIZE = 0 BLOCKS
                                ;
                                ;  Now R3 = Address of channel block, R4 = Index into device tables.
                                ;
                                ;  See if the device is spooled.
                                ;
002156  105737  000000G                 TSTB    NSPLDV          ;Are there any spooled devices?
002162  001404                          BEQ     1$              ;Br if not
002164                                  OCALL   CHKSD           ;See if we are opening to spooled device
002172  103063                          BCC     3$              ;Br if this is a spooled device
                                ;
                                ;  This is not a spooled device.
                                ;  See if this is a magnetic tape.
                                ;
002174  032764  000000G 000000G 1$:     BIT     #DS$NRD,DVSTAT(R4) ;IS THIS A NON-RT-11 DIRECTORY DEVICE?
002202  001004                          BNE     6$              ;Br if yes
002204  032764  000000G 000000G         BIT     #DX$NRD,DVFLAG(R4);Internal non-RT-11 dir flag set?
002212  001404                          BEQ     5$              ;BR IF NOT
002214  012702  000000G         6$:     MOV     #DF$LOK,R2      ;CODE FOR LOOKUP
002220  000137  014212'                 JMP     SPLDIR          ;DO SPECIAL DEVICE LOOKUP
                                ;
                                ;  See if this is a file structured device.
                                ;
002224  032764  000000G 000000G 5$:     BIT     #DS$DIR,DVSTAT(R4);DIRECTORY STRUCTURED DEVICE?
002232  001443                          BEQ     3$              ;Br if not -- We are finished
                                ;
                                ;  This is a file structured device.
                                ;  See if this is a non-file-structured lookup (null file name).
                                ;
002234  005737  000002G                 TST     FILSPC+2        ;IS FILE NAME NULL?
002240  001440                          BEQ     3$              ;BR IF NON-FILE-STRUCTURED LOOKUP
                                ;
                                ;  Lookup file name on a directory structured device.
                                ;  See if file info is stored in directory cache.
                                ;
002242  012701  000002G                 MOV     #FILSPC+2,R1    ;POINT TO NAME OF FILE
002246  004737  015054'                 CALL    CSHCHK          ;LOOK FOR FILE IN DIRECTORY CACHE
002252  103013                          BCC     2$              ;BR IF FOUND IT
                                ;
                                ;  File info was not found in directory cache
                                ;  so we must look at real directory.
                                ;
002254  004737  011376'                 CALL    FNDFIL          ;LOOKUP FILE NAME IN DEVICE DIRECTORY
002260  103004                          BCC     4$              ;BR IF FOUND
002262  012700  000001                 MOV     #1,R0           ;RETURN ERROR CODE OF 1 IF FILE DOESN'T EXIST
002266  000137  015356'                 JMP     USRCLS
                                ;
                                ;  Add file entry to directory cache.
                                ;
002272  010046                  4$:     MOV     R0,-(SP)        ;SAVE FILE STARTING BLOCK #
002274  004737  014700'                 CALL    CSHADD          ;ADD FILE INFO TO DIRECTORY CACHE
002300  012600                          MOV     (SP)+,R0        ;GET BACK FILE STARTING BLOCK #
```

.LOOKUP

```
                                        ;
                                        ;  Found file.  Save info about file in channel.
                                        ;  (R1 now points to directory entry for file).
                                        ;  (R0 Contains starting block number of the file).
                                        ;
002302  010063  000000G         2$:     MOV     R0,C.SBLK(R3)   ;SET FILE STARTING BLOCK #
002306  016100  000000G                 MOV     FD$LEN(R1),R0   ;GET ALLOCATED SIZE OF FILE
002312  010063  000000G                 MOV     R0,C.LENG(R3)   ;STORE IN CHANNEL BLOCK
002316  010037  000000G                 MOV     R0,UR0          ;RETURN TO USER IN R0
002322  016137  000000G 000000G         MOV     FD$TIM(R1),LSTFDT;SAVE TIME ENTRY FROM EACH LOOKUP
002330  016137  000000G 000000G         MOV     FD$DAT(R1),LSTFDD;ALSO SAVE DATE ENTRY
                                        ;
                                        ;  Allow user to diddle with directory entry if he wants to.
                                        ;
002336  004737  013456'                 CALL    DIDDLE          ;ALLOW DIRECTORY ENTRY DIDDLE
                                        ;
                                        ;  Finished
                                        ;
002342  000137  015404'         3$:     JMP     USRXIT          ;EXIT FROM USR PROCESSING
```

```
                                        .SBTTL  .ENTER
                                ;-------------------------------------------------------------------------------
                                ;  Open a new file.
                                ;
002346  004737  007376'         ENTER:  CALL    USRCOM          ;DO COMMON SETUP
002352  005037  000000G                 CLR     URO             ;FOR NOW, SAY FILE SIZE = 0
                                ;
                                ;  See if we have write access to the file.
                                ;
002356  032763  000000G 000000G         BIT     #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO THE FILE?
002364  001406                         BEQ     4$              ;BR IF YES
002366  004737  015426'                 CALL    ERRNAM          ;SET NAME OF FILE FOR TSKMON ERROR MESSAGE
002372  012700  177763                 MOV     #-15,R0         ;SET ABORT CODE
002376  000137  015356'                 JMP     USRCLS          ;ABORT OPERATION
                                ;
                                ;  See if it is a spooled device.
                                ;
002402  105737  000000G         4$:     TSTB    NSPLDV          ;Are there any spooled devices?
002406  001404                         BEQ     3$              ;Br if not
002410                                 OCALL   CHKSD           ;Are we opening to a spooled device?
002416  103074                         BCC     9$              ;Br if this is a spooled device
                                ;
                                ;  This is not a spooled device.
                                ;  See if this is a magnetic tape.
                                ;
002420  032764  000000G 000000G 3$:     BIT     #DS$NRD,DVSTAT(R4);MAG TAPE TYPE DIRECTORY DEVICE?
002426  001004                         BNE     7$              ;Br if yes
002430  032764  000000G 000000G         BIT     #DX$NRD,DVFLAG(R4);Internal flag set?
002436  001407                         BEQ     5$              ;BR IF NOT
002440  052763  000000G 000000G 7$:     BIS     #CS$ENT,C.CSW(R3);SET FLAG TO CAUSE TO CALL HANDLER ON CLOSE
002446  012702  000000G                 MOV     #DF$ENT,R2      ;GET CODE FOR ENTER
002452  000137  014212'                 JMP     SPLDIR          ;GO DO SPECIAL DEVICE ENTER
                                ;
                                ;  See if this is a file structured device.
                                ;
002456  032764  000000G 000000G 5$:     BIT     #DS$DIR,DVSTAT(R4);FILE STRUCTURED DEVICE?
002464  001451                         BEQ     9$              ;Br if not file structured
                                ;
                                ;  This is a file structured device.
                                ;  Look for a free slot large enough for the file.
                                ;
002466  013702  002044'                 MOV     ASNSIZ,R2       ;WAS SIZE SPECIFIED WITH ASSIGN STATEMENT?
002472  001002                         BNE     6$              ;BR IF YES
002474  013702  000004G                 MOV     EMTBLK+4,R2     ;GET REQUESTED SIZE FROM EMT ARG BLOCK
002500  004737  011460'         6$:     CALL    FNDFRE          ;TRY TO FIND A FREE SLOT
002504  103002                         BCC     1$              ;BR IF FOUND ONE
                                ;  Error: Can't find a free slot or protected file exists with same name.
                                ;  (Error code is returned in R0 by FNDFRE.)
002506  000137  015356'                 JMP     USRCLS
                                ;
                                ;  We have found a free slot.
                                ;  Create a tenative file entry.
                                ;
002512  010237  000000G         1$:     MOV     R2,URO          ;RETURN FILE SIZE IN USER'S R0
002516  004737  012210'                 CALL    ADDENT          ;CREATE A TENTATIVE FILE ENTRY
002522  103004                         BCC     2$              ;BR IF WE WERE SUCCESSFUL
                                ;  Error: Directory overflow -- Return error code 1.
```

.ENTER

```
     58 002524  012700  000001                MOV     #1,R0           ;SET ERROR CODE
     59 002530  000137  015356'                JMP     USRCLS          ;ABORT ENTER
     60                                ;
     61                                ;  Set up info in channel about file.
     62                                ;
     63 002534  013703  000000G        2$:     MOV     CHNADR,R3       ;ADDRESS OF CHANNEL BLOCK
     64 002540  013700  002002'                MOV     CURSEG,R0       ;CURRENT DIRECTORY SEGMENT #
     65 002544  000300                         SWAB    R0              ;POSITION SEG # TO LEFT BYTE
     66 002546  052700  000000G                BIS     #CS$ENT,R0      ;REMEMBER WE ARE DOING AN ENTER
     67 002552  050063  000000G                BIS     R0,C.CSW(R3)    ;SET UP CSW
     68 002556  010263  000000G                MOV     R2,C.LENG(R3)   ;SET UP ALLOCATED LENGTH OF FILE
     69 002562  004737  013622'                CALL    SBCALC          ;CALCULATE STARTING BLOCK # OF FILE
     70 002566  010002                         MOV     R0,R2           ;SAVE STARTING BLOCK NUMBER OF FILE
     71                                ;
     72                                ;  Give user a chance to diddle with directory entry.
     73                                ;
     74 002570  004737  013456'                CALL    DIDDLE          ;ALLOW USER TO DIDDLE WITH DIRECTORY ENTRY
     75                                ;
     76                                ;  Write out the directory segment.
     77                                ;
     78 002574  004737  014110'                CALL    WRTSEG          ;WRITE OUT UPDATED DIRECTORY SEGMENT
     79                                ;
     80                                ;  Set up file starting block number in channel
     81                                ;
     82 002600  010263  000000G                MOV     R2,C.SBLK(R3)   ;SET FILE STARTING BLOCK NUMBER IN CHANNEL
     83 002604  005063  000000G                CLR     C.USED(R3)      ;SAY NO BLOCKS WRITTEN TO FILE YET
     84                                ;
     85                                ;  Finished
     86                                ;
     87 002610  000137  015404'        9$:     JMP     USRXIT
```

```
                                        .SBTTL  .RENAME
                                ;-------------------------------------------------------------------------------
                                ;  Rename a file.
                                ;
002614  004737  007376'        RENAME: CALL    USRCOM          ;DO COMMON SETUP
                                ;
                                ;  See if this is a file structured device.
                                ;
002620  032764  000000G 000000G        BIT     #DS$DIR,DVSTAT(R4);FILE STRUCTURED DEVICE?
002626  001004                         BNE     1$              ;BR IF YES
002630  012700  000002                 MOV     #2,R0           ;INVALID OPERATION
002634  000137  015356'                JMP     USRCLS
                                ;
                                ;  Make sure we have write access to old file.
                                ;
002640  032763  000000G 000000G 1$:     BIT     #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO OLD FILE?
002646  001063                         BNE     11$             ;BR IF NOT
                                ;
                                ;  Locate directory entry with old name.
                                ;  (Note: FILSPC currently contains old file name)
                                ;
002650  004737  011376'                CALL    FNDFIL          ;LOOK UP THE ENTRY
002654  103004                         BCC     2$              ;BR IF FOUND
002656  012700  000001                 MOV     #1,R0           ;FILE NOT FOUND
002662  000137  015356'                JMP     USRCLS
                                ;  Save info about old file entry.
002666  010146                 2$:     MOV     R1,-(SP)        ;ADDRESS OF ITS DIRECTORY ENTRY
002670  010105                         MOV     R1,R5           ;CARRY ADDR OF DIR ENTRY IN R5 FOR A WHILE
002672  013746  002002'                MOV     CURSEG,-(SP)    ;SEGMENT NUMBER CONTAINING ENTRY
002676  010446                         MOV     R4,-(SP)        ;DEVICE TABLE INDEX
                                ;
                                ;  Remove old file entry from directory cache.
                                ;
002700  004737  015030'                CALL    CSHDEL          ;DELETE FILE ENTRY FROM CACHE TABLE
                                ;
                                ;  Set up new file spec.
                                ;
002704  013701  000002G                MOV     EMTBLK+2,R1     ;ADDRESS OF FILE SPEC AREA
002710  042701  000001                 BIC     #1,R1           ;MAKE SURE ADDRESS IS EVEN
002714  062701  000010                 ADD     #10,R1          ;POINT TO NEW FILE SPEC
002720  004737  007546'                CALL    GETSPC          ;MOVE DEVICE-FILE SPEC TO FILSPC
002724  004737  010226'                CALL    CHKDEV          ;MAKE SURE DEVICE IS LEGAL
002730  103006                         BCC     3$              ;BR IF OK
002732  004737  015426'                CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
002736  012700  177776                 MOV     #-2,R0          ;ILLEGAL DEVICE
002742  000137  015356'                JMP     USRCLS
                                ;
                                ;  Make sure user is authorized to access to new file.
                                ;
002746  004737  010602'        3$:     CALL    CHKACC          ;SEE IF USER IS AUTHORIZED FOR ACCESS
002752  032763  000000G 000000G        BIT     #CS$RON,C.CSW(R3);IS USER AUTHORIZED TO WRITE TO NEW FILE?
002760  001016                         BNE     11$             ;BR IF NOT
                                ;
                                ;  Make sure device and unit match that of old file.
                                ;  R4 = Device table index for new file.
                                ;  R0 = Device unit number for new file.
                                ;
```

.RENAME

```
     58 002762  020426                          CMP     R4,(SP)+        ;COMPARE DEVICE INDEX NUMBERS
     59 002764  001405                          BEQ     4$              ;BR IF OK
     60 002766  022626                  5$:     CMP     (SP)+,(SP)+    ;CLEAN OFF STACK
     61 002770  012700  000002                  MOV     #2,R0           ;INVALID OPERATION
     62 002774  000137  015356'                 JMP     USRCLS
     63 003000  120063  000000G         4$:     CMPB    R0,C.DEVQ(R3)   ;DO UNIT NUMBERS MATCH?
     64 003004  001370                          BNE     5$              ;BR IF DON'T MATCH
     65                                 ;
     66                                 ;  Make sure we have write access to new file.
     67                                 ;
     68 003006  032763  000000G 000000G        BIT     #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO NEW FILE?
     69 003014  001406                          BEQ     10$             ;BR IF YES
     70 003016  004737  015426'         11$:    CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
     71 003022  012700  177763                  MOV     #-15,R0         ;SET ABORT CODE
     72 003026  000137  015356'                 JMP     USRCLS          ;ABORT THE OPERATION
     73                                 ;
     74                                 ;  Delete any existing file with new file name.
     75                                 ;
     76                                 ;  Temporarily mark old file entry as empty so we won't find it when
     77                                 ;  searching for file that has same name as new file.
     78                                 ;  Note that this is safe since the USR is locked and noone else can
     79                                 ;  access the directory.
     80 003032  042765  000000G 000000G 10$:    BIC     #FS$PRM,FD$STA(R5);CLEAR PERMANENT-FILE STATUS FLAG
     81 003040  052765  000000G 000000G        BIS     #FS$EMP,FD$STA(R5);SET EMPTY-FILE STATUS FLAG
     82 003046  023727  002002' 000001        CMP     CURSEG,#1       ;IS THIS DIR ENTRY IN SEGMENT # 1?
     83 003054  001402                          BEQ     12$             ;BR IF YES -- IT WILL NOT BE REREAD BY FNDFIL
     84 003056  004737  014110'                 CALL    WRTSEG          ;WRITE OUT MODIFIED ENTRY SO FNDFIL WILL READ IT
     85                                 ;  Try to locate directory entry for file being deleted.
     86                                 ;  (Note: FILSPC now contains new file name)
     87 003062  004737  011376'         12$:    CALL    FNDFIL          ;LOOKUP NEW FILE NAME
     88 003066  103442                          BCS     9$              ;BR IF DON'T NEED TO DELETE
     89                                 ;
     90                                 ;  File with new name already exists.
     91                                 ;  See if it is protected.
     92                                 ;
     93 003070  032761  000000G 000000G        BIT     #FS$PRO,FD$STA(R1);IS FILE PROTECTED?
     94 003076  001420                          BEQ     13$             ;BR IF NOT
     95                                 ;  File is protected so we cannot delete it.
     96                                 ;  Restore file status word for old file then return error code 3 for rename.
     97 003100  012600                          MOV     (SP)+,R0        ;GET DIR SEG # FOR OLD FILE ENTRY
     98 003102  004737  013772'                 CALL    RDSEG           ;READ IN THE SEGMENT
     99 003106  012601                          MOV     (SP)+,R1        ;GET ADDRESS OF DIR ENTRY IN SEGMENT
    100 003110  042761  000000G 000000G        BIC     #FS$EMP,FD$STA(R1);CLEAR EMPTY-FILE FLAG
    101 003116  052761  000000G 000000G        BIS     #FS$PRM,FD$STA(R1);SET PERMANENT-FILE FLAG
    102 003124  004737  014110'                 CALL    WRTSEG          ;REWRITE THE DIR SEGMENT
    103 003130  012700  000003                  MOV     #3,R0           ;RETURN ERROR CODE 3
    104 003134  000137  015356'                 JMP     USRCLS
    105                                 ;  File is not protected.
    106                                 ;  Remove its entry from directory cache.
    107 003140  004737  015030'         13$:    CALL    CSHDEL          ;DELETE FILE ENTRY FROM CACHE TABLE
    108                                 ;  Mark its entry as empty.
    109 003144  012761  000000G 000000G        MOV     #FS$EMP,FD$STA(R1);FILE IS NOW DELETED
    110                                 ;  See if this entry is in same directory segment as the file being renamed.
    111                                 ;  If it is, bypass the consolidation & rewrite.
    112 003152  023716  002002'                 CMP     CURSEG,(SP)     ;IS DELETED FILE ENTRY IN SAME SEG AS RENAMED ENTRY?
    113 003156  001002                          BNE     6$              ;BR IF NOT
    114 003160  012600                          MOV     (SP)+,R0        ;POP SEGMENT NUMBER
```

```
                        BR      7$              ;GO FILL IN NEW NAME
                ;  Consolidate & rewrite segment with deleted file entry.
003164  004737  013016' 6$:    CALL    CONSOL          ;CONSOLIDATE DIRECTORY SEGMENT
003170  004737  014110'        CALL    WRTSEG          ;WRITE IT OUT
                ;
                ;  Now change the name in the old file entry.
                ;
003174  012600          9$:    MOV     (SP)+,R0        ;GET DIRECTORY SEGMENT #
003176  004737  013772'        CALL    RDSEG           ;READ IN THE SEGMENT
003202  012601          7$:    MOV     (SP)+,R1        ;GET ADDRESS OF ENTRY WITHIN SEGMENT
003204  042761  000000G 000000G BIC     #FS$EMP,FD$STA(R1);CLEAR EMPTY-FILE FLAG
003212  052761  000000G 000000G BIS     #FS$PRM,FD$STA(R1);SET PERMANENT-FILE FLAG
003220  010105                  MOV     R1,R5           ;GET ADDRESS OF DIR ENTRY BEING RENAMED
003222  062705  000000G         ADD     #FD$NAM,R5      ;POINT TO FILE NAME AREA IN ENTRY
003226  012702  000002G         MOV     #FILSPC+2,R2    ;POINT TO NEW FILE NAME
003232  012225                  MOV     (R2)+,(R5)+     ;MOVE IN NEW NAME
003234  012225                  MOV     (R2)+,(R5)+
003236  011215                  MOV     (R2),(R5)
                ;
                ;  Give user a chance to diddle with the dir entry if he wishes.
                ;
003240  004737  013456'         CALL    DIDDLE          ;ALLOW USER TO DIDDLE
                ;
                ;  Add new file entry to cache table.
                ;
003244  004737  014700'         CALL    CSHADD          ;ADD NEW FILE ENTRY TO CACHE TABLE
                ;
                ;  Consolidate and rewrite directory segment.
                ;
003250  004737  013016'         CALL    CONSOL          ;CONSOLIDATE DIRECTORY SEGMENT
003254  004737  014110'         CALL    WRTSEG          ;WRITE IT OUT
                ;
                ;  Finished
                ;
003260  005063  000000G         CLR     C.CSW(R3)       ;PURGE THE CHANNEL
003264  000137  015404'         JMP     USRXIT
```

```
                                        .SBTTL  .DELETE
                                ;-------------------------------------------------------------------------------
                                ;  Delete a permanent file entry.
                                ;
003270  004737  007376'         DELETE: CALL    USRCOM          ;DO COMMON SETUP
                                ;
                                ;  See if we have write access to file being deleted.
                                ;
003274  032763  000000G 000000G         BIT     #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO FILE?
003302  001406                        BEQ     2$              ;BR IF YES
003304  004737  015426'               CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
003310  012700  177763                MOV     #-15,R0         ;SET ABORT ERROR CODE
003314  000137  015356'               JMP     USRCLS          ;ABORT THE OPERATION
                                ;
                                ;  See if device is a magnetic tape.
                                ;
003320  032764  000000G 000000G 2$:     BIT     #DS$NRD,DVSTAT(R4);IS THIS A MAG TAPE DEVICE?
003326  001004                        BNE     6$              ;Br if yes
003330  032764  000000G 000000G       BIT     #DX$NRD,DVFLAG(R4);Internal flag set?
003336  001404                        BEQ     5$              ;BR IF NOT
003340  012702  000000G         6$:     MOV     #DF$DEL,R2      ;SET DELETE FUNCTION CODE
003344  000137  014212'               JMP     SPLDIR          ;DO SPECIAL DEVICE DELETE
                                ;
                                ;  See if this is a file structured device.
                                ;
003350  032764  000000G 000000G 5$:     BIT     #DS$DIR,DVSTAT(R4);IS THIS A DIRECTORY STRUCTURED DEVICE?
003356  001004                        BNE     1$              ;BR IF YES
                                ;
                                ;  Delete is invalid on non-file-structured device.
                                ;
003360  012700  000002                MOV     #2,R0           ;RETURN INVALID-OPERATION ERROR CODE
003364  000137  015356'               JMP     USRCLS
                                ;
                                ;  Try to locate directory entry for file being deleted.
                                ;
003370  004737  011376'         1$:     CALL    FNDFIL          ;LOOK UP THE FILE
003374  103425                        BCS     9$              ;BR IF FILE DOES NOT EXIST
                                ;
                                ;  See if file is protected.
                                ;
003376  032761  000000G 000000G       BIT     #FS$PRO,FD$STA(R1);IS FILE PROTECTED?
003404  001404                        BEQ     3$              ;BR IF NOT
003406  012700  000003                MOV     #3,R0           ;RETURN ERROR CODE 3 IF FILE IS PROTECTED
003412  000137  015356'               JMP     USRCLS
                                ;
                                ;  Remove file entry from directory cache.
                                ;
003416  004737  015030'         3$:     CALL    CSHDEL          ;REMOVE FILE ENTRY FROM DIRECTORY CACHE
                                ;
                                ;  File exists.  Mark its entry as empty.
                                ;
003422  012761  000000G 000000G       MOV     #FS$EMP,FD$STA(R1);MARK DIR ENTRY AS EMPTY
                                ;
                                ;  Consolidate and rewrite the directory segment.
                                ;
003430  004737  013016'               CALL    CONSOL          ;CONSOLIDATE THE SEGMENT
003434  004737  014110'               CALL    WRTSEG          ;WRITE SEGMENT TO DISK
```

.DELETE

```
                                        ;
                                        ;  Finished
                                        ;
003440  005063  000000G                       CLR     C.CSW(R3)       ;PURGE THE CHANNEL
003444  000137  015404'                       JMP     USRXIT
                                        ;
                                        ;  File does not exist.
                                        ;
003450  012700  000001          9$:           MOV     #1,R0           ;FILE DOES NOT EXIST
003454  000137  015356'                       JMP     USRCLS
```

```
                                        .SBTTL  .CLOSE
                                ;-------------------------------------------------------------------------------
                                ;  Close a channel.
                                ;
003460  013703  000000G         CLOSE:  MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
003464  032763  000000G 000000G         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
003472  001002                          BNE     7$              ;BR IF CHANNEL IS OPEN
003474  000137  004204'                 JMP     8$              ;NOP IF NOT OPEN
                                ;
                                ;  Wait for all I/O to finish on this channel
                                ;
003500  004737  000000G         7$:     CALL    IOWAIT          ;WAIT FOR ALL I/O ON CHANNEL TO FINISH
                                ;
                                ;  See if we are closing a spooled device.
                                ;
003504  032763  000000G 000000G         BIT     #CS$SPL,C.CSW(R3);IS THIS A SPOOLED DEVICE CHANNEL?
003512  001405                          BEQ     1$              ;BR IF NOT
003514                                  OCALL   SDCLOS          ;CLOSE A SPOOLED DEVICE CHANNEL
003522  000137  004200'                 JMP     9$
                                ;
                                ;  See if device being closed is a magnetic tape.
                                ;
003526  016300  000000G         1$:     MOV     C.CSW(R3),R0    ;GET CSW
003532  042700  000000C                 BIC     #^C<CS$NMX>,R0  ;MASK OUT ALL BUT DEVICE TABLE INDEX
003536  032760  000000G 000000G         BIT     #DS$NRD,DVSTAT(R0);IS DEVICE A MAG TAPE?
003544  001004                          BNE     18$             ;Br if yes
003546  032760  000000G 000000G         BIT     #DX$NRD,DVFLAG(R0);Internal flag set?
003554  001404                          BEQ     6$              ;BR IF NOT
003556  012702  000000G         18$:    MOV     #DF$CLS,R2      ;DO SPECIAL DEVICE CLOSE
003562  000137  014212'                 JMP     SPLDIR
                                ;
                                ;  See if we are closing a shared file.
                                ;
003566  005737  000000G         6$:     TST     JCDB            ;Does job have any shared file chans open?
003572  001402                          BEQ     17$             ;BR IF NOT
003574  004777  000000G                 CALL    @SFCLS          ;CHECK FOR CLOSING A SHARED FILE
                                ;
                                ;  See if this file was opened with a .LOOKUP or .ENTER
                                ;
003600  032763  000000G 000000G 17$:     BIT     #CS$ENT,C.CSW(R3);DID WE DO A .ENTER ON THE CHANNEL?
003606  001574                          BEQ     9$              ;BR IF NOT
                                ;
                                ;  We are closing a new file on a directory structured device.
                                ;  Gain exclusive access to USR data base.
                                ;
003610  004737  010004'                 CALL    GETUSR          ;CLAIM EXCLUSIVE ACCESS TO USR
                                ;
                                ;  Set up device # and unit # in FILDVU.
                                ;
003614  116337  000000G 002015'         MOVB    C.DEVQ(R3),FILDVU+1;SET UNIT #
003622  016300  000000G                 MOV     C.CSW(R3),R0    ;GET CHANNEL STATUS WORD
003626  042700  000000C                 BIC     #^C<CS$NMX>,R0  ;MASK OUT ALL BUT DEVICE INDEX #
003632  110037  002014'                 MOVB    R0,FILDVU       ;SET DEVICE INDEX #
                                ;
                                ;  Set up info about this directory
                                ;
003636  005063  000000G                 CLR     C.SBLK(R3)      ;SAY CHANNEL BEING USED FOR DIRECTORY OP
```

.CLOSE

```
     58 003642  042763  000000C 000000G          BIC     #<CS$EOF!CS$ERR>,C.CSW(R3) ;IGNORE PRIOR ERROR
     59 003650  004737  013656'                  CALL    RDSEG1                  ;READ SEG1 AND SET DIRSIZ
     60                                  ;
     61                                  ;  Find the tentative file entry.
     62                                  ;
     63 003654  016300  000000G                  MOV     C.CSW(R3),R0            ;GET CHANNEL STATUS WORD
     64 003660  042700  160377                   BIC     #^C<17400>,R0           ;GET DIRECTORY SEGMENT # WITH TENTATIVE ENTRY
     65 003664  000300                           SWAB    R0                      ;RIGHT-JUSTIFY SEG #
     66 003666  004737  013772'                  CALL    RDSEG                   ;READ IN THAT SEGMENT
     67 003672  013702  000000G                  MOV     CHNNUM,R2               ;GET OUR CHANNEL #
     68 003676  012700  000000G          2$:     MOV     #FS$TEN,R0              ;FIND NEXT TENTATIVE FILE ENTRY
     69 003702  004737  013370'                  CALL    GETDIR
     70 003706  103545                           BCS     11$                     ;BR IF CAN'T FIND TENTATIVE FILE ENTRY
     71 003710  120261  000000G                  CMPB    R2,FD$CHN(R1)           ;IS THIS ENTRY FOR RIGHT CHANNEL?
     72 003714  001370                           BNE     2$                      ;BR IF NOT
     73 003716  123761  000000G 000000G          CMPB    CORUSR,FD$JOB(R1);IS THIS FILE FOR RIGHT JOB?
     74 003724  001364                           BNE     2$                      ;BR IF NOT
     75                                  ;  Found the tentative file entry.
     76                                  ;  Save the file name in FILSPC.
     77 003726  010102                           MOV     R1,R2                   ;ADDRESS OF TENTATIVE FILE ENTRY
     78 003730  062702  000000G                  ADD     #FD$NAM,R2              ;POINT TO NAME IN ENTRY
     79 003734  012704  000002G                  MOV     #FILSPC+2,R4            ;SAVE THE NAME HERE
     80 003740  012224                           MOV     (R2)+,(R4)+             ;SAVE FILE NAME IN FILSPC
     81 003742  012224                           MOV     (R2)+,(R4)+
     82 003744  011214                           MOV     (R2),(R4)
     83                                  ;  Remember the position of the tentative file entry.
     84 003746  010146                           MOV     R1,-(SP)
     85 003750  013746  002002'                  MOV     CURSEG,-(SP)
     86                                  ;
     87                                  ;  Delete any permanent file entry that has the same name as the new file.
     88                                  ;
     89 003754  004737  011376'                  CALL    FNDFIL                  ;SEARCH FOR PERM FILE ENTRY WITH THIS NAME
     90 003760  103416                           BCS     5$                      ;BR IF NO PERM FILE ENTRY WITH SAME NAME
     91                                  ;  Delete old file entry from directory cache.
     92 003762  004737  015030'                  CALL    CSHDEL                  ;DELETE OLD FILE ENTRY FROM DIRECTORY CACHE
     93                                  ;  Mark old file as deleted.
     94 003766  012761  000000G 000000G          MOV     #FS$EMP,FD$STA(R1);CHANGE OLD FILE ENTRY TO BE EMPTY
     95                                  ;  Consolidate and rewrite old segment unless it is same as one with new entry
     96 003774  023716  002002'                  CMP     CURSEG,(SP)             ;IS OLD SEG SAME AS NEW ONE?
     97 004000  001002                           BNE     3$                      ;BR IF NOT
     98 004002  005726                           TST     (SP)+                   ;POP NEW SEG #
     99 004004  000407                           BR      4$
    100 004006  004737  013016'          3$:     CALL    CONSOL                  ;CONSOLIDATE THE OLD SEGMENT
    101 004012  004737  014110'                  CALL    WRTSEG                  ;WRITE OUT THE OLD SEGMENT
    102 004016  012600                  5$:     MOV     (SP)+,R0                ;GET # OF NEW SEGMENT
    103 004020  004737  013772'                  CALL    RDSEG                   ;READ IT IN
    104                                 ;
    105                                 ;  Convert tentative file entry to permanent.
    106                                 ;
    107 004024  012601                  4$:     MOV     (SP)+,R1                ;GET ADDRESS OF TENTATIVE FILE ENTRY
    108 004026  032761  000000G 000000G          BIT     #FS$TEN,FD$STA(R1);MAKE SURE WE'RE POINTING TO TENTATIVE ENTRY
    109 004034  001506                           BEQ     15$                     ;BAD ERROR IF NOT
    110 004036  012761  000000G 000000G          MOV     #FS$PRM,FD$STA(R1);SET ITS TYPE TO PERMANENT FILE
    111 004044  016102  000000G                  MOV     FD$LEN(R1),R2           ;GET ALLOCATED SIZE OF FILE
    112 004050  020263  000000G                  CMP     R2,C.LENG(R3)           ;MAKE SURE FILE LENGTHS AGREE
    113 004054  001065                           BNE     12$                     ;BAD ERROR IF NOT
    114 004056  016300  000000G                  MOV     C.USED(R3),R0           ;GET # BLOCKS ACTUALLY USED IN FILE
```

.CLOSE

```
                                        MOV     RO,FD$LEN(R1)   ;SET THIS AS THE ACTUAL FILE LENGTH
                                        CMP     RO,R2           ;MAKE SURE HIGHEST BLOCK NOT ABOVE FILE LENGTH
                                        BHI     13$             ;ERROR IF TOO BIG
                                        SUB     RO,R2           ;GET # BLOCKS LEFT OVER
                                        ADD     DIRSIZ,R1       ;POINT TO EMPTY FILE ENTRY THAT FOLLOWS
                                        BIT     #FS$EMP,FD$STA(R1);MAKE SURE THIS REALLY IS AN EMPTY ENTRY
                                        BEQ     14$             ;BAD ERROR IF NOT
                                        ADD     R2,FD$LEN(R1)   ;ADD RESIDUAL BLOCKS TO EMPTY ENTRY
                                ;  Set date and time of file creation in file directory entry.
                                        SUB     DIRSIZ,R1       ;POINT BACK TO NEW DIRECTORY ENTRY
                                        CLR     FD$TIM(R1)      ;SAY CREATION TIME UNKNOWN
                                        TST     FD$DAT(R1)      ;DID DIDDLE ROUTINE SET FILE DATE?
                                        BNE     16$             ;BR IF YES
                                        MOV     SYSDAT,FD$DAT(R1);SET DATA THAT FILE WAS CREATED (CLOSED)
                                        MOV     SYTIMH,R4       ;GET CURRENT TIME OF DAY (HIGH ORDER)
                                        MOV     SYTIML,R5       ;GET LOW-ORDER TIME OF DAY
                                        DIV     TK3SVL,R4       ;CONVERT TO 3-SECOND UNITS
                                        MOV     R4,FD$TIM(R1)   ;SET TIME OF FILE CREATION
                                ;  Add entry for new file to directory cache.
                                16$:    CALL    CSHADD          ;ADD TO CACHE TABLE
                                ;  Consolidate and rewrite the directory segment.
                                        CALL    CONSOL          ;CONSOLIDATE THE DIRECTORY
                                        CALL    WRTSEG          ;WRITE OUT THE SEGMENT
                                ;
                                ;  Free the USR
                                ;
                                        CALL    FREUSR          ;RELEASE USR DATA BASE
                                ;
                                ;  Mark the channel as closed.
                                ;
                                9$:     CLR     C.CSW(R3)       ;SAY THE CHANNEL IS CLOSED
                                ;
                                ;  Finished, check for non-fatal errors
                                ;
                                8$:     MOVB    CLZERR,RO       ;Do we need to report .CLOSZ error?
                                        BEQ     10$             ;Br if not
                                        JMP     SETERR          ;If so, report error to user
                                ;
                                ;  Successful completion
                                ;
                                10$:    JMP     EMTXIT
                                ;
                                ;  Consistency check errors -- May indicate hardware errors or bugs.
                                ;
                                ;  Can't find tentative file during close.
                                11$:    MOV     #UERR1,RO
                                        BR      20$
                                ;  File size in channel block doesn't agree with size in directory entry.
                                12$:    MOV     #UERR2,RO
                                        BR      20$
                                ;  Highest block # written is greater than file length.
                                13$:    MOV     #UERR3,RO
                                        BR      20$
                                ;  Empty file entry does not follow tentative file entry.
                                14$:    MOV     #UERR4,RO
                                        BR      20$
                                ;  Not pointing to tentative file entry during close.
```

.CLOSE

```
    172 004252  012700  177756          15$:    MOV     #UERR5,R0
    173 004256  000137  015414'         20$:    JMP     UABORT          ;GIVE FATAL ABORT
```

```
                                        .SBTTL  .DSTATUS
                                ;-------------------------------------------------------------------
                                ;  .DSTATUS -- Determine device status.
                                ;
004262  004737  010004'         DSTAT:  CALL    GETUSR          ;GET EXCLUSIVE ACCESS TO USR DATA BASE
                                ;  Move device name to FILSPC buffer and perform any assigns.
004266  013701  000000G                 MOV     URO,R1          ;POINT TO DEVICE SPEC
004272  004737  007546'                 CALL    GETSPC          ;MOVE TO FILSPC BUFFER
                                ;  Look up device in perm name table.
004276  004737  010226'                 CALL    CHKDEV          ;LOOK UP THE DEVICE NAME
004302  103003                         BCC     1$              ;BR IF FOUND
                                ;  Invalid device name
004304  005000                         CLR     R0              ;SET ERROR CODE
004306  000137  015370'                 JMP     USRERR
                                ;
                                ;  Valid device.
                                ;  Return info about it.
                                ;  (R4 now has device table index)
                                ;
004312  010005                 1$:     MOV     R0,R5           ;CARRY UNIT NUMBER IN R5
004314  013700  000004G                 MOV     EMTBLK+4,R0     ;GET POINTER TO USER'S RESULT AREA
004320  010037  000000G                 MOV     R0,URO          ;RETURN POINTER TO RESULT IN R0
004324  004737  000000G                 CALL    VALADW          ;VALIDATE THE ADDRESS
004330  016446  000000G                 MOV     DVSTAT(R4),-(SP);DEVICE STATUS WORD
004334  106620                         MTPD    (R0)+
004336  016446  000000G                 MOV     HANSIZ(R4),-(SP);HANDLER SIZE
004342  106620                         MTPD    (R0)+
                                ;
004344  016446  000000G                 MOV     HANENT(R4),-(SP);HANDLER ENTRY POINT
004350  106620                         MTPD    (R0)+
004352  016446  000000G                 MOV     DEVSIZ(R4),-(SP);DEVICE SIZE
004356  106620                         MTPD    (R0)+
                                ;
                                ;  If this is a logical disk, return file size as device size
                                ;
004360  020437  000000G                 CMP     R4,LDDEVX       ;IS THIS A LOGICAL DISK?
004364  001004                         BNE     2$              ;BR IF NOT
004366  006305                         ASL     R5              ;CONVERT UNIT # TO WORD TABLE INDEX
004370  016546  000000G                 MOV     LDSIZE(R5),-(SP);GET LOGICAL DISK SIZE
004374  106640                         MTPD    -(R0)           ;PASS TO USER
                                ;
                                ;  Finished
                                ;
004376  000137  015404'         2$:     JMP     USRXIT          ;FINISHED
```

```
                                        .SBTTL  .FETCH
                                ;-------------------------------------------------------------------------
                                ;  .FETCH EMT, This is basically a NOP under TSX-Plus since all device
                                ;  handlers are permanently resident.  However, we do check to make sure
                                ;  the device is installed.
                                ;
004402  004737  010004'         FETCH:  CALL    GETUSR          ;Get exclusive access to USR data base
                                ;
                                ;  Move device name to FILSPC buffer and perform any assigns
                                ;
004406  013701  000000G                 MOV     URO,R1          ;Point to device spec
004412  004737  007546'                 CALL    GETSPC          ;Move to filspc buffer
                                ;
                                ;  Look up device in perm name table
                                ;
004416  004737  010226'                 CALL    CHKDEV          ;Look up the device name
004422  103003                         BCC     1$              ;Br if found
                                ;
                                ;  Invalid device
                                ;
004424  005000                         CLR     R0              ;Set error code = 0
004426  000137  015370'                 JMP     USRERR          ;Return with error
                                ;
                                ;  The device name is valid.
                                ;  Return handler load address to user in R0.
                                ;
004432  013700  000004G         1$:     MOV     EMTBLK+4,R0     ;Get user's load address
004436  005200                         INC     R0              ;Round up
004440  042700  000001                 BIC     #1,R0           ;And force to even address
004444  010037  000000G                 MOV     R0,URO          ;And return as top of handler
004450  000137  015404'                 JMP     USRXIT          ;Exit from EMT
```

```
                                        .SBTTL  .SFDAT, .SFTIM, & .FPROT
                                ;-------------------------------------------------------------------------
                                ;  .SFDAT  --  Set date in file entry
                                ;
004454  004737  004720'         SFDATE: CALL    SFCOM           ;LOCATE FILE ENTRY
004460  004737  004770'                 CALL    SFWRIT          ;MAKE SURE WE HAVE WRITE ACCESS
004464  013700  000004G                 MOV     EMTBLK+4,R0     ;GET SPECIFIED DATE VALUE
004470  001002                         BNE     1$              ;BR IF DATE SPECIFIED
004472  013700  000000G                 MOV     SYSDAT,R0       ;USE CURRENT DATE
004476  016105  000000G         1$:     MOV     FD$DAT(R1),R5   ;SAVE OLD FILE DATE
004502  010061  000000G                 MOV     R0,FD$DAT(R1)   ;SET NEW DATE IN FILE ENTRY
                                ;
                                ;  If PIP is doing a .SFDAT, set the time entry in the file directory
                                ;  to the time from the file last looked up.
                                ;  This is done to preserve both date and time for a file across
                                ;  a PIP copy operation.
                                ;
                                ;       MOVB    CORUSR,R4       ;GET CURRENT JOB INDEX NUMBER
                                ;       BIT     #$PIPRN,LSW6(R4);IS PIP RUNNING?
                                ;       BEQ     SFEXIT          ;BR IF NOT
                                ;       CMP     R0,LSTFDD       ;ARE WE USING THE OLD FILE'S DATE?
                                ;       BNE     SFEXIT          ;BR IF NOT
                                ;       MOV     LSTFDT,FD$TIM(R1);USE TIME VALUE FROM LAST .LOOKUP
                                ;
                                ;  Add new file entry to directory cache
                                ;
004506  004737  014700'         SFEXIT: CALL    CSHADD          ;ADD ENTRY TO CACHE
                                ;
                                ;  Rewrite directory segment
                                ;
004512  004737  014110'                 CALL WRTSEG             ;REWRITE DIRECTORY SEGMENT
                                ;
                                ;  Finished
                                ;
004516  005063  000000G                 CLR     C.CSW(R3)       ;SAY CHANNEL IS CLOSED
004522  000137  015404'                 JMP     USRXIT          ;EXIT

                                ;-------------------------------------------------------------------------
                                ;  .SFTIM -- Set file time in file entry
                                ;
004526  004737  004720'         SFTIME: CALL    SFCOM           ;LOCATE FILE ENTRY
004532  004737  004770'                 CALL    SFWRIT          ;MAKE SURE WE HAVE WRITE ACCESS
004536  013761  000004G 000000G        MOV     EMTBLK+4,FD$TIM(R1) ;SET TIME VALUE IN DIRECTORY ENTRY
004544  000760                         BR      SFEXIT          ;REWRITE DIRECTORY SEGMENT

                                ;-------------------------------------------------------------------------
                                ;  .FPROT -- Set file protection
                                ;
004546  004737  004720'         SFPROT: CALL    SFCOM           ;DO COMMON SETUP
004552  004737  004770'                 CALL    SFWRIT          ;MAKE SURE WE HAVE WRITE ACCESS
004556  113700  000004G                 MOVB    EMTBLK+4,R0     ;GET PROTECT/UNPROTECT FLAG
004562  001407                         BEQ     2$              ;BR IF UNPROTECT WANTED
004564  020027  000001                 CMP     R0,#1           ;VALUE MUST BE 0 OR 1
004570  001410                         BEQ     3$              ;BR IF PROTECT WANTED
004572  012700  000003                 MOV     #3,R0           ;ERROR IF NOT 0 OR 1
004576  000137  015356'                 JMP     USRCLS
004602  042761  000000G 000000G 2$:     BIC     #FS$PRO,FD$STA(R1) ;UNPROTECT THE FILE
```

```
     58 004610  000736                          BR      SFEXIT           ;GO EXIT
     59 004612  052761  000000G 000000G 3$:     BIS     #FS$PRO,FD$STA(R1) ;PROTECT THE FILE
     60 004620  000732                          BR      SFEXIT
```

```
                                        .SBTTL  .GFINF -- Get information about a file
                                ;-------------------------------------------------------------------------
                                ;  .GFINF Get the following information about a file:
                                ;    1.  Size of the file.
                                ;    2.  Protected/Unprotected status.
                                ;    3.  Date of creation.
                                ;    4.  Time of creation.
                                ;    5.  Starting block number of file.
                                ;
004622  004737  004720'        GFINFO: CALL    SFCOM           ;LOCATE FILE ENTRY
004626  013700  000004G                MOV     EMTBLK+4,R0     ;GET ADDRESS OF USER'S INFO BUFFER
004632  004737  000000G                CALL    VALADW          ;MAKE SURE ADDRESS IS VALID
                                ;  File size
004636  016146  000000G                MOV     FD$LEN(R1),-(SP);FILE SIZE
004642  106620                        MTPD    (R0)+
                                ;  Protected/Unprotected status
004644  005046                        CLR     -(SP)           ;ASSUME FILE IS NOT PROTECTED
004646  032761  000000G 000000G       BIT     #FS$PRO,FD$STA(R1);IS FILE PROTECTED?
004654  001401                        BEQ     1$              ;BR IF NOT
004656  005216                        INC     (SP)            ;RETURN 1 IF FILE IS PROTECTED
004660  106620                1$:     MTPD    (R0)+
                                ;  Creation date
004662  016146  000000G                MOV     FD$DAT(R1),-(SP);CREATION DATE
004666  106620                        MTPD    (R0)+
                                ;  Creation time
004670  016146  000000G                MOV     FD$TIM(R1),-(SP);CREATION TIME
004674  106620                        MTPD    (R0)+
                                ;  Starting block number
004676  010002                        MOV     R0,R2           ;SAVE BUFFER POINTER
004700  004737  013622'                CALL    SBCALC          ;CALCULATE STARTING BLOCK NUMBER
004704  010046                        MOV     R0,-(SP)
004706  106622                        MTPD    (R2)+           ;RETURN STARTING BLOCK NUMBER
                                ;
                                ;  Finished
                                ;
004710  005063  000000G               CLR     C.CSW(R3)       ;SAY CHANNEL IS CLOSED
004714  000137  015404'               JMP     USRXIT          ;EXIT
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Common setup routine used by SFDATE and SFPROT routines.
                                ;  The USR entry setup is done including moving the file spec and checking
                                ;  to see that the channel is open.
                                ;  The directory entry is found.
                                ;  If any errors are detected, an error return from the EMT is taken.
                                ;
                                ;  Outputs:
                                ;   R1 = Address of directory entry for file.
                                ;   R3 = Address of CSW for channel.
                                ;   R4 = Index into device table for device being opened.
                                ;
004720  004737  007376'         SFCOM:  CALL    USRCOM          ;DO USR ENTRY SETUP
                                ;
                                ;  Make sure this is a file structured device and we have write
                                ;  access to it.
                                ;
004724  032764  000000G 000000G         BIT     #DS$DIR,DVSTAT(R4) ;IS THIS A FILE STRUCTURED DEVICE?
004732  001004                        BNE     1$              ;BR IF YES
004734  012700  000002              MOV     #2,R0           ;RETURN ERROR CODE 2 IF NOT
004740  000137  015356'              JMP     USRCLS
                                ;
                                ;  Locate directory entry for the file
                                ;
004744  004737  011376'         1$:     CALL    FNDFIL          ;LOCATE THE DIRECTORY ENTRY
004750  103004                        BCC     3$              ;BR IF ENTRY FOR FILE FOUND
004752  012700  000001              MOV     #1,R0           ;ERROR 1 IF CAN'T FIND ENTRY
004756  000137  015356'              JMP     USRCLS
                                ;
                                ;  Remove directory entry from cache
                                ;
004762  004737  015030'         3$:     CALL    CSHDEL          ;REMOVE ENTRY FROM CACHE
                                ;
                                ;  Finished
                                ;
004766  000207                  1$:     RETURN
                                ;-------------------------------------------------------------------------------
                                ;  Make sure that we have write access to a file.
                                ;  Produce an abort if not.
                                ;
                                ;  Inputs:
                                ;   R3 = Address of CSW for channel opened to the file.
                                ;
004770  032763  000000G 000000G SFWRIT: BIT     #CS$RON,C.CSW(R3) ;DO WE HAVE WRITE ACCESS TO THE FILE?
004776  001406                  1$:     BEQ     1$              ;BR IF YES
005000  004737  015426'              CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
005004  012700  177763              MOV     #-15,R0         ;ERROR IF NOT
005010  000137  015356'              JMP     USRCLS
005014  000207                  1$:     RETURN                  ;SUCCESSFUL RETURN
```

```
                        .SBTTL  ALCEMT -- Allocate a device
                ;*****************************************************************************
                ;  The ALLOCATE EMT is used to allocate a device for exclusive access by
                ;  the current job.
                ;
005016          ALCEMT:
                ;
                ;  Determine if this is an Allocate, Deallocate, or test allocation EMT
                ;
005016  113700  000000G         MOVB    EMTBLK,R0       ;Get sub-function code
005022  001410                  BEQ     10$             ;0==>Allocate
005024  120027  000001          CMPB    R0,#1           ;1==>Deallocate
005030  001475                  BEQ     DLCEMT
005032  120027  000002          CMPB    R0,#2           ;2==>Test allocation
005036  001567                  BEQ     TLCEMT
005040  000137  000000G         JMP     BADEMT          ;Invalid sub-function code
                ;
                ;  Allocate a device
                ;
005044  032737  000000G 000000G 10$:    BIT     #PO$ALC,PRIVCO  ;Are we authorized to allocate devices?
005052  001004                  BNE     11$             ;Br if yes
005054  012700  000005          MOV     #5,R0           ;Error code 5 if not authorized
005060  000137  000000G         JMP     SETERR
                ;
                ;  Do common allocate setup
                ;
005064  004737  005636'         11$:    CALL    ALCCOM          ;Do common setup
                ;
                ;  See if the device being allocated is currently allocated to another job
                ;
005070  004737  005436'         CALL    ALCTST          ;Test for allocation to another job
                ;
                ;  Device is not currently allocated.
                ;  Check to see if the device is already allocated by our job or a
                ;  related job.
                ;
005074  012702  000000G         MOV     #ALCTBL,R2      ;Point to start of allocation table
005100  013700  002014'         MOV     FILDVU,R0       ;Get device # and index #
005104  020062  000000G 4$:     CMP     R0,AD$DVU(R2)   ;Search for device in allocation table
005110  001406                  BEQ     5$              ;Br if found it
005112  062702  000000G         ADD     #AD$$SZ,R2      ;Point to next entry
005116  020227  000000G         CMP     R2,#ALCEND      ;Checked all entries?
005122  103770                  BLO     4$              ;Br if not
005124  000410                  BR      7$              ;Br if not currently in allocation table
                ;
                ;  Device is already in allocation table
                ;
005126  116200  000000G 5$:     MOVB    AD$JOB(R2),R0   ;Get # of job that owns device
005132  120160  000000G         CMPB    R1,LNPRIM(R0)   ;Is primary job reallocating device?
005136  001030                  BNE     9$              ;Br if not
005140  110162  000000G         MOVB    R1,AD$JOB(R2)   ;Let primary job take over device
005144  000425                  BR      9$
                ;
                ;  Device is not currently allocated.
                ;  Search for a free entry in the allocation table.
                ;
005146  012702  000000G 7$:     MOV     #ALCTBL,R2      ;Point to start of allocation table
```

```
     58 005152  105762  000000G          2$:     TSTB    AD$JOB(R2)          ;Is this entry free?
     59 005156  001411                           BEQ     3$                  ;Br if yes
     60 005160  062702  000000G                  ADD     #AD$$SZ,R2          ;Point to next table entry
     61 005164  020227  000000G                  CMP     R2,#ALCEND          ;Checked all entries?
     62 005170  103770                           BLO     2$                  ;Loop if more to check
     63 005172  012700  000003                   MOV     #3,R0               ;Error 3 -- Allocation table is full
     64 005176  000137  015370'                  JMP     USRERR              ;Error abort
     65                                     ;
     66                                     ;  Found a free entry, set it up for this device
     67                                     ;
     68 005202  013762  002014' 000000G  3$:     MOV     FILDVU,AD$DVU(R2);Store device and unit numbers
     69 005210  110162  000000G                  MOVB    R1,AD$JOB(R2)       ;Store job number
     70 005214  105062  000000G                  CLRB    AD$FLG(R2)          ;No flags yet
     71                                     ;
     72                                     ;  Finished
     73                                     ;
     74 005220  000137  015404'          9$:     JMP     USRXIT
```

```
                                        .SBTTL  DLCEMT -- Deallocate a device
                                ;-------------------------------------------------------------------------
                                ;  Deallocate a device.
                                ;
005224  032737  000000G 000000G DLCEMT: BIT     #PO$ALC,PRIVCO  ;Are we authorized to allocate devices?
005232  001004                         BNE     11$             ;Br if yes
005234  012700  000005                 MOV     #5,R0           ;Error code 5 if not authorized
005240  000137  000000G                JMP     SETERR
005244  004737  005636'         11$:    CALL    ALCCOM          ;Do common setup
005250  012702  000000G                MOV     #ALCTBL,R2      ;Point to start of allocation table
                                ;
                                ;  If the device name is null, deallocate all devices allocated by
                                ;  this user.
                                ;
005254  013700  000000G                MOV     FILSPC,R0       ;Is device name null?
005260  001015                         BNE     2$              ;Br if not
005262  120162  000000G         1$:     CMPB    R1,AD$JOB(R2)   ;Is this entry an allocation for our job?
005266  001004                         BNE     3$              ;Br if not
005270  105062  000000G                CLRB    AD$JOB(R2)      ;Say no job using this entry
005274  005062  000000G                CLR     AD$DVU(R2)      ;Say entry is free
005300  062702  000000G         3$:     ADD     #AD$$SZ,R2      ;Point to next allocation table entry
005304  020227  000000G                CMP     R2,#ALCEND      ;Checked all entries?
005310  103764                         BLO     1$              ;Loop if more to check
005312  000437                         BR      9$
                                ;
                                ;  The device name is not null.
                                ;  Deallocate the specified device.
                                ;
005314  105737  002014'         2$:     TSTB    FILDVU          ;Trying to deallocate TT?
005320  001434                         BEQ     9$              ;Ignore if yes
005322  013700  002014'                 MOV     FILDVU,R0       ;Get device/unit ID for device being dealloc
005326  020062  000000G         7$:     CMP     R0,AD$DVU(R2)   ;Search for specified device in alloc table
005332  001406                         BEQ     5$              ;Br if found it
005334  062702  000000G                ADD     #AD$$SZ,R2      ;Point to next entry in table
005340  020227  000000G                CMP     R2,#ALCEND      ;Checked all entries?
005344  103770                         BLO     7$              ;Keep looking if not
005346  000421                         BR      9$              ;Device was not allocated by anyone
005350  116200  000000G         5$:     MOVB    AD$JOB(R2),R0   ;Get # of job that owns this device
005354  126061  000000G 000000G        CMPB    LNPRIM(R0),LNPRIM(R1) ;Is device owned by us?
005362  001407                         BEQ     6$              ;Br if yes
005364  006200                         ASR     R0              ;Convert job index # to job #
005366  010037  000000G                MOV     R0,URO          ;Return owner job # in R0
005372  012700  000001                 MOV     #1,R0           ;Error 1 -- Device is allocated to someone
005376  000137  015370'                JMP     USRERR
005402  005062  000000G         6$:     CLR     AD$DVU(R2)      ;Say this table entry is free
005406  105062  000000G                CLRB    AD$JOB(R2)
                                ;
                                ;  Finished
                                ;
005412  000137  015404'         9$:     JMP     USRXIT
```

```
                                        .SBTTL  TLCEMT -- Check to see if a device is allocated
                                ;-----------------------------------------------------------------------
                                ;  Check to see if a device is allocated by another user.
                                ;
005416  005037  000000G  TLCEMT: CLR     URO             ;Clear user's RO in case ALCCOM aborts
005422  004737  005636'         CALL    ALCCOM          ;Do common setup
                                ;
                                ;  See if device is currently allocated
                                ;
005426  004737  005436'         CALL    ALCTST          ;See if device is allocated to another user
                                ;
                                ;  Finished -- Device is not allocated to any other user
                                ;
005432  000137  015404'         JMP     USRXIT          ;Device is not allocated to anyone else
```

```
                                        .SBTTL  ALCTST -- See if device is allocated to another user
                                ;-------------------------------------------------------------------------
                                ;  ALCTST is a subroutine called to determine if a device is allocated to
                                ;  another user or in use by another user.
                                ;
                                ;  Inputs:
                                ;   FILSPC = File spec for device being allocated.
                                ;   FILDVU = Device index # and unit # for device being tested.
                                ;   R1 = Current job index number
                                ;
                                ;  Outputs:
                                ;   If the device is in use by another user, error returns are made
                                ;   by jumping to USRERR.
                                ;   If the device is not allocated by other users, this routine returns.
                                ;
005436  010246                  ALCTST: MOV     R2,-(SP)
005440  010346                          MOV     R3,-(SP)
                                ;
                                ;  Clear returned value for R0 in case device is not in use or allocated
                                ;
005442  005037  000000G                 CLR     UR0             ;Say no use of device found yet
                                ;
                                ;  See if the device is legal
                                ;
005446  005737  000000G                 TST     FILSPC          ;Don't allow null device
005452  001413                          BEQ     11$             ;Error if name null
005454  105737  002014'                 TSTB    FILDVU          ;Trying to allocate TT?
005460  001410                          BEQ     11$             ;Br if yes
005462  123737  002014'  000000G 6$:    CMPB    FILDVU,SYINDX   ;Is this the system device?
005470  001010                          BNE     12$             ;Br if not
005472  123737  002015'  000001G         CMPB    FILDVU+1,SYUNIT+1;Is this the system unit?
005500  001004                          BNE     12$             ;Br if not
005502  012700  000002          11$:    MOV     #2,R0           ;Invalid device if null
005506  000137  015370'                  JMP     USRERR
                                ;
                                ;  See if the device is currently allocated to any user
                                ;
005512  012702  000000G         12$:    MOV     #ALCTBL,R2      ;Point to start of allocation table
005516  023762  002014'  000000G 1$:     CMP     FILDVU,AD$DVU(R2);Search for entry for this device and unit
005524  001423                          BEQ     5$              ;Br if found it
005526  062702  000000G                 ADD     #AD$$SZ,R2      ;Point to next entry
005532  020227  000000G                 CMP     R2,#ALCEND      ;Searched all entries?
005536  103767                          BLO     1$              ;Loop if not
                                ;
                                ;  Device is not currently allocated.
                                ;  See if any other job has a channel open to this device.
                                ;
005540  004737  005756'                 CALL    CHKUSE          ;See if device is being used by another job
005544  103007                          BCC     13$             ;Br if no other jobs have device open
005546  006200                          ASR     R0              ;Convert job index # to job #
005550  010037  000000G                 MOV     R0,UR0          ;Return in R0
005554  012700  000004                  MOV     #4,R0           ;Error 4 -- Device in use by another job
005560  000137  015370'                  JMP     USRERR
005564  006200                  13$:    ASR     R0              ;Get # of job that is using the device
005566  010037  000000G                 MOV     R0,UR0          ;Return in R0
005572  000416                          BR      9$
                                ;
```

```
                                        ;  The device is allocated.
                                        ;  See if it is allocated to our job or to another job.
                                        ;
005574  116203  0000000G      5$:       MOVB    AD$JOB(R2),R3   ;Get # of job to which dev is allocated
005600  010337  0000000G                MOV     R3,URO          ;Return job # to user in R0
005604  006237  0000000G                ASR     URO
005610  126361  0000000G 0000000G       CMPB    LNPRIM(R3),LNPRIM(R1) ;Is dev allocated to our family?
005616  001404                          BEQ     9$              ;Br if yes
005620  012700  000001                  MOV     #1,R0           ;Error 1 -- Device is allocated to another job
005624  000137  015370'                 JMP     USRERR
                                        ;
                                        ;  Device is not allocated to another user
                                        ;
005630  012603                  9$:     MOV     (SP)+,R3
005632  012602                          MOV     (SP)+,R2
005634  000207                          RETURN
```

```
                                        .SBTTL  ALCCOM -- Common setup for Allocate/Deallocate
                                ;-------------------------------------------------------------------------------
                                ;  Do common setup for Allocate/Deallocate EMT's.
                                ;
                                ;  Outputs:
                                ;   USR data base is locked for exclusive access.
                                ;   FILSPC = File spec passed with EMT.
                                ;   FILDVU = Device index # and unit #
                                ;   R4 = Device index number.
                                ;
005636  010146                  ALCCOM: MOV     R1,-(SP)
                                ;
                                ;  Gain exclusive access to USR data base
                                ;
005640  004737  010004'                 CALL    GETUSR          ;Get exclusive use of USR
                                ;
                                ;  Move file spec to FILSPC and apply any assigns
                                ;
005644  013701  000002G                 MOV     EMTBLK+2,R1     ;Get pointer to device spec argument
005650  004737  007546'                 CALL    GETSPC          ;Setup FILSPC
                                ;
                                ;  Check to see if the device is valid
                                ;
005654  005737  000000G                 TST     FILSPC          ;Is the device null?
005660  001430                        BEQ     2$              ;Treat this as a special valid device
005662  004737  010226'                 CALL    CHKDEV          ;See if device is valid
005666  103016                        BCC     1$              ;Br if CHKDEV says it is valid
005670  020437  000000G                 CMP     R4,CLDEVX       ;Is this a CL unit?
005674  001004                        BNE     3$              ;Br if not
005676  020027  000000G                 CMP     R0,#CLTOTL      ;Valid CL unit number?
005702  103417                        BLO     2$              ;Br if yes
005704  000412                        BR      8$              ;Invalid device
005706  020437  000000G         3$:     CMP     R4,C1DEVX       ;Is this a C1 unit?
005712  001007                        BNE     8$              ;Br if not
005714  020027  177770G                 CMP     R0,#CLTOTL-8.   ;Is this a valid C1 unit number?
005720  002410                        BLT     2$              ;Br if yes
005722  000403                        BR      8$              ;Br if not
005724  020437  000000G         1$:     CMP     R4,LDDEVX       ;Is this a logical disk?
005730  001004                        BNE     2$              ;Br if not -- Don't allow alloc of LD
                                ;
                                ;  This is an invalid device
                                ;
005732  012700  000002          8$:     MOV     #2,R0           ;Error 2 -- Invalid device
005736  000137  015370'                 JMP     USRERR
                                ;
                                ;  This is a valid device, Build the FILDVU word
                                ;
005742  110437  002014'         2$:     MOVB    R4,FILDVU       ;Set device index number
005746  110037  002015'                 MOVB    R0,FILDVU+1     ;Set unit number
                                ;
                                ;  Finished
                                ;
005752  012601                          MOV     (SP)+,R1
005754  000207                          RETURN
```

```
                                        .SBTTL  CHKUSE -- See if any channels open to a specified device
                                ;-------------------------------------------------------------------------
                                ;  CHKUSE is called to see if any jobs other than a specified job and its
                                ;  associated virtual jobs have any channels open to a specified device.
                                ;
                                ;  Inputs:
                                ;   R1 = Job whose access is to be allowed.
                                ;   FILDVU = Device index # and unit # of device to be checked.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> No job other than current job has
                                ;                      channels opened to the device.
                                ;   C-flag set     ==> Some other job has a channel open to the device.
                                ;   R0 = Job index number of job that is accessing the device.
                                ;        (Zero returned if no job is using the device).
                                ;
005756  010246                  CHKUSE: MOV     R2,-(SP)
005760  010346                          MOV     R3,-(SP)
005762  010446                          MOV     R4,-(SP)
005764  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get exclusive access to job context block buffer
                                ;
005766                                  QCALL   GETCXT          ;Get exclusive access to context block buffer
                                ;
                                ;  First check to see if the specified device is mounted by another user.
                                ;  We also check to see if any logical disk on the specified device is
                                ;  mounted.
                                ;
005774  005004                          CLR     R4              ;Have not found any job accessing device
005776  013705  000000G                 MOV     CSHDEV,R5       ;Point to start of mount table
006002  005765  000000G         6$:     TST     CD$DVU(R5)      ;Is this mount entry in use?
006006  001430                          BEQ     7$              ;Br if not
006010  026537  000000G 002014'         CMP     CD$DVU(R5),FILDVU ;Does this match device of interest?
006016  001024                          BNE     7$              ;Br if not
                                ;
                                ;  We found a mount entry for the device we are checking.
                                ;  Now we must determine which job(s) have mounted the device.
                                ;
006020  012702  000000G                 MOV     #LSTSL,R2       ;Get index # of last job
006024  020201                  8$:     CMP     R2,R1           ;Are we checking current job?
006026  001415                          BEQ     10$             ;Br if yes -- It is ok
006030  010203                          MOV     R2,R3           ;Get index number
006032  010246                          MOV     R2,-(SP)        ;Preserve R2
006034  004737  015324'                 CALL    CDJFLG          ;Get pointer to flag for this job
006040  010200                          MOV     R2,R0           ;Get pointer to byte with flag bit
006042  012602                          MOV     (SP)+,R2        ;Recover job number
006044  130310                          BITB    R3,(R0)         ;Does this job have device mounted?
006046  001405                          BEQ     10$             ;Br if not
                                ;
                                ;  We have found a job that has the device mounted.
                                ;  See if it is ok for that job to access the device.
                                ;
006050  126261  000000G 000000G         CMPB    LNPRIM(R2),LNPRIM(R1) ;Is this job allowed to access the dev?
006056  001073                          BNE     5$              ;Br if not
006060  010204                          MOV     R2,R4           ;Remember # of job accessing device
006062  162702  000002          10$:    SUB     #2,R2           ;More jobs to check?
```

```
     58 006066  003356                          BGT     8$              ;Br if yes
     59                                 ;
     60                                 ;  Check next mount table entry
     61                                 ;
     62 006070  062705  000000G         7$:     ADD     #CD$$SZ,R5      ;Point to next mount table entry
     63 006074  020537  000000G                 CMP     R5,CSHDVN       ;Have we checked all mount table entries?
     64 006100  103740                          BLO     6$              ;Br if not
     65                                 ;
     66                                 ;  Finished checking all mount table entries
     67                                 ;
     68 006102  010405                          MOV     R4,R5           ;Get # of any job accessing device
     69                                 ;
     70                                 ;  Begin loop to cycle through all jobs to see if any job has
     71                                 ;  a channel opened to this device.
     72                                 ;
     73 006104  012702  000002                  MOV     #2,R2           ;Get index # of 1st job
     74                                 ;
     75                                 ;  Ignore this job if it is not logged on.
     76                                 ;
     77 006110  032762  000000G 000000G 1$:     BIT     #$$KINIT,LSW(R2) ;Is this job logged on?
     78 006116  001440                          BEQ     2$              ;Br if not
     79                                 ;
     80                                 ;  Begin loop to cycle through all channels for this job
     81                                 ;
     82 006120  005003                          CLR     R3              ;Start with channel # 0
     83 006122                          3$:     OCALL   GETCHA          ;Get address of this channel
     84                                 ;
     85                                 ;  Ignore this channel if it is not open
     86                                 ;
     87 006130  032760  000000G 000000G        BIT     #CS$OPN,C.CSW(R0) ;Is this channel open?
     88 006136  001424                          BEQ     4$              ;Br if not
     89                                 ;
     90                                 ;  Get the device index number and the channel number out of the
     91                                 ;  channel block.
     92                                 ;
     93 006140  010004                          MOV     R0,R4           ;Get addr of channel block to R4
     94 006142  016400  000000G                 MOV     C.CSW(R4),R0    ;Get CSW
     95 006146  042700  177701                  BIC     #^C76,R0        ;Extract device index number
     96 006152  116404  000000G                 MOVB    C.DEVQ(R4),R4   ;Get unit number
     97 006156  042704  177770                  BIC     #^C7,R4         ;Clear all but unit number
     98 006162  120037  002014'                 CMPB    R0,FILDVU       ;Does it match the device of interest?
     99 006166  001010                          BNE     4$              ;Br if not
    100 006170  120437  002015'                 CMPB    R4,FILDVU+1     ;Does unit number match?
    101 006174  001005                          BNE     4$              ;Br if not
    102                                 ;
    103                                 ;  We found a job that has a channel open to the device
    104                                 ;
    105 006176  126261  000000G 000000G        CMPB    LNPRIM(R2),LNPRIM(R1) ;Is this job allowed to access the dev?
    106 006204  001020                          BNE     5$              ;Br if not
    107 006206  010205                          MOV     R2,R5           ;Remember # of friendly job
    108                                 ;
    109                                 ;  Check next channel (including Command file, log file, spool control
    110                                 ;  and SAV file load channels)
    111                                 ;
    112 006210  005203                 4$:     INC     R3              ;Advance the channel number
    113 006212  020327  000000G                 CMP     R3,#NLCHN       ;Checked all of user's channels?
    114 006216  103741                          BLO     3$              ;Loop if more to check
```

```
                                        ;
                                        ;  Check next job
                                        ;
006220  062702  000002          2$:     ADD     #2,R2           ;Get # of next job
006224  020227  000000G                 CMP     R2,#LSTSL       ;Checked all jobs?
006230  101727                          BLOS    1$              ;Loop if more to check
                                        ;
                                        ;  Finished checking all jobs
                                        ;  See if a friendly job is using the device
                                        ;
006232                                  OCALL   FRECXT          ;Free context block buffer
006240  010500                          MOV     R5,R0           ;Get # of any job accessing the device
006242  000241                          CLC                     ;Set flag saying device is free
006244  000405                          BR      9$
                                        ;
                                        ;  We found a job accessing the channel who is in conflict with testing job
                                        ;
006246                          5$:     OCALL   FRECXT          ;Free context block buffer
006254  010200                          MOV     R2,R0           ;Return job index number in R0
006256  000261                          SEC                     ;Set flag saying that device is in use
                                        ;
                                        ;  Finished
                                        ;
006260  012605                  9$:     MOV     (SP)+,R5
006262  012604                          MOV     (SP)+,R4
006264  012603                          MOV     (SP)+,R3
006266  012602                          MOV     (SP)+,R2
006270  000207                          RETURN
```

```
                                        .SBTTL  MOUNT  -- Mount a new file structure
                                ;-------------------------------------------------------------------------
                                ;  The MOUNT EMT is used to introduce a new file structure to the system.
                                ;
006272  004737  007134'         MOUNT:  CALL    MNTCOM          ;DO COMMON SETUP
                                ;
                                ;  First dismount the device to clear the directory file entries
                                ;
006276  004737  006746'                 CALL    DMTSUB          ;DISMOUNT THE DEVICE
                                ;
                                ;  See if this device is mounted by other users.
                                ;
006302  004737  015210'                 CALL    CSHTST          ;SEARCH FOR DEVICE IN MOUNT TABLE
006306  103105                         BCC     4$              ;BR IF DEVICE IS ALREADY MOUNTED
                                ;
                                ;  Device is not currently mounted by any job.
                                ;  Check to see if this device is eligible for directory caching.
                                ;
006310  113705  002014'                 MOVB    FILDVU,R5       ;GET DEVICE INDEX NUMBER
006314  016500  000000G                 MOV     DVSTAT(R5),R0   ;GET STATUS FLAGS FOR THIS DEVICE
006320  032700  000000G                 BIT     #DS$DIR,R0      ;IS THIS A DIRECTORY STRUCTURED DEVICE?
006324  001503                         BEQ     3$              ;BR IF NOT -- DON'T CACHE
006326  032700  000000G                 BIT     #DS$NRD,R0      ;NON RT-11 DIRECTORY STRUCTURE (MAG TAPE)?
006332  001100                         BNE     3$              ;BR IF YES -- DON'T CACHE
006334  032765  000000G 000000G         BIT     #DX$NRD,DVFLAG(R5);Internal non-RT-dir flag set?
006342  001074                         BNE     3$              ;Br if yes
006344  032765  000000G 000000G         BIT     #DX$NMT,DVFLAG(R5);Is it flagged for no mount?
006352  001070                         BNE     3$              ;Br if yes
                                ;
                                ;  Look for a free entry in the mount table.
                                ;
006354  013705  000000G                 MOV     CSHDEV,R5       ;POINT TO TABLE OF STRUCTURES
006360  005765  000000G         1$:     TST     CD$DVU(R5)      ;SEARCH FOR A FREE SLOT IN THE TABLE
006364  001411                         BEQ     2$              ;BR IF FOUND ONE
006366  062705  000000G                 ADD     #CD$$SZ,R5      ;POINT TO NEXT TABLE ENTRY
006372  020537  000000G                 CMP     R5,CSHDVN       ;HIT END OF TABLE?
006376  103770                         BLO     1$              ;BR IF NOT
006400  012700  000001                 MOV     #1,R0           ;NO FREE SLOTS IN TABLE
006404  000137  015370'                 JMP     USRERR
                                ;
                                ;  We found a free entry in the mount table.
                                ;  Set up the entry for this device.
                                ;
006410  004737  015266'         2$:     CALL    GETDVU          ;GET PHYSICAL DEVICE # AND UNIT #
006414  005065  000000G                 CLR     CD$NAM(R5)      ;CLEAR FILE NAME (ASSUME NOT LOGICAL DISK)
006420  005065  000002G                 CLR     CD$NAM+2(R5)
006424  012765  177777  000000G         MOV     #177777,CD$TOP(R5);ASSUME NOT LOGICAL DISK FOR TOP BLOCK
006432  010365  000000G                 MOV     R3,CD$DVU(R5)   ;STORE PHYSICAL DEVICE # AND UNIT #
006436  010465  000000G                 MOV     R4,CD$BAS(R5)   ;STORE BASE BLOCK # IF LOGICAL DISK
006442  001420                         BEQ     6$              ;BR IF NOT A LOGICAL DISK
006444  113703  002015'                 MOVB    FILDVU+1,R3     ;THIS IS A LOGICAL DISK, GET LD UNIT #
006450  006303                         ASL     R3              ;CONVERT UNIT # TO WORD TABLE INDEX
006452  066304  000000G                 ADD     LDSIZE(R3),R4   ;GET # OF BLOCK ABOVE TOP OF LOGICAL DISK
006456  010465  000000G                 MOV     R4,CD$TOP(R5)   ;SET TOP BLOCK NUMBER OF LOGICAL DISK
006462  072327  000002                 ASH     #2,R3           ;TIMES 8 BYTES PER ENTRY
006466  062703  000000G                 ADD     #LDNAME,R3      ;POINT TO ENTRY FOR NAME OF THIS LD FILE
006472  005723                         TST     (R3)+           ;SKIP OVER DEVICE NAME
```

```
     58 006474  012365  000000G                MOV     (R3)+,CD$NAM(R5);STORE 1ST 3 CHARS OF FILE NAME
     59 006500  011365  000002G                MOV     (R3),CD$NAM+2(R5);STORE 2ND 3 CHARS OF NAME
     60                                 ;
     61                                 ;  Initially clear all mount flags saying device is not mounted by
     62                                 ;  any job
     63                                 ;
     64 006504  010503                  6$:     MOV     R5,R3           ;POINT TO SET OF MOUNT-FLAGS FOR
     65 006506  062703  000000G                 ADD     #CD$JOB,R3      ; ALL JOBS
     66 006512  012700  000000G                 MOV     #CD$$UB,R0      ;GET # BYTES USED FOR MOUNT FLAGS
     67 006516  105023                  5$:     CLRB    (R3)+           ;SAY DEVICE NOT MOUNTED BY ANY JOBS
     68 006520  077002                          SOB     R0,5$
     69                                 ;
     70                                 ;  Set mount flag for our job
     71                                 ;
     72 006522  113703  000000G         4$:     MOVB    CORUSR,R3       ;Get our job index number
     73 006526  004737  015324'                CALL    CDJFLG          ;GET MOUNT FLAG FOR OUR JOB
     74 006532  150312                          BISB    R3,(R2)         ;SET MOUNT FLAG FOR OUR JOB
     75 006534  000137  015404'         3$:     JMP     USRXIT
```

```
                                        .SBTTL  DISMNT -- Dismount a file structure
                                ;-------------------------------------------------------------------
                                ;  The DISMNT EMT is used to remove a file structure from the system tables.
                                ;
006540                          DISMNT:
                                ;
                                ;  Determine if this is a request to dismount a specific file structure
                                ;  or a request to clean out the directory cache.
                                ;
006540  113700  000000G                MOVB    EMTBLK,R0       ;Get sub-function code
006544  120027  000001                 CMPB    R0,#1           ;Clean out directory cache?
006550  001430                         BEQ     CSHDMT          ;Br if yes
006552  120027  000002                 CMPB    R0,#2           ;Dismount all devices for job?
006556  001442                         BEQ     DMTALL          ;Br if yes
006560  120027  000003                 CMPB    R0,#3           ;Dismount a LD structure?
006564  001414                         BEQ     1$              ;Br if yes
006566  120027  000004                 CMPB    R0,#4           ;Get information about a LD structure?
006572  001413                         BEQ     2$              ;Br if yes
006574  120027  000005                 CMPB    R0,#5           ;Dismount all LD structures?
006600  001412                         BEQ     3$              ;Br if yes
                                ;
                                ;  This is a request to dismount a file structure
                                ;
006602  004737  007134'                CALL    MNTCOM          ;Do common setup
                                ;
                                ;  Call DMTSUB subroutine to do real dismount work.
                                ;
006606  004737  006746'                CALL    DMTSUB          ;Do the dismount
                                ;
                                ;  Finished
                                ;
006612  000137  015404'                JMP     USRXIT
                                ;
                                ;  Dismount a LD structure
                                ;
006616  000137  000000G         1$:    JMP     LDDEMT          ;Dismount a LD
                                ;
                                ;  Get information about a LD structure
                                ;
006622  000137  000000G         2$:    JMP     LDIEMT          ;Get info about a LD
                                ;
                                ;  Dismount all LD structures
                                ;
006626  000137  000000G         3$:    JMP     LDAEMT          ;Dismount all LD's

                                ;-------------------------------------------------------------------
                                ;  CSHDMT removes all entries from the directory cache.
                                ;
006632  004737  010004'         CSHDMT: CALL    GETUSR          ;GET EXCLUSIVE USE OF USR
006636  013700  000000G                 MOV     CSHHD,R0        ;POINT TO 1ST CACHED DIRECTORY ENTRY
006642  005060  000000G         1$:     CLR     FC$CDX(R0)      ;SAY ENTRY IS EMPTY
006646  005060  000000G                 CLR     FD$NAM(R0)
006652  016000  000000G                 MOV     FC$LNK(R0),R0   ;LINK TO NEXT ENTRY
006656  001371                         BNE     1$              ;BRANCH BACK IF ANOTHER TO FREE
006660  000137  015404'                JMP     USRXIT          ;FINISHED
```

```
                                        .SBTTL  DMTALL -- Dismount all mounted devices for job
                                ;-------------------------------------------------------------------------
                                ;  Dismount all devices for this job.
                                ;
006664                          DMTALL:
                                ;
                                ;  Gain exclusive access to USR data base
                                ;
006664  004737  010004'                 CALL    GETUSR          ;Get exclusive access to USR
                                ;
                                ;  Get flag bit that identifies our job in mount table entries
                                ;
006670  113703  000000G                 MOVB    CORUSR,R3       ;Get our job index number
006674  005005                          CLR     R5              ;Say no entry to point to yet
006676  004737  015324'                 CALL    CDJFLG          ;Get flag bit for our job
                                ;
                                ;  Locate each device this is mounted by our job
                                ;
006702  013705  000000G                 MOV     CSHDEV,R5       ;Point to 1st cache device entry
006706  005765  000000G         1$:     TST     CD$DVU(R5)      ;Is this entry in use?
006712  001406                          BEQ     2$              ;Br if not
006714  010500                          MOV     R5,R0           ;Get address of entry
006716  060200                          ADD     R2,R0           ;Point to byte with our job flag bit
006720  130310                          BITB    R3,(R0)         ;Is this device mounted by our job?
006722  001402                          BEQ     2$              ;Br if not
006724  004737  006766'                 CALL    DMTDEV          ;Dismount this entry
006730  062705  000000G         2$:     ADD     #CD$$SZ,R5      ;Point to next mount table entry
006734  020537  000000G                 CMP     R5,CSHDVN       ;Checked all entries?
006740  103762                          BLO     1$              ;Loop if not
                                ;
                                ;  Say job has no logical disks mounted
                                ;
                                ;       MOV     #MAXLD,R0       ;Get # of LD units
                                ;       MOV     #LDNAME,R5      ;Point to LD name table
                                ;3$:    CLR     (R5)            ;Say this LD not in use
                                ;       ADD     #8.,R5          ;Point to next LD table entry
                                ;       SOB     R0,3$           ;Br if more to do
                                ;
                                ;  Finished
                                ;
006742  000137  015404'                 JMP     USRXIT          ;Finished
```

```
                                ;-------------------------------------------------------------------------------
                                ;  The DMTSUB subroutine is called to do the actual work of dismounting
                                ;  a file structure.
                                ;
                                ;  Inputs:
                                ;   FILDVU = Device/Unit number info for device being dismounted.
                                ;
006746  010546                  DMTSUB: MOV     R5,-(SP)
                                ;
                                ;  Locate entry for device in cached-device table.
                                ;
006750  004737  015210'                 CALL    CSHTST          ;Locate entry for device in cached dev table
006754  103402                          BCS     9$              ;Br if device is not mounted
                                ;
                                ;  Device is mounted, dismount it.
                                ;
006756  004737  006766'                 CALL    DMTDEV          ;Dismount the device
                                ;
                                ;  Finished
                                ;
006762  012605                  9$:     MOV     (SP)+,R5
006764  000207                          RETURN
```

```
                                        .SBTTL  DMTDEV -- Remove entry from cache table
                                ;-------------------------------------------------------------------------
                                ;  DMTDEV is called to remove a particular entry from the mount table.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to cached device entry to be removed.
                                ;
006766  010246                  DMTDEV: MOV     R2,-(SP)
006770  010346                          MOV     R3,-(SP)
                                ;
                                ;  Reset mount flag for our job.
                                ;
006772  113703  000000G                 MOVB    CORUSR,R3       ;Get our job index number
006776  004737  015324'                 CALL    CDJFLG          ;GET MOUNT FLAG FOR OUR JOB
007002  140312                          BICB    R3,(R2)         ;RESET MOUNT FLAG FOR OUR JOB
                                ;
                                ;  See if any other users still have device mounted.
                                ;
007004  010503                          MOV     R5,R3           ;Point to bytes with mount flags
007006  062703  000000G                 ADD     #CD$JOB,R3
007012  012700  000000G                 MOV     #CD$$UB,R0      ;GET # BYTES WITH MOUNT FLAGS
007016  105723                  1$:     TSTB    (R3)+           ;ANY OTHER JOBS HAVE THIS DEVICE MOUNTED?
007020  001042                          BNE     9$              ;BR IF YES
007022  077003                          SOB     R0,1$
                                ;
                                ;  No other jobs have this device mounted.
                                ;  Check if this is the SY disk.  If so, don't dismount it.
                                ;
007024  005765  000000G                 TST     CD$BAS(R5)      ;IS THIS A LOGICAL DISK?
007030  001011                          BNE     2$              ;BR IF YES
007032  016500  000000G                 MOV     CD$DVU(R5),R0   ;GET DEVICE # AND UNIT #
007036  120037  000000G                 CMPB    R0,SYINDX       ;IS THIS SY DEVICE?
007042  001004                          BNE     2$              ;BR IF NOT
007044  000300                          SWAB    R0              ;GET UNIT # TO LOW-ORDER BYTE
007046  120037  000001G                 CMPB    R0,SYUNIT+1     ;IS THIS SY UNIT?
007052  001425                          BEQ     9$              ;BR IF YES -- DON'T DISMOUNT SY
                                ;
                                ;  This is not SY device, do the actual dismount.
                                ;  Tell data caching facility to clean out the data cache for this device.
                                ;
007054  005737  000000G         2$:     TST     CSHALC          ;Is data caching genned into system?
007060  001404                          BEQ     5$              ;Br if not
007062  016503  000000G                 MOV     CD$DVU(R5),R3   ;Get device and unit number for CSHCLN
007066  004777  000000G                 CALL    @CSHCLN         ;Clean out the data cache
                                ;
                                ;  Say device is no longer mounted
                                ;
007072  005065  000000G         5$:     CLR     CD$DVU(R5)      ;SAY DEVICE IS NOT MOUNTED
                                ;
                                ;  Remove file entries for this device from directory cache.
                                ;
007076  013700  000000G                 MOV     CSHHD,R0        ;POINT TO FIRST CACHED DIRECTORY ENTRY
007102  020560  000000G         3$:     CMP     R5,FC$CDX(R0)   ;IS FILE ON THIS DEVICE?
007106  001004                          BNE     4$              ;BR IF NOT
007110  005060  000000G                 CLR     FC$CDX(R0)      ;REMOVE ENTRY FROM CACHED DIRECTORY
007114  005060  000000G                 CLR     FD$NAM(R0)
007120  016000  000000G         4$:     MOV     FC$LNK(R0),R0   ;CHAIN TO NEXT ENTRY
```

```
     58 007124  001366                          BNE     3$              ;BR IF THERE IS ANOTHER
     59                                 ;
     60                                 ;  Finished
     61                                 ;
     62 007126  012603                  9$:     MOV     (SP)+,R3
     63 007130  012602                          MOV     (SP)+,R2
     64 007132  000207                          RETURN
```

```
                                        .SBTTL  MNTCOM -- MOUNT/DISMOUNT common setup
                                ;-------------------------------------------------------------------
                                ;  Do common setup for Mount/Dismount EMT's.
                                ;
                                ;  Outputs:
                                ;   USR data base locked for exclusive access.
                                ;   FILSPC = File spec passed with EMT.
                                ;   FILDVU = Device index # & Unit #.
                                ;
007134  010146                 MNTCOM: MOV     R1,-(SP)
007136  010446                         MOV     R4,-(SP)
                                ;
                                ;  Gain exclusive access to USR data base.
                                ;
007140  004737  010004'                 CALL    GETUSR          ;GET EXCLUSIVE USE OF USR
                                ;
                                ;  Move file spec to FILSPC and perform any assigns.
                                ;
007144  013701  000002G                 MOV     EMTBLK+2,R1     ;POINT TO DEVICE SPEC ARG
007150  004737  007546'                 CALL    GETSPC          ;SET UP FILSPC
                                ;
                                ;  Check to see if device is legal.
                                ;
007154  004737  010226'                 CALL    CHKDEV          ;LOOK UP THE DEVICE NAME
007160  103006                         BCC     1$              ;BR IF DEVICE IS OK
007162  004737  015426'                 CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
007166  012700  177776                 MOV     #-2,R0          ;INVALID DEVICE NAME
007172  000137  015370'                 JMP     USRERR
                                ;
                                ;  Build FILDVU word.
                                ;
007176  110437  002014'         1$:     MOVB    R4,FILDVU       ;SET DEVICE INDEX #
007202  110037  002015'                 MOVB    R0,FILDVU+1     ;SET UNIT #
                                ;
                                ;  Finished
                                ;
007206  012604                         MOV     (SP)+,R4
007210  012601                         MOV     (SP)+,R1
007212  000207                         RETURN
```

```
                                        .SBTTL  CPYMNT -- Copy mount entries from another job
                                ;-------------------------------------------------------------------------
                                ;  CPYMNT is called to mount for the current job all of the devices
                                ;  mounted by another job.
                                ;
                                ;  Inputs:
                                ;   R2 = Job index number of job whose mounts are to be copied.
                                ;
007214  010246                  CPYMNT: MOV     R2,-(SP)
007216  010346                          MOV     R3,-(SP)
007220  010446                          MOV     R4,-(SP)
007222  010546                          MOV     R5,-(SP)
007224  010204                          MOV     R2,R4           ;Save job index in R4
                                ;
                                ;  Begin to search through mount table for mounts for other job
                                ;
007226  013705  000000G                 MOV     CSHDEV,R5       ;Point to start of mount table
007232  010403                  1$:     MOV     R4,R3           ;Get # of job we are copying from
007234  004737  015324'                 CALL    CDJFLG          ;Get job flag for other job
007240  130312                          BITB    R3,(R2)         ;Is this entry mounted by other job?
007242  001405                          BEQ     2$              ;Br if not
007244  113703  000000G                 MOVB    CORUSR,R3       ;Get our job index #
007250  004737  015324'                 CALL    CDJFLG          ;Get mount bit for our job
007254  150312                          BISB    R3,(R2)         ;Say device is mounted by our job
007256  062705  000000G         2$:     ADD     #CD$$SZ,R5      ;Point to next mount table entry
007262  020537  000000G                 CMP     R5,CSHDVN       ;Checked all entries?
007266  103761                          BLO     1$              ;Br if not
                                ;
                                ;  Finished
                                ;
007270  012605                          MOV     (SP)+,R5
007272  012604                          MOV     (SP)+,R4
007274  012603                          MOV     (SP)+,R3
007276  012602                          MOV     (SP)+,R2
007300  000207                          RETURN
```

```
                        .SBTTL  CLRDIR -- Remove directory entries from dir cache
                ;-----------------------------------------------------------------------
                ;  This routine is called when a user-mode program writes to a directory
                ;  structured device that has been opened in non-file-structured mode.
                ;  It cleans out all directory cache entries for the device that is being
                ;  written to.
                ;
                ;  Inputs:
                ;   R2 = Device index number
                ;   R3 = Pointer to Channel Status Block being used by write
                ;
007302  010546          CLRDIR: MOV     R5,-(SP)
007304  013746  002014'          MOV     FILDVU,-(SP)    ;Save current device/file index info
                ;
                ;  Set up device and unit numbers in FILDVU
                ;
007310  110237  002014'          MOVB    R2,FILDVU       ;Set device index number
007314  116300  000000G          MOVB    C.DEVQ(R3),R0   ;Get device unit number
007320  042700  177770           BIC     #^C<7>,R0       ;Clear out all but unit number
007324  110037  002015'          MOVB    R0,FILDVU+1     ;Set unit number
                ;
                ;  See if this device is being cached
                ;
007330  004737  015210'          CALL    CSHTST          ;Is this device being cached?
007334  103414                   BCS     9$              ;Br if not
                ;
                ;  Clean out directory cache entries for this device
                ;
007336  013700  000000G          MOV     CSHHD,R0        ;Point to 1st cached dir entry
007342  020560  000000G  1$:     CMP     R5,FC$CDX(R0)   ;Is file on this device?
007346  001004                   BNE     2$              ;Br if not
007350  005060  000000G          CLR     FC$CDX(R0)      ;Remove entry from directory cache
007354  005060  000000G          CLR     FD$NAM(R0)
007360  016000  000000G  2$:     MOV     FC$LNK(R0),R0   ;Chain to next cache entry
007364  001366                   BNE     1$              ;Loop if more to check
                ;
                ;  Finished
                ;
007366  012637  002014'  9$:     MOV     (SP)+,FILDVU
007372  012605                   MOV     (SP)+,R5
007374  000207                   RETURN
```

```
                                .SBTTL  USRCOM -- Common setup
                        ;-------------------------------------------------------------------------
                        ;  USRCOM is called to perform the common setup operation for
                        ;  .lookup, .enter, .delete and .rename emts.
                        ;  It performs the following functions:
                        ;   1. Claim the USR data base for current user.
                        ;   2. Make sure channel is closed.
                        ;   3. Move file spec to FILSPC buffer and perform any assigns.
                        ;   4. Test that the specified device is legal.
                        ;   5. Mark channel open and set up device index and unit #.
                        ;
                        ;  Inputs:
                        ;   EMTBLK = Emt argument block.
                        ;   CHNADR = Address of CSW for current channel.
                        ;
                        ;  Outputs:
                        ;   Channel area = Marked as open, device index # and unit # set up.
                        ;   FILSPC = File spec after any assign performed.
                        ;   FILDVU = Device index # in low-byte, unit # in high byte.
                        ;   ASNSIZ = Size specified with ASSIGN command for logical device.
                        ;            0 if no size was specified with assign.
                        ;   R0 = Unit number of device being accessed.
                        ;   R3 = Address of CSW for channel.
                        ;   R4 = Index into device table for device being opened.
                        ;
007376  010146          USRCOM: MOV     R1,-(SP)
007400  013703  000000G         MOV     CHNADR,R3       ;GET ADDRESS OF CURRENT CHANNEL AREA
                        ;
                        ;  Claim USR data base for us
                        ;
007404  004737  010004'         CALL    GETUSR          ;GAIN EXCLUSIVE ACCESS TO USR DATA BASE
                        ;
                        ;  Make sure channel is closed now.
                        ;
007410  032763  000000G 000000G BIT     #CS$OPN,C.CSW(R3);IS CHANNEL OPEN NOW?
007416  001403                  BEQ     2$              ;BR IF NOT OPEN NOW
007420  005000                  CLR     R0              ;RETURN ERROR CODE OF 0
007422  000137  015370'         JMP     USRERR
                        ;
                        ;  Move file spec to FILSPC and apply any assigns.
                        ;
007426  013701  000002G 2$:     MOV     EMTBLK+2,R1     ;GET ADDRESS OF FILE SPEC IN USER'S AREA
007432  042701  000001          BIC     #1,R1           ;MAKE SURE ADDRESS IS EVEN
007436  004737  007546'         CALL    GETSPC          ;GET FILE SPEC TO FILSPC
                        ;
                        ;  Now check to see if the device is legal.
                        ;
007442  004737  010226'         CALL    CHKDEV          ;IS THE DEVICE LEGAL?
007446  103006                  BCC     1$              ;BR IF RECOGNIZED DEVICE
007450  004737  015426'         CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
007454  012700  177776          MOV     #-2,R0          ;INVALID DEVICE
007460  000137  015370'         JMP     USRERR
                        ;  (At this point R4 has device table index and R0 has device unit #)
                        ;  Set up FILDVU word.
007464  110437  002014' 1$:     MOVB    R4,FILDVU       ;SET DEVICE #
007470  110037  002015'         MOVB    R0,FILDVU+1     ;SET UNIT #
                        ;
```

```
                                        ;  Set up information in channel.
                                        ;
007474  110063  000000G                       MOVB    R0,C.DEVQ(R3)   ;SET DEVICE UNIT #
007500  010463  000000G                       MOV     R4,C.CSW(R3)    ;SET DEVICE TABLE INDEX # IN CSW
007504  052763  000000G 000000G               BIS     #CS$OPN,C.CSW(R3);MARK CHANNEL AS OPEN
007512  005063  000000G                       CLR     C.SBLK(R3)      ;STARTING BLOCK # OF FILE
007516  005063  000000G                       CLR     C.LENG(R3)      ;ALLOCATED LENGTH OF FILE
007522  005063  000000G                       CLR     C.USED(R3)      ;NO BLOCKS WRITTEN TO FILE YET
007526  105063  000000G                       CLRB    C.NUMQ(R3)      ;NO I/O OPERATIONS QUEUED ON CHANNEL YET
                                        ;
                                        ;  See if this device is allocated to some other user
                                        ;
007532  004737  010446'                       CALL    CHKALC          ;Check for allocation problems
                                        ;
                                        ;  See if we are authorized to access this device and file.
                                        ;
007536  004737  010602'                       CALL    CHKACC          ;CHECK FOR ACCESS AUTHORIZATION
                                        ;
                                        ;  Return successfully
                                        ;
007542  012601                              MOV     (SP)+,R1
007544  000207                              RETURN
```

```
                        .SBTTL  GETSPC -- Get file spec from user's area
                ;-----------------------------------------------------------------------
                ;  GETSPC is called to move a file specification from the user's address
                ;  space to the FILSPC buffer area.
                ;
                ;  Inputs:
                ;   R1 = Address of file spec in user's area.
                ;
                ;  Outputs:
                ;   FILSPC = File spec after processing.
                ;   ASNSIZ = Size specified with ASSIGN for logical device (0 if not spec)
                ;
007546  010146  GETSPC: MOV     R1,-(SP)
007550  010246          MOV     R2,-(SP)
007552  005037  002044'        CLR     ASNSIZ          ;NO ASSIGN SIZE YET
                ;
                ;  Copy the file specification from the user's area into FILSPC.
                ;
007556  012702  000000G         MOV     #FILSPC,R2      ;Copy spec into FILSPC
007562  004737  007720'         CALL    CPYSPC          ;Copy file spec from user's area
                ;
                ;  Apply any assignments to this file spec.
                ;
007566  013700  000000G         MOV     FILSPC,R0       ;GET DEVICE NAME
007572  001447                  BEQ     4$              ;BR IF NULL NAME
007574  032737  000000G 000000G BIT     #AF$BYA,RUNFLG  ;Should we bypass logical assignment?
007602  001043                  BNE     4$              ;Br if yes
007604  012702  000000G         MOV     #ASNTBL,R2      ;POINT TO START OF ASSIGN TABLE
007610  020062  000000G 2$:     CMP     R0,AT$LOG(R2)   ;SEARCH FOR DEVICE NAME IN ASSIGN TABLE
007614  001406                  BEQ     3$              ;BR IF FOUND
007616  062702  000000G         ADD     #AT$$SZ,R2      ;TRY NEXT ENTRY
007622  020227  000000G         CMP     R2,#ASNEND      ;DONE ALL?
007626  103770                  BLO     2$              ;BR IF NOT
007630  000430                  BR      4$              ;NAME WAS NOT ASSIGNED
                ;
                ;  Name was assigned.  Substitute assigned name.
                ;
007632  012701  000000G 3$:     MOV     #FILSPC,R1      ;POINT TO AREA WITH FILE SPEC
007636  016211  000000G         MOV     AT$DEV(R2),(R1) ;PUT IN REAL DEVICE NAME
007642  016200  000000G         MOV     AT$FIL(R2),R0   ;WAS FILE NAME ASSIGNED?
007646  001421                  BEQ     4$              ;BR IF NOT
007650  010061  000002          MOV     R0,2(R1)        ;PUT IN FILE NAME
007654  016261  000002G 000004  MOV     AT$FIL+2(R2),4(R1)
007662  016200  000000G         MOV     AT$EXT(R2),R0   ;WAS FILE EXTENSION ASSIGNED?
007666  001402                  BEQ     8$              ;BR IF NOT
007670  010061  000006          MOV     R0,6(R1)        ;PUT IN FILE EXT
007674  016200  000000G 8$:     MOV     AT$SIZ(R2),R0   ;WAS FILE SIZE ASSIGNED?
007700  001404                  BEQ     4$              ;BR IF NOT
007702  010061  000010          MOV     R0,8.(R1)       ;PUT IN FILE SIZE
007706  010037  002044'         MOV     R0,ASNSIZ       ;RETURN ASSIGN SIZE IN ASNSIZ
                ;
                ;  finished
                ;
007712  012602  4$:     MOV     (SP)+,R2
007714  012601          MOV     (SP)+,R1
007716  000207          RETURN
```

```
                                        .SBTTL  CPYSPC -- Copy file specification from user's area
                                ;-------------------------------------------------------------------
                                ;  Copy a 5-word file specification from the user's area into a
                                ;  system buffer.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of file specification in user's area.
                                ;   R2 = Address of 5-word buffer to receive specification.
                                ;
007720                          CPYSPC:
                                ;
                                ;  Make sure address of file spec is legal
                                ;
007720  010100                          MOV     R1,R0           ;GET ADDRESS OF FILE SPEC BUFFER
007722  004737  000000G                 CALL    UACHKW          ;MAKE SURE ADDRESS IS LEGAL
007726  103004                          BCC     5$              ;BR IF ADDRESS IS OK
007730  012700  177766                  MOV     #-12,R0         ;GET ERROR ABORT CODE
007734  000137  015356'                 JMP     USRCLS          ;ABORT THE JOB
                                ;
                                ;  Copy file specification from user's area to FILSPC.
                                ;
007740  032737  000000G 000000G 5$:     BIT     #UMODE,EMTPS    ;WAS EMT DONE IN USER OR KERNEL MODE?
007746  001410                          BEQ     6$              ;BR IF KERNEL MODE
007750  013700  000000G                 MOV     ODTBAS,R0       ;GET THE USER'S HIGH MEMORY LIMIT
007754  160100                          SUB     R1,R0           ;SUBTRACT ADDRESS OF FILE SPEC AREA
007756  006200                          ASR     R0              ;CONVERT TO NUMBER OF WORDS
007760  001410                          BEQ     9$              ;BR IF NOTHING TO MOVE
007762  020027  000004                  CMP     R0,#4.          ;COMPARE WITH SIZE OF FULL FILE SPEC
007766  101402                          BLOS    1$              ;BR IF WE MUST MOVE LESS BECAUSE OF MEMORY TOP
007770  012700  000004          6$:     MOV     #4.,R0          ;# WORDS TO MOVE
007774  106521                  1$:     MFPD    (R1)+           ;GET A WORD FROM USER'S AREA
007776  012622                          MOV     (SP)+,(R2)+     ;MOVE TO FILSPC
010000  077003                          SOB     R0,1$
                                ;
                                ;  Finished
                                ;
010002  000207                  9$:     RETURN
```

```
                                        .SBTTL  GETUSR -- Claim usr data base for our job
                                ;-------------------------------------------------------------------------
                                ;  GETUSR is called to gain exclusive access to the usr data base.
                                ;  If the data base is currently in use by some other job, our job
                                ;  is suspended until it is freed.
                                ;
010004  010546                  GETUSR: MOV     R5,-(SP)
                                ;
                                ;  See if we can access the USR
                                ;
010006                          4$:     DISABL                  ;;;DISABLE INTERRUPTS
010014  113700  000000G                MOVB    USRJOB,R0       ;;;IS USR AVAILABLE NOW?
010020  001426                  1$      BEQ     1$              ;;;BR IF YES
010022  120037  000000G                CMPB    R0,CORUSR       ;;;DO WE ALREADY OWN USR?
010026  001426                  2$      BEQ     2$              ;;;BR IF YES
                                ;
                                ;  Someone else owns the USR now.
                                ;  Suspend our job until we can get it.
                                ;
010030                                  ENABL                   ;ENABLE INTERRUPTS
010036  113705  000000G                MOVB    EXCJOB,R5       ;REMEMBER IF WE HAVE EXCLUSIVE SYS USE
010042  105037  000000G                CLRB    EXCJOB          ;ENABLE OTHER JOBS TO RUN
010046  012700  000000G                MOV     #S$QUSR,R0      ;PUT US IN QUEUE FOR USR
010052  004737  000000G                CALL    QNSPNX          ;SUSPEND JOB AND WAIT FOR USR
010056  004737  000000G                CALL    CHKABT          ;WERE WE ABORTED WHILE ASLEEP?
010062  020537  000000G                CMP     R5,CORUSR       ;DO WE WANT EXCLUSIVE ACCESS TO SYSTEM?
010066  001347                          BNE     4$              ;LOOP IF NOT
010070  110537  000000G                MOVB    R5,EXCJOB       ;GET EXCLUSIVE ACCESS TO THE SYSTEM
010074  000744                          BR      4$              ;GO TRY TO GET USR NOW
                                ;
                                ;  We can get USR now.
                                ;
010076  113737  000000G 000000G 1$:     MOVB    CORUSR,USRJOB   ;;;CLAIM USR FOR US
010104                          2$:     ENABL                   ;ENABLE INTERRUPTS
010112  005237  002026'                 INC     USRCNT          ;REMEMBER # TIMES WE CLAIMED USR
                                ;
                                ;  Make sure USR is not being called from a completion routine.
                                ;
010116  105737  000000G                TSTB    CURCP           ;ARE WE IN A COMPLETION ROUTINE NOW?
010122  001404                          BEQ     3$              ;BR IF NOT
010124  012700  177765                 MOV     #-13,R0         ;ABORT IF IN COMPLETION ROUTINE
010130  000137  015370'                 JMP     USRERR
                                ;
                                ;  Initialize USR data base.
                                ;
010134  012737  177777  002002' 3$:     MOV     #-1,CURSEG      ;NO DIR SEG IN BUFFER NOW
                                ;
                                ;  Finished
                                ;
010142  012605                          MOV     (SP)+,R5
010144  000207                          RETURN

                                        .SBTTL  FREUSR -- Free the USR
                                ;-------------------------------------------------------------------------
                                ;  FREUSR is called to release our ownership of the USR data base.
                                ;
010146                          FREUSR: DISABL                  ;;;DISABLE INTERRUPTS
```

```
     58 010154  123737  000000G 000000G          CMPB    CORUSR,USRJOB   ;;;DO WE OWN USR?
     59 010162  001015                          BNE     1$              ;;;BR IF NOT
     60 010164  005337  002026'                 DEC     USRCNT          ;;;IS THIS LAST UNLOCK OF USR?
     61 010170  002012                          BGE     1$              ;;;BR IF NOT
     62 010172  105037  000000G                 CLRB    USRJOB          ;;;SAY USR IS FREE
     63                                 ;
     64                                 ;  Restart any jobs waiting for USR
     65                                 ;
     66 010176                                  ENABL                   ;ENABLE INTERRUPTS
     67 010204  012700  000000G                 MOV     #S$QUSR,R0      ;QUEUE OF WAITING JOBS
     68 010210  004737  000000G                 CALL    UREGO           ;RESTART WAITING JOBS
     69 010214  000207                          RETURN
     70                                 ;
     71                                 ;  Finished
     72                                 ;
     73 010216                          1$:     ENABL                   ;ENABLE INTERRUPTS
     74 010224  000207                          RETURN
```

```
                                        .SBTTL  CHKDEV -- See if requested device is legal
                                ;-------------------------------------------------------------------
                                ;  CHKDEV is called to see if access to the device specified by the
                                ;  file spec in FILSPC is legal.
                                ;
                                ;  Inputs:
                                ;   FILSPC = Device file specification
                                ;
                                ;  Outputs:
                                ;   R0 = Unit number of device
                                ;   R4 = Index into device tables
                                ;   C-flag set on return if the device is not recognized.
                                ;
010226  010146          CHKDEV: MOV     R1,-(SP)
010230  010246                  MOV     R2,-(SP)
                                ;
                                ;  Get device name and split off unit number.
                                ;
010232  013701  000000G         MOV     FILSPC,R1       ;GET DEVICE NAME (RAD50)
010236  005000                  CLR     R0              ;SET FOR DIVIDE
010240  071027  000050          DIV     #50,R0          ;SPLIT OFF LOW-ORDER RAD50 CHARACTER
010244  012702  177777          MOV     #-1,R2          ;ASSUME NO UNIT NUMBER SPECIFIED
010250  005701                  TST     R1              ;WAS A UNIT NUMBER SPECIFIED?
010252  001406                  BEQ     6$              ;BR IF NOT
010254  162701  000036          SUB     #36,R1          ;CONVERT RAD50 DIGIT TO BINARY VALUE
010260  010102                  MOV     R1,R2           ;GET BINARY VALUE OF UNIT NUMBER
010262  020227  000007          CMP     R2,#7           ;RESTRICT UNIT NUMBER TO RANGE 0-7
010266  101063                  BHI     8$              ;BR IF INVALID UNIT NUMBER
010270  070027  000050  6$:     MUL     #50,R0          ;GET DEVICE NAME WITHOUT UNIT NUMBER
                                ;
                                ;  The rad50 device name less unit number is now in R1.
                                ;  R2 has the binary value of the unit number or -1 if no unit number
                                ;  was specified.
                                ;
                                ;  Translate "SY:" and "DK:" to physical device.
                                ;
010274  020137  002020'          CMP     R1,R50DK        ;IS DEVICE NAME "DK"?
010300  001403                  BEQ     2$              ;BR IF YES
010302  020137  002016'          CMP     R1,R50SY        ;IS DEVICE NAME "SY"?
010306  001007                  BNE     3$              ;BR IF NOT
010310  013704  000000G 2$:     MOV     SYINDX,R4       ;GET SY DEVICE INDEX NUMBER
010314  005702                  TST     R2              ;DID USER SPECIFY A UNIT NUMBER?
010316  002014                  BGE     7$              ;BR IF YES
010320  113702  000001G         MOVB    SYUNIT+1,R2     ;GET SYSTEM DEVICE UNIT NUMBER
010324  000411                  BR      7$
                                ;
                                ;  Look up device name in permanent device name table.
                                ;
010326  013704  000000G 3$:     MOV     NUMDEV,R4       ;GET INDEX NUMBER OF LAST DEVICE
010332  020164  000000G 5$:     CMP     R1,PNAME(R4)    ;SEARCH FOR NAME IN TABLE
010336  001404                  BEQ     7$              ;BR IF FOUND
010340  162704  000002          SUB     #2,R4           ;TRY NEXT ENTRY
010344  002372                  BGE     5$              ;LOOP IF MORE TO CHECK
010346  000433                  BR      8$              ;Invalid device
                                ;
                                ;  Found device name.  Translate no unit number into # 0.
                                ;
```

```
     58 010350  010200                7$:     MOV     R2,R0           ;GET UNIT NUMBER VALUE
     59 010352  002001                        BGE     1$              ;BR IF UNIT NUMBER WAS SPECIFIED
     60 010354  005000                        CLR     R0              ;SAY UNIT # = 0 IF NONE SPECIFIED
     61                               ;
     62                               ;  If the device is a logical disk (LDn), check to make sure the
     63                               ;  particular unit is mapped to a file
     64                               ;
     65 010356  020437  000000G       1$:     CMP     R4,LDDEVX       ;IS THIS A LOGICAL DISK?
     66 010362  001006                        BNE     11$             ;BR IF NOT
     67 010364  010002                        MOV     R0,R2           ;GET UNIT NUMBER
     68 010366  006302                        ASL     R2              ;CONVERT TO WORD TABLE INDEX
     69 010370  005762  000000G               TST     LDPDEV(R2)      ;IS UNIT MAPPED TO A FILE?
     70 010374  001420                        BEQ     8$              ;BR IF NOT
     71 010376  000415                        BR      9$
     72                               ;
     73                               ;  If the device is a communications line (CLn), check to make sure
     74                               ;  the specified CL unit is assigned to some line.
     75                               ;
     76 010400  010002                11$:    MOV     R0,R2           ;Get unit number
     77 010402  020437  000000G               CMP     R4,CLDEVX       ;Is this a CL unit?
     78 010406  001405                        BEQ     12$             ;Br if yes
     79 010410  020437  000000G               CMP     R4,C1DEVX       ;Is this a C1 unit?
     80 010414  001006                        BNE     9$              ;Br if not
     81 010416  062702  000010                ADD     #8.,R2          ;Bias unit number by 8 for C1
     82 010422  006302                12$:    ASL     R2              ;Convert to word table index
     83 010424  005762  000000G               TST     CL$LIX(R2)      ;Is this unit assigned to some line?
     84 010430  001402                        BEQ     8$              ;Br if not
     85                               ;
     86                               ;  This device access is ok
     87                               ;
     88 010432  000241                9$:     CLC                     ;SIGNAL GOOD RETURN
     89 010434  000401                        BR      10$
     90                               ;
     91                               ;  Invalid device
     92                               ;
     93 010436  000261                8$:     SEC                     ;SIGNAL INVALID DEVICE NAME
     94                               ;
     95                               ;  Finished -- Return
     96                               ;
     97 010440  012602                10$:    MOV     (SP)+,R2
     98 010442  012601                        MOV     (SP)+,R1
     99 010444  000207                        RETURN
```

```
                                        .SBTTL  CHKALC -- Check for device allocation
                                ;-------------------------------------------------------------------
                                ;  CHKALC is called to determine if a device being accessed is available
                                ;  to the current job due to allocation considerations.
                                ;  If an allocation failure is detected, a fatal error return is taken.
                                ;
                                ;  Inputs:
                                ;   FILDVU = Device index # and unit # of device to be checked.
                                ;
010446  010046                  CHKALC: MOV     R0,-(SP)
010450  010246                          MOV     R2,-(SP)
                                ;
                                ;  Don't do allocation test for TT
                                ;
010452  105737  002014'                 TSTB    FILDVU          ;Device # 0 ==> TT
010456  001446                          BEQ     9$              ;Br if this device is TT
                                ;
                                ;  See if this device is in the allocation table.
                                ;
010460  012702  000000G                 MOV     #ALCTBL,R2      ;Point to start of allocation table
010464  013700  002014'                 MOV     FILDVU,R0       ;Get device/unit id
010470  020062  000000G          1$:    CMP     R0,AD$DVU(R2)   ;Is this entry for this device?
010474  001421                          BEQ     2$              ;Br if yes
010476  062702  000000G                 ADD     #AD$$SZ,R2      ;Point to next table entry
010502  020227  000000G                 CMP     R2,#ALCEND      ;Have we checked all entries?
010506  103770                          BLO     1$              ;Br if not
                                ;
                                ;  This device is not in the allocation table.
                                ;  See if it must be allocated before it can be accessed.
                                ;
010510  113700  002014'                 MOVB    FILDVU,R0       ;Get the device index number
010514  032760  000000G 000000G         BIT     #DX$RAL,DVFLAG(R0);Is allocation required before access?
010522  001424                          BEQ     9$              ;Br if not
010524  004737  015426'                 CALL    ERRNAM          ;Set file spec for Kmon error message
010530  012700  177751                  MOV     #-27,R0         ;Set error code
010534  000137  015356'                 JMP     USRCLS          ;Error return from USR
                                ;
                                ;  This device is in the allocation table.
                                ;  See if it is allocated to our job or to another job.
                                ;
010540  116202  000000G          2$:    MOVB    AD$JOB(R2),R2   ;Get # of job that owns the device
010544  113700  000000G                 MOVB    CORUSR,R0       ;Get current job index number
010550  126260  000000G 000000G         CMPB    LNPRIM(R2),LNPRIM(R0) ;Do we own the device?
010556  001406                          BEQ     9$              ;Br if yes
010560  004737  015426'                 CALL    ERRNAM          ;Set file spec for Kmon error message
010564  012700  177752                  MOV     #-26,R0         ;Set error code
010570  000137  015356'                 JMP     USRCLS          ;Error return
                                ;
                                ;  Device access is ok
                                ;
010574  012602                  9$:    MOV     (SP)+,R2
010576  012600                          MOV     (SP)+,R0
010600  000207                          RETURN
```

```
                                        .SBTTL  CHKACC -- Check legality of file access
                                ;-------------------------------------------------------------------
                                ;  CHKACC is called to see if a user is authorized to access a
                                ;  certain device and file.
                                ;
                                ;  Inputs:
                                ;   R0 = Unit number of device being accessed.
                                ;   R4 = Device index number of device being accessed.
                                ;   FILSPC = Device-file specification of file being accessed.
                                ;   R3 = Address of channel block associated with operation.
                                ;
                                ;  Outputs:
                                ;   Abort if access denied.
                                ;   CS$RON set in channel CSW if read-only access authorized.
                                ;
010602  010046                 CHKACC: MOV     R0,-(SP)
010604  010146                         MOV     R1,-(SP)
010606  010246                         MOV     R2,-(SP)
010610  010346                         MOV     R3,-(SP)
010612  010446                         MOV     R4,-(SP)
010614  010546                         MOV     R5,-(SP)
010616  113701  000000G                MOVB    CORUSR,R1       ;GET JOB INDEX NUMBER
                                ;
                                ;  See if we are accessing a logical disk with NOWRITE attribute set.
                                ;
010622  020437  000000G                CMP     R4,LDDEVX       ;IS THIS A LOGICAL DISK?
010626  001011                         BNE     13$             ;BR IF NOT
010630  010002                         MOV     R0,R2           ;GET UNIT NUMBER
010632  006302                         ASL     R2              ;CONVERT TO WORD TABLE INDEX
010634  032762  000000G 000000G        BIT     #LD$RON,LDFLAG(R2) ;WAS /NOWRITE SPECIFIED FOR DISK?
010642  001403                         BEQ     13$             ;BR IF NOT
010644  052763  000000G 000000G        BIS     #CS$RON,C.CSW(R3);SAY READ-ONLY ACCESS ALLOWED
                                ;
                                ;  Allow access to all devices if user has BYPASS privilege
                                ;
010652  032737  000000G 000000G 13$:  BIT     #PO$BYP,PRIVCO  ;Does user have BYPASS privilege?
010660  001004                         BNE     19$             ;Br if yes
                                ;
                                ;  See if TSXUCL is trying to access its command file
                                ;
010662  032761  000000G 000000G        BIT     #$UCLRN,LSW7(R1);IS TSXUCL RUNNING?
010670  001402                         BEQ     16$             ;Br if not
010672  000137  011250'         19$:  JMP     9$              ;Allow full file access
                                ;
                                ;  See if a non-privileged user is trying to access a SYS or TSX file on SY:
                                ;
010676  032737  000000G 000000G 16$:  BIT     #PO$SYS,PRIVCO  ;IS THIS A PRIVILEGED USER?
010704  001026                         BNE     1$              ;BR IF PRIVILEGED USER
010706  032761  000000G 000000G        BIT     #$NOIN,LSW3(R1) ;ARE WE RUNNING IN START-UP COMMAND FILE?
010714  001022                         BNE     1$              ;BR IF YES -- GIVE FULL ACCESS DURING STARTUP
010716  120437  000000G                CMPB    R4,SYINDX       ;Is device = SY?
010722  001017                         BNE     1$              ;Br if not SY
010724  120037  000001G                CMPB    R0,SYUNIT+1     ;Is unit = SY?
010730  001014                         BNE     1$              ;Br if not
010732  013702  000006G                MOV     FILSPC+6,R2     ;GET FILE EXTENSION
010736  020237  002022'                CMP     R2,R50SYS       ;IS EXTENSION "SYS"?
010742  001531                         BEQ     8$              ;BR IF YES -- DISALLOW ACCESS
```

```
     58 010744  020237  002024'                    CMP     R2,R50TSX       ;IS EXTENSION "TSX"?
     59 010750  001004                             BNE     1$              ;BR IF NOT .TSX FILE
     60 010752  105737  000000G                    TSTB    KMONCE          ;IS ONCE-ONLY CODE LOOKING FOR HANDLERS?
     61 010756  001131                             BNE     7$              ;IF YES -- ALLOW READ-ONLY ACCESS
     62 010760  000522                             BR      8$              ;BR IF NOT -- DISALLOW ACCESS
     63                                 ;
     64                                 ;  See if this is a non-file-structured lookup to a file-structured device
     65                                 ;
     66 010762  005737  000002G         1$:     TST     FILSPC+2        ;Non-file-structured lookup?
     67 010766  001017                          BNE     15$             ;Br if not
     68 010770  032764  000000G 000000G        BIT     #DS$DIR,DVSTAT(R4) ;Is this a directory structured device?
     69 010776  001413                          BEQ     15$             ;Br if not
     70 011000  032737  000000G 000000G        BIT     #PO$NFW,PRIVCO  ;May user do NFS open and then write?
     71 011006  001007                          BNE     15$             ;Br if he has write access
     72 011010  032737  000000G 000000G        BIT     #PO$NFR,PRIVCO  ;May user do NFS open and read?
     73 011016  001503                          BEQ     8$              ;Br if not -- No access allowed
     74 011020  052763  000000G 000000G        BIS     #CS$RON,C.CSW(R3) ;Force read-only access
     75                                 ;
     76                                 ;  See if ACCESS command was used to restrict access to any devices or files.
     77                                 ;
     78 011026  005737  000000G         15$:    TST     RESDEV          ;ANY RESTRICTED DEVICES AND FILES?
     79 011032  001506                          BEQ     9$              ;BR IF NOT
     80                                 ;
     81                                 ;  There are restricted devices and files.
     82                                 ;  See if access is authorized.
     83                                 ;
     84 011034  110437  002046'                 MOVB    R4,CKADEV       ;SAVE DEVICE INDEX NUMBER
     85 011040  001503                          BEQ     9$              ;You always get full access to TT:
     86 011042  110037  002047'                 MOVB    R0,CKAUNT       ;SAVE DEVICE UNIT #
     87 011046  005002                          CLR     R2              ;Init read-only flag
     88                                 ;
     89                                 ;  Check entries on the ACCESS list
     90                                 ;
     91 011050  012705  000000G                 MOV     #OKFILE,R5      ;Point to start of ACCESS list
     92 011054  020537  000000G                 CMP     R5,OKFAND       ;Are there any ACCESS entries?
     93 011060  103021                          BHIS    2$              ;If not, access OK, go check NOACCESSes
     94 011062  004737  011266'         6$:     CALL    ACCMCH          ;See if entry is on ACCESS list
     95 011066  103407                          BCS     4$              ;If not, go to next entry
     96 011070  032765  000000G 000000G        BIT     #OT$RON,OF$FLG(R5);Match found, was it read-only?
     97 011076  001002                          BNE     41$             ;If read only, go set read-only flag
     98 011100  005002                          CLR     R2              ;Hey! this gets full access
     99 011102  000410                          BR      2$              ;Go check NOACCESSes
    100 011104  005202                 41$:     INC     R2              ;Remember we have limited access
    101 011106  062705  000000G         4$:     ADD     #OF$$SZ,R5      ;Point to next authorized file spec
    102 011112  020537  000000G                 CMP     R5,OKFAND       ;Hit end of ACCESS list?
    103 011116  103761                          BLO     6$              ;And keep checking to end
    104                                ;There are entries on the ACCESS list, but we never got full access
    105 011120  005702                          TST     R2              ;Did we get at least partial access?
    106 011122  001423                          BEQ     5$              ;If not, have to deny access
    107                                ;
    108                                ;  We have at least partial access to this file, check NOACCESS list
    109                                ;
    110 011124  013705  000000G         2$:     MOV     OKFNND,R5       ;Point to start of NOACCESS list
    111 011130  020527  000000G         3$:     CMP     R5,#OKFEND      ;Are there any more entries on the list?
    112 011134  103013                          BHIS    38$             ;Br if not
    113 011136  004737  011266'                 CALL    ACCMCH          ;See if this entry is forbidden
    114 011142  103405                          BCS     35$             ;If no match, step up to next
```

```
    115 011144  032765  000000G 000000G          BIT     #OT$RON,OF$FLG(R5) ;Match, is write access forbidden?
    116 011152  001407                          BEQ     5$                 ;No, all access is denied
    117 011154  005202                          INC     R2                 ;Just disallow write access for now
    118 011156  062705  000000G           35$:  ADD     #OF$$SZ,R5         ;Point to next entry
    119 011162  000762                          BR      3$                 ;And keep checking
    120                                 ;  We got here without being totally denied access
    121 011164  005702                  38$:    TST     R2                 ;Were we denied write access?
    122 011166  001430                          BEQ     9$                 ;No access restrictions
    123 011170  000424                          BR      7$                 ;Go flag as read-only
    124                                 ;
    125                                 ;  Access restrictions forbid any access to this file.
    126                                 ;  However, allow implicit access to some devices and files.
    127                                 ;
    128                                 ;  Allow DUP to access SY: on a read-only basis
    129                                 ;
    130 011172  113701  000000G         5$:     MOVB    CORUSR,R1          ;GET JOB INDEX NUMBER
    131 011176  032761  000000G 000000G        BIT     #$DUPRN,LSW6(R1);IS DUP RUNNING?
    132 011204  001410                          BEQ     8$                 ;BR IF NOT
    133 011206  123737  002046' 000000G        CMPB    CKADEV,SYINDX      ;IS DEVICE = SY?
    134 011214  001004                          BNE     8$                 ;BR IF NOT
    135 011216  123737  002047' 000001G        CMPB    CKAUNT,SYUNIT+1 ;IS UNIT = SY?
    136 011224  001406                          BEQ     7$                 ;BR IF SY -- ALLOW READ ONLY ACCESS
    137                                 ;
    138                                 ;  User is not allowed to access this device or file.
    139                                 ;
    140 011226  004737  015426'         8$:     CALL    ERRNAM             ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
    141 011232  012700  177764                  MOV     #-14,R0            ;SET ERROR CODE
    142 011236  000137  015356'                 JMP     USRCLS             ;ABORT
    143                                 ;
    144                                 ;  User is allowed read-only access to this device or file.
    145                                 ;  Set CS$RON flag in CSW.
    146                                 ;
    147 011242  052763  0000000 0000000 7$:     BIS     #CS$RON,C.CSW(R3);SET READ-ONLY ACCESS FLAG IN CSW
    148                                 ;
    149                                 ;  User is allowed full access to file.
    150                                 ;
    151 011250  012605                  9$:     MOV     (SP)+,R5
    152 011252  012604                          MOV     (SP)+,R4
    153 011254  012603                          MOV     (SP)+,R3
    154 011256  012602                          MOV     (SP)+,R2
    155 011260  012601                          MOV     (SP)+,R1
    156 011262  012600                          MOV     (SP)+,R0
    157 011264  000207                          RETURN
```

```
                                        .SBTTL  ACCMCH -- Compare file spec with [NO]ACCESS entry
                                ;-----------------------------------------------------------------
                                ;  Check to see if dev/file specification matches a [NO]ACCESS table entry
                                ;
                                ;  Inputs:
                                ;       R5      points to [NO]ACCESS entry to be tested
                                ;       CKADEV  contains device index in question
                                ;       CKAUNT  contains device unit # in question
                                ;       FILSPC  contains ^R/FILNAMEXT/ in question
                                ;
                                ;  Outputs:
                                ;       C-clear entry matches
                                ;       C-set   entry does not match
                                ;
011266                          ACCMCH:
011266  010146                          MOV     R1,-(SP)
011270  010546                          MOV     R5,-(SP)
011272  005765  000000G                 TST     OF$FIL(R5)      ;Is this entry in use?
011276  001433                          BEQ     7$              ;If not, certainly cannot match
                                ;
                                ;  Check for match on device index and unit number
                                ;
011300  116500  000000G                 MOVB    OF$DEV(R5),R0   ;Get [NO]ACCESS device index
011304  100411                          BMI     2$              ;Wildcard always matches
011306  123700  002046'                 CMPB    CKADEV,R0       ;Does the device index match?
011312  001025                          BNE     7$              ;Br to no match if not
011314  116500  000000G                 MOVB    OF$UNT(R5),R0   ;Get [NO]ACCESS unit number
011320  100403                          BMI     2$              ;Wildcard always matches
011322  123700  002047'                 CMPB    CKAUNT,R0       ;Does the unit # match?
011326  001017                          BNE     7$              ;Br to no match if not
                                ;
                                ;  Device and unit are an acceptable match
                                ;  Check file specification
                                ;
011330  062705  000000G         2$:     ADD     #OF$FIL,R5      ;Point to [NO]ACCESS file spec
011334  012701  000002G                 MOV     #FILSPC+2,R1    ;Point to file spec in question
011340  012700  000003                  MOV     #3,R0           ;Get # words to compare (FIL NAM EXT)
011344  022125                  5$:     CMP     (R1)+,(R5)+     ;Compare FIL NAM EXT
011346  001404                          BEQ     3$              ;Br if this one matches
011350  026527  177776  000000G         CMP     -2(R5),#WLDNAM  ;Doesn't match, was [NO]ACCESS wild?
011356  001003                          BNE     7$              ;If not, no match
011360  077007                  3$:     SOB     R0,5$           ;This part matches, check rest?
                                ;
011362  000241                          CLC                     ;Entry matches, signal success
011364  000401                          BR      9$
                                ;
011366  000261                  7$:     SEC                     ;Entry does not match, signal no match
                                ;
011370  012605                  9$:     MOV     (SP)+,R5
011372  012601                          MOV     (SP)+,R1
011374  000207                          RETURN
```

```
                                        .SBTTL  FNDFIL -- Find file in directory
                                ;-------------------------------------------------------------------
                                ;  FNDFIL is called to locate the directory entry for a permanent file.
                                ;
                                ;  Inputs:
                                ;   FILSPC = File spec to be searched for.
                                ;
                                ;  Outputs:
                                ;   R0 = Starting block number of file.
                                ;   R1 = Address of directory entry (located in USRBUF).
                                ;   USRBUF = Directory segment containing entry.
                                ;   CURSEG = Number of directory segment.
                                ;  C-flag set if file not found.
                                ;
011376  010246                  FNDFIL: MOV     R2,-(SP)
011400  010346                          MOV     R3,-(SP)
                                ;
                                ;  Read in directory segment number 1 and set up info about directory.
                                ;
011402  004737  013656'                 CALL    RDSEG1          ;READ IN FIRST DIRECTORY SEGMENT
                                ;
                                ;  Search for permanent file entry that matches our name.
                                ;
011406  012700  000000G         2$:     MOV     #FS$PRM,R0      ;LOCATE NEXT PERM FILE ENTRY
011412  004737  013370'                 CALL    GETDIR
011416  103415                          BCS     3$              ;BR IF HIT END OF DIRECTORY
                                ;  Compare file names.
011420  010103                          MOV     R1,R3           ;GET ADDRESS OF DIRECTORY ENTRY
011422  062703  000000G               ADD     #FD$NAM,R3      ;POINT TO FILE NAME IN DIR ENTRY
011426  012702  000002G               MOV     #FILSPC+2,R2    ;POINT TO NAME WE WANT TO FIND
011432  012700  000003                MOV     #3,R0           ;COMPARE 3 WORDS
011436  022223                  1$:     CMP     (R2)+,(R3)+     ;DO NAMES MATCH?
011440  001362                          BNE     2$              ;BR IF NOT
011442  077003                          SOB     R0,1$           ;CHECK ALL OF NAME
                                ;
                                ;  Found the file entry -- Calculate starting block number.
                                ;
011444  004737  013622'                 CALL    SBCALC          ;CALCULATE STARTING BLOCK # FOR ENTRY
                                ;
                                ;  Finished
                                ;
011450  000241                  4$:     CLC                     ;SIGNAL GOOD RETURN
011452  012603                  3$:     MOV     (SP)+,R3
011454  012602                          MOV     (SP)+,R2
011456  000207                          RETURN
```

```
                                        .SBTTL  FNDFRE -- Find a free slot for a new file
                                ;-------------------------------------------------------------------------
                                ;  FNDFRE is called to locate a free file entry for a new file.
                                ;
                                ;  Inputs:
                                ;   R2 = Desired size (-1==>largest; 0==>Default algorithm)
                                ;   FILSPC = Device-file spec for file being entered.
                                ;
                                ;  Outputs:
                                ;   R0 = Error code if C-flag set on return.
                                ;        (1==>No free space of adequate size; 3==>Protected file conflict).
                                ;   R1 = Address of free file directory entry.
                                ;   R2 = Actual # of blocks to be used for file.
                                ;   USRBUF = Directory segment that has free slot entry.
                                ;   C-flag set if insufficient disk space or protected file conflict.
                                ;
011460  010346                 FNDFRE: MOV     R3,-(SP)
011462  010446                         MOV     R4,-(SP)
                                ;
                                ;  Initialize tables that hold info about 2 largest free slots.
                                ;
011464  005037  002030'                 CLR     L1SIZ           ;SIZE OF LARGEST FREE SLOT
011470  005037  002036'                 CLR     L2SIZ           ;SIZE OF 2ND LARGEST FREE SLOT
                                ;
                                ;  Read in 1st directory segment and set up info about directory parameters.
                                ;
011474  004737  013656'                 CALL    RDSEG1          ;READ IN 1ST DIRECTORY SEGMENT
                                ;
                                ;  Consolidate entries in directory segment.
                                ;
011500  004737  013016'         7$:     CALL    CONSOL          ;CONSOLIDATE DIRECTORY SEGMENT
                                ;
                                ;  Find next directory entry for a permanent file or a free slot.
                                ;
011504  063701  002006'         4$:     ADD     DIRSIZ,R1       ;POINT TO NEXT DIRECTORY ENTRY
011510  016100  000000G                 MOV     FD$STA(R1),R0   ;PICK UP STATUS WORD FROM DIRECTORY ENTRY
011514  032700  000000C                 BIT     #FS$PRM+FS$EMP+FS$EOS,R0;PERM FILE, EMPTY SLOT OR END OF SEGMENT?
011520  001771                         BEQ     4$              ;BR IF NONE OF ABOVE
011522  032700  000000G                 BIT     #FS$EMP,R0      ;IS THIS AN EMPTY SPACE ENTRY?
011526  001024                         BNE     13$             ;BR IF YES
011530  032700  000000G                 BIT     #FS$EOS,R0      ;IS THIS THE END OF SEGMENT MARKER ENTRY?
011534  001106                         BNE     1$              ;BR IF YES
                                ;
                                ;  Found permanent file entry.  See if this is a protected file entry
                                ;  with the same name as the file being entered.
                                ;
011536  032700  000000G                 BIT     #FS$PRO,R0      ;IS THIS FILE PROTECTED?
011542  001760                         BEQ     4$              ;BR IF NOT
011544  010103                         MOV     R1,R3           ;POINT TO NAME IN DIR ENTRY
011546  062703  000000G                 ADD     #FD$NAM,R3
011552  012704  000002G                 MOV     #FILSPC+2,R4    ;POINT TO NAME OF FILE BEING ENTERED
011556  012700  000003                 MOV     #3,R0           ;GET # WORDS TO COMPARE
011562  022324                 14$:    CMP     (R3)+,(R4)+     ;COMPARE FILE NAMES
011564  001347                         BNE     4$              ;BR IF DIFFERENT
011566  077003                         SOB     R0,14$          ;COMPARE ALL OF NAMES
                                ;  Error: Protected file with same name.
                                ;  Return error code 3.
```

```
     58 011570  012700  000003                    MOV     #3,R0           ;SET ERROR CODE
     59 011574  000137  012174'                    JMP     24$             ;RETURN ERROR STATUS
     60                                     ;
     61                                     ;  We found a free slot.  Check its size.
     62                                     ;
     63 011600  016100  000000G            13$:    MOV     FD$LEN(R1),R0   ;GET SIZE OF FREE SLOT
     64 011604  005702                             TST     R2              ;WHAT KIND OF ALLOCATION IS BEING REQUESTED?
     65 011606  001422                             BEQ     2$              ;BR IF HE WANTS DEFAULT ALGORITHM
     66 011610  020227  177777                     CMP     R2,#-1          ;DOES HE WANT LARGEST FREE SLOT?
     67 011614  001417                             BEQ     2$              ;BR IF YES
     68                                     ;  See if free slot is large enough to satisfy specific request.
     69 011616  020002                             CMP     R0,R2           ;IS FREE SLOT LARGE ENOUGH?
     70 011620  103731                             BLO     4$              ;BR IF NOT
     71 011622  013703  002030'                    MOV     L1SIZ,R3        ;GET SIZE OF BEST FIT SO FAR
     72 011626  001402                             BEQ     15$             ;BR IF THIS IS 1ST SLOT THAT WILL FIT
     73 011630  020003                             CMP     R0,R3           ;IS NEW SLOT A BETTER FIT THAN LAST ONE?
     74 011632  103324                             BHIS    4$              ;BR IF NOT
     75 011634  010037  002030'            15$:    MOV     R0,L1SIZ        ;REMEMBER SIZE OF BEST FIT FOUND SO FAR
     76 011640  010137  002032'                    MOV     R1,L1OFF        ;OFFSET OF DIR ENTRY WITHIN SEGMENT
     77 011644  013737  002002' 002034'            MOV     CURSEG,L1SEG    ;# OF DIR SEGMENT WITH ENTRY
     78 011652  000714                             BR      4$
     79                                     ;  We have a requested size of 0 or -1.
     80                                     ;  Remember two largest free slots.
     81 011654  020037  002030'            2$:     CMP     R0,L1SIZ        ;IS NEW SLOT LARGER THAN LARGEST WE'VE SEEN?
     82 011660  101421                             BLOS    5$              ;BR IF NOT
     83 011662  013737  002030' 002036'            MOV     L1SIZ,L2SIZ     ;MOVE INFO INTO 2ND LARGEST ENTRY
     84 011670  013737  002034' 002042'            MOV     L1SEG,L2SEG
     85 011676  013737  002032' 002040'            MOV     L1OFF,L2OFF
     86 011704  010037  002030'                    MOV     R0,L1SIZ        ;SAVE SIZE OF NEW LARGEST ENTRY
     87 011710  010137  002032'                    MOV     R1,L1OFF        ;SAVE ADDRESS OF DIR ENTRY
     88 011714  013737  002002' 002034'            MOV     CURSEG,L1SEG    ;SAVE # OF SEGMENT WITH ENTRY
     89 011722  000670                             BR      4$              ;KEEP LOOKING FOR LARGER FREE SLOTS
     90 011724  020037  002036'            5$:     CMP     R0,L2SIZ        ;IS NEW SLOT LARGER THAN 2ND LARGEST SO FAR?
     91 011730  101665                             BLOS    4$              ;BR IF NOT
     92 011732  010037  002036'                    MOV     R0,L2SIZ        ;SAVE NEW 2ND LARGEST SIZE
     93 011736  010137  002040'                    MOV     R1,L2OFF        ;SAVE ADDRESS OF DIR ENTRY
     94 011742  013737  002002' 002042'            MOV     CURSEG,L2SEG    ;SAVE DIR SEGMENT #
     95 011750  000655                             BR      4$              ;KEEP LOOKING
     96                                     ;
     97                                     ;  There are no more free slots in the current directory segment.
     98                                     ;  If after scanning all the segments to find the best fit entry, the
     99                                     ;  current segment ends up having the best fit (L1SEG or L2SEG), then
    100                                     ;  we will have to re-read the segment, but since it may have been
    101                                     ;  consolidated in memory, we have to write it out now so that the
    102                                     ;  offset to the desired entry (L1OFF or L2OFF) will be correct.
    103                                     ;
    104 011752  023737  002034' 002002'    1$:     CMP     L1SEG,CURSEG    ;IS THIS A POTENTIAL TARGET SEGMENT?
    105 011760  001404                             BEQ     30$             ;IF YES, MAY WANT TO REWRITE
    106 011762  023737  002042' 002002'            CMP     L2SEG,CURSEG    ;IS THIS AN ALTERNATE TARGET SEGMENT?
    107 011770  001005                             BNE     35$             ;IF WON'T USE THIS SEGMENT, SKIP REWRITE
    108 011772  105737  002050'            30$:    TSTB    CNSLXD          ;DID CONSOL REORGANIZE THE SEGMENT?
    109 011776  001402                             BEQ     35$             ;IF NOT, SKIP REWRITE
    110 012000  004737  014110'                    CALL    WRTSEG          ;REWRITE SEGMENT
    111                                     ;
    112                                     ;  See if there are more segments to check.
    113                                     ;
    114 012004  013700  000000C            35$:    MOV     USRBUF+DH$NXT,R0;GET # OF NEXT SEGMENT
```

```
    115 012010  001403                          BEQ     6$              ;BR IF THERE ARE NO MORE SEGMENTS
    116 012012  004737  013772'                 CALL    RDSEG           ;READ IN NEXT SEGMENT
    117 012016  000630                          BR      7$              ;SEARCH THIS SEGMENT
    118                                 ;
    119                                 ;  We have reached the end of the last active directory segment.
    120                                 ;  See if we were able to satisfy request.
    121                                 ;
    122 012020  005737  002030'         6$:     TST     L1SIZ           ;DID WE FIND ANY ENTRY THAT WOULD DO?
    123 012024  001461                          BEQ     9$              ;BR IF NOT
    124 012026  005702                          TST     R2              ;WHAT KIND OF REQUEST WAS IT?
    125 012030  001412                          BEQ     8$              ;DEFAULT ALLOCATION
    126                                 ;  User wants largest free slot or he specified a specific size.
    127 012032  020227  177777                  CMP     R2,#-1          ;DOES HE WANT LARGEST OR SPECIFIC SIZE?
    128 012036  001002                          BNE     10$             ;BR IF HE SPECIFIED EXACT AMT HE WANTS
    129 012040  013702  002030'                 MOV     L1SIZ,R2        ;RETURN SIZE OF LARGEST FREE SLOT
    130 012044  013700  002034'         10$:    MOV     L1SEG,R0        ;GET SEGMENT # WITH ENTRY
    131 012050  013701  002032'                 MOV     L1OFF,R1        ;GET ADDRESS OF ENTRY
    132 012054  000435                          BR      23$             ;GO REREAD SEGMENT WITH EMPTY ENTRY
    133                                 ;  User wants default allocation.
    134                                 ;  (i.e., Larger of (1/2 largest slot  or  2nd largest slot) )
    135 012056  013702  002030'         8$:     MOV     L1SIZ,R2        ;GET SIZE OF LARGEST SLOT
    136 012062  020227  000001                  CMP     R2,#1           ;IS LARGEST SLOT 1 BLOCK LONG?
    137 012066  001414                          BEQ     11$             ;IF SO THEN USE ALL OF IT
    138 012070  000241                          CLC                     ;DIVIDE BY 2
    139 012072  006002                          ROR     R2
    140 012074  020237  002036'                 CMP     R2,L2SIZ        ;COMPARE TO 2ND LARGEST SLOT
    141 012100  101007                          BHI     11$             ;USE 1/2 OF LARGEST
    142 012102  013700  002042'                 MOV     L2SEG,R0        ;GET # OF SEGMENT WITH 2ND LARGEST FREE BLOCK
    143 012106  013701  002040'                 MOV     L2OFF,R1        ;GET OFFSET TO ENTRY WITHIN SEGMENT
    144 012112  013702  002036'                 MOV     L2SIZ,R2        ;GET SIZE OF ENTRY
    145 012116  000404                          BR      12$
    146 012120  013700  002034'         11$:    MOV     L1SEG,R0        ;GET # OF SEGMENT WITH LARGEST FREE BLOCK
    147 012124  013701  002032'                 MOV     L1OFF,R1        ;GET OFFSET TO ENTRY WITHIN SEGMENT
    148                                 ;
    149                                 ;  Constrain the largest file size returned in response to a 0-size request
    150                                 ;  to a sysgen supplied parameter (MAXFIL).
    151                                 ;
    152 012130  005737  000000G         12$:    TST     VMXFIL          ;Did user specifiy a size to constrain alloc?
    153 012134  001405                          BEQ     23$             ;Br if not
    154 012136  020237  000000G                 CMP     R2,VMXFIL       ;IS FREE SLOT LARGER THAN MAXFIL PARAMETER?
    155 012142  101402                          BLOS    23$             ;BR IF NOT
    156 012144  013702  000000G                 MOV     VMXFIL,R2       ;ONLY USE THIS MUCH SPACE
    157                                 ;
    158                                 ;  Reread the segment that contains the empty entry we are going to
    159                                 ;  use, unless that segment is the current segment.
    160                                 ;
    161 012150  020037  002002'         23$:    CMP     R0,CURSEG       ;IS SEG WE WANT THE CURRENT SEGMENT?
    162 012154  001411                          BEQ     21$             ;BR IF YES
    163 012156  010146                          MOV     R1,-(SP)        ;SAVE ADDRESS OF DIRECTORY ENTRY
    164 012160  004737  013772'                 CALL    RDSEG           ;REREAD THE SEGMENT WITH THE EMPTY ENTRY
    165 012164  012601                          MOV     (SP)+,R1        ;RETRIEVE ADDRESS OF DIRECTORY ENTRY
    166 012166  000404                          BR      21$             ;RETURN SUCCESSFULLY
    167                                 ;
    168                                 ;  We could not satisfy the request.
    169                                 ;
    170 012170  012700  000001         9$:     MOV     #1,R0           ;RETURN ERROR CODE 1
    171 012174  000261                 24$:    SEC                     ;SIGNAL ERROR ON RETURN
```

```
    172 012176  000401                          BR      22$
    173                                 ;
    174                                 ;  Successful return
    175                                 ;
    176 012200  000241                  21$:    CLC                     ;SIGNAL GOOD RETURN
    177 012202  012604                  22$:    MOV     (SP)+,R4
    178 012204  012603                          MOV     (SP)+,R3
    179 012206  000207                          RETURN
```

```
                                        .SBTTL  ADDENT -- Add a tentative file entry to directory
                                ;-------------------------------------------------------------------------
                                ;  ADDENT is called to add a tentative file entry to the current
                                ;  directory segment.  This may cause the directory segment to be
                                ;  split if it overflows.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of empty-file directory entry to be converted to tentative file.
                                ;   R2 = # blocks to be allocated to tentative file.
                                ;   FILSPC = File spec for file being created.
                                ;
                                ;  Outputs:
                                ;   R1 = Address of tentative file directory entry.
                                ;   C-flag set if directory overflows.
                                ;
012210  010246                  ADDENT: MOV     R2,-(SP)
012212  010346                          MOV     R3,-(SP)
                                ;
                                ;  See if there is room in the current directory segment for three new entries
                                ;  (free-0, tentative, free-remainder)
                                ;
012214  010146                          MOV     R1,-(SP)        ;SAVE ADDRESS OF ENTRY WE ARE WORKING ON
012216  005000                          CLR     R0              ;FIND END OF SEGMENT ENTRY
012220  004737  013424'                 CALL    NXTDIR
012224  010103                          MOV     R1,R3           ;GET ADDRESS OF END OF SEGMENT ENTRY
012226  012601                          MOV     (SP)+,R1        ;GET BACK ADDRESS OF EMPTY FILE ENTRY
012230  013700  002006'                 MOV     DIRSIZ,R0       ;GET SIZE OF A DIRECTORY ENTRY
012234  060003                          ADD     R0,R3           ;ADD SPACE NEEDED FOR 3 ENTRIES
012236  060003                          ADD     R0,R3
012240  060003                          ADD     R0,R3
012242  020327  002002'                 CMP     R3,#USRBUF+1024.;IS THERE ROOM IN THIS SEGMENT?
012246  103403                          BLO     1$              ;BR IF THERE IS ROOM
                                ;
                                ;  Directory segment is full.
                                ;  Attempt to split it.
                                ;
012250  004737  012510'                 CALL    SPLIT           ;SPLIT THE SEGMENT
012254  103461                          BCS     9$              ;BR IF DIRECTORY OVERFLOW
                                ;
                                ;  There is room in the current directory segment for the new entries.
                                ;  Convert the empty file entry to a tentative file entry.
                                ;
012256  016100  000000G         1$:     MOV     FD$LEN(R1),R0   ;GET SIZE OF EMPTY FILE ENTRY
012262  010261  000000G                 MOV     R2,FD$LEN(R1)   ;PUT IN ALLOCATED SIZE OF TENTATIVE FILE
012266  160200                          SUB     R2,R0           ;CALC # BLOCKS LEFT OVER IN EMPTY AREA
012270  010002                          MOV     R0,R2
012272  012761  000000G 000000G         MOV     #FS$TEN,FD$STA(R1);MARK ENTRY AS TENTATIVE
012300  113761  000000G 000000G         MOVB    CORUSR,FD$JOB(R1);PUT IN JOB NUMBER
012306  113761  000000G 000000G         MOVB    CHNNUM,FD$CHN(R1);PUT IN CHANNEL NUMBER
012314  005061  000000G                 CLR     FD$DAT(R1)      ;SAY NO CREATION DATE YET -- SET IN CLOSE
012320  010103                          MOV     R1,R3           ;SAVE ADDRESS OF TENTATIVE ENTRY
012322  062703  000000G                 ADD     #FD$NAM,R3      ;POINT TO FIELD WHERE NAME GOES
012326  012700  000002G                 MOV     #FILSPC+2,R0    ;POINT TO FILE SPEC WE ARE CREATING FOR
012332  012023                          MOV     (R0)+,(R3)+     ;MOVE IN FILE NAME
012334  012023                          MOV     (R0)+,(R3)+
012336  012023                          MOV     (R0)+,(R3)+     ;AND EXTENSTION
                                ;
```

```
                                        ;  Add an empty file entry following the tentative file entry to hold
                                        ;  any left over blocks.
                                        ;
012340  063701  002006'                         ADD     DIRSIZ,R1       ;POINT TO FOLLOWING DIR ENTRY
012344  004737  012426'                         CALL    INSERT          ;INSERT AN EMPTY FILE ENTRY
012350  010261  000000G                         MOV     R2,FD$LEN(R1)   ;FILL IN # LEFT OVER BLOCKS
                                        ;
                                        ;  If the empty file entry we turned into a tentative file immediately followed
                                        ;  a tentative file entry, add a zero length empty file entry in front of
                                        ;  our new tentative file entry (this entry may be used when the earlier
                                        ;  tentative file entry is closed).
                                        ;
012354  163701  002006'                         SUB     DIRSIZ,R1       ;POINT TO OUR NEW TENTATIVE FILE ENTRY
012360  163701  002006'                         SUB     DIRSIZ,R1       ;POINT TO PREVIOUS ENTRY
012364  020127  000000C                         CMP     R1,#USRBUF+DH$$SZ;WAS THERE A PREVIOUS ENTRY?
012370  103410                                  BLO     3$              ;BR IF NOT
012372  032761  000000G 000000G                 BIT     #FS$TEN,FD$STA(R1);WAS PREVIOUS ENTRY A TENTATIVE FILE ENTRY?
012400  001404                                  BEQ     3$              ;BR IF NOT
012402  063701  002006'                         ADD     DIRSIZ,R1       ;POINT BACK TO OUR NEW TENTATIVE ENTRY
012406  004737  012426'                         CALL    INSERT          ;INSERT A ZERO-LENGTH EMPTY FILE ENTRY
012412  063701  002006'                 3$:     ADD     DIRSIZ,R1       ;POINT TO TENTATIVE FILE ENTRY WE CREATED
                                        ;
                                        ;  Finished
                                        ;
012416  000241                                  CLC                     ;SIGNAL GOOD RETURN
012420  012603                          9$:     MOV     (SP)+,R3
012422  012602                                  MOV     (SP)+,R2
012424  000207                                  RETURN
```

```
                                        .SBTTL  INSERT -- Add an empty file entry to directory
                                ;-------------------------------------------------------------------------
                                ;  INSERT is called to create a new empty file directory entry and insert
                                ;  it into the current directory segment.
                                ;  Note:  Tests before calling us guarantee that there is room in the
                                ;         current directory segment.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of entry new empty file entry is to go in front of.
                                ;
                                ;  Outputs:
                                ;   R1 = Address of new empty file entry.
                                ;
012426  010146                  INSERT: MOV     R1,-(SP)
                                ;
                                ;  Locate end-of-segment entry.
                                ;
012430  012700  000000G                 MOV     #FS$EOS,R0      ;SEARCH FOR END-OF-SEGMENT MARKER
012434  030061  000000G                 BIT     R0,FD$STA(R1)   ;ARE WE POINTING TO IT NOW?
012440  001002                          BNE     2$              ;BR IF YES
012442  004737  013424'                 CALL    NXTDIR          ;FIND END OF SEGMENT ENTRY
                                ;
                                ;  Shove down all entries beyond insert point.
                                ;
012446  005721                  2$:     TST     (R1)+           ;POINT BEYOND END-OF-SEGMENT WORD
012450  010100                          MOV     R1,R0
012452  063700  002006'                 ADD     DIRSIZ,R0       ;MAKE ROOM FOR NEW ENTRY
012456  014140                  1$:     MOV     -(R1),-(R0)     ;SHOVE DOWN ALL ENTRIES
012460  020116                          CMP     R1,(SP)         ;TILL WE REACH INSERT POINT
012462  101375                          BHI     1$
                                ;
                                ;  Initialize the new entry.
                                ;
012464  013700  002006'                 MOV     DIRSIZ,R0       ;SIZE OF ENTRY (BYTES)
012470  006200                          ASR     R0              ;GET # WORDS
012472  005021                  3$:     CLR     (R1)+           ;ZERO THE ENTRY
012474  077002                          SOB     R0,3$
012476  012601                          MOV     (SP)+,R1        ;GET ADDRESS OF NEW ENTRY
012500  012761  000000G 000000G         MOV     #FS$EMP,FD$STA(R1);SAY ENTRY IS EMPTY
                                ;
                                ;  Finished
                                ;
012506  000207                          RETURN
```

```
                                        .SBTTL  SPLIT  -- Split a directory segment
                                ;-------------------------------------------------------------------------
                                ;  SPLIT is called to split a directory segment when it overflows.
                                ;  SPLIT moves approximately half of the entries in the current
                                ;  segment to a new segment and links the new segment into the directory
                                ;  chain.  The number of open directory segments (stored in segment # 1)
                                ;  is updated also.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of a directory entry of interest in current segment.
                                ;
                                ;  Outputs:
                                ;   R1 = Address of directory entry of interest after split.
                                ;   USRBUF = Directory segment containing directory entry pointed to by R1.
                                ;   C-flag set on return if there are no available free directory segments.
                                ;
012510  010246                  SPLIT:  MOV     R2,-(SP)
012512  010346                          MOV     R3,-(SP)
012514  010446                          MOV     R4,-(SP)
012516  010546                          MOV     R5,-(SP)
012520  010105                          MOV     R1,R5           ;SAVE ADDRESS OF ENTRY WE ARE INTERESTED IN
                                ;
                                ;  See if there is a free segment for us to split into.
                                ;
012522  023737  002010' 002004'         CMP     DIRNSG,DIRHIS   ;IS THERE A FREE SEGMENT?
012530  101524                          BLOS    10$             ;BR IF NOT
012532  013704  002002'                 MOV     CURSEG,R4       ;REMEMBER CURRENT DIR SEGMENT #
                                ;
                                ;  There is a free segment.
                                ;  Determine where to split this segment.
                                ;
                                ;  Count # of entries in this segment.
                                ;
012536  012701  000000C                 MOV     #USRBUF+DH$$SZ,R1;POINT TO FIRST ENTRY IN SEGMENT
012542  005000                          CLR     R0              ;COUNT # ENTRIES IN R0
012544  032761  000000G 000000G 1$:     BIT     #FS$EOS,FD$STA(R1);IS THIS THE END OF SEGMENT MARKER?
012552  001004                          BNE     2$              ;BR IF YES
012554  005200                          INC     R0              ;COUNT ANOTHER ENTRY
012556  063701  002006'                 ADD     DIRSIZ,R1       ;POINT TO NEXT ENTRY
012562  000770                          BR      1$
                                ;  Leave approximately half the entries in the current segment.
012564  006200                  2$:     ASR     R0              ;GET 1/2 # ENTRIES
012566  013702  000000C                 MOV     USRBUF+DH$BLK,R2;GET BASE BLOCK # OF 1ST FILE IN SEGMENT
012572  012701  000000C                 MOV     #USRBUF+DH$$SZ,R1;POINT TO 1ST ENTRY
012576  066102  000000G         3$:     ADD     FD$LEN(R1),R2   ;KEEP TRACK OF BLOCK #'S OF FILES
012602  063701  002006'                 ADD     DIRSIZ,R1       ;POINT TO NEXT DIRECTORY ENTRY
012606  077005                          SOB     R0,3$           ;SKIP OVER 1/2 OF THE ENTRIES
                                ;  Split on 1st permanent or tentative entry beyond mid-point.
012610  032761  000000C 000000G 4$:     BIT     #<FS$PRM+FS$TEN>,FD$STA(R1);IS THIS A PERM OR TENTATIVE ENTRY?
012616  001005                          BNE     7$              ;BR IF YES
012620  066102  000000G                 ADD     FD$LEN(R1),R2   ;KEEP TRACK OF STARTING BLOCK NUMBERS
012624  063701  002006'                 ADD     DIRSIZ,R1       ;POINT TO NEXT ENTRY
012630  000767                          BR      4$
                                ;
                                ;  Found split point.
                                ;  R1 = Address of 1st entry to go in new segment.
                                ;  R2 = Base block number of 1st file in new segment.
```

```
                                        ;
                                        ;  Calculate address of dir entry we are interested in after split.
012632  020501                  7$:       CMP     R5,R1           ;WILL ENTRY GO IN OLD OR NEW SEGMENT?
012634  103404                            BLO     5$              ;BR IF GOING IN OLD SEGMENT
012636  160105                            SUB     R1,R5           ;CALCULATE ITS ADDRESS IN NEW SEGMENT
012640  062705  000000C                   ADD     #USRBUF+DH$$SZ,R5
012644  005004                            CLR     R4              ;REMEMBER ENTRY GOES IN NEW SEGMENT
                                        ;
                                        ;  Truncate current segment and set link to new segment.
                                        ;
012646  011146                  5$:       MOV     (R1),-(SP)      ;SAVE FD$STAT FOR ENTRY FOLLOWING SPLIT
012650  012711  000000G                   MOV     #FS$EOS,(R1)    ;SET END-OF-SEGMENT MARKER
012654  013746  000000C                   MOV     USRBUF+DH$NXT,-(SP);SAVE LINK TO NEXT SEGMENT
012660  013703  002004'                   MOV     DIRHIS,R3       ;GET # OF HIGHEST SEGMENT CURRENTLY IN USE
012664  005203                            INC     R3              ;GET # OF 1ST FREE SEGMENT
012666  010337  000000C                   MOV     R3,USRBUF+DH$NXT;SET AS NEW LINK
012672  004737  014110'                   CALL    WRTSEG          ;WRITE OUT TRUNCATED OLD SEGMENT
                                        ;
                                        ;  Now set up new segment.
                                        ;
                                        ;  Set new header.
012676  012637  000000C                   MOV     (SP)+,USRBUF+DH$NXT;SET FLINK
012702  010337  002002'                   MOV     R3,CURSEG       ;SET OUR SEGMENT #
012706  010237  000000C                   MOV     R2,USRBUF+DH$BLK;STARTING BLOCK # FOR FILES IN SEGMENT
                                        ;  Slide up directory entries from end of segment.
012712  012611                            MOV     (SP)+,(R1)      ;FIX FD$STA OF 1ST ENTRY
012714  012700  000000C                   MOV     #USRBUF+DH$$SZ,R0;POINT TO TOP OF AREA FOR ENTRIES
012720  012702  002002'                   MOV     #USRBUF+1024.,R2;POINT PAST END OF BUFFER
012724  160102                            SUB     R1,R2           ;GET # BYTES TO MOVE
012726  006202                            ASR     R2              ;GET # WORDS TO MOVE
012730  012120                  6$:       MOV     (R1)+,(R0)+     ;MOVE UP ENTRIES
012732  077202                            SOB     R2,6$
                                        ;  Write out the new segment.
012734  004737  014110'                   CALL    WRTSEG
                                        ;
                                        ;  Update information in header of 1st (master) directory segment.
                                        ;
012740  004737  013656'                   CALL    RDSEG1          ;READ IN DIR SEG 1
012744  010337  000000C                   MOV     R3,USRBUF+DH$HIS;SET # OF HIGHEST SEG IN USE
012750  010337  002004'                   MOV     R3,DIRHIS
012754  004737  014110'                   CALL    WRTSEG          ;WRITE OUT SEG # 1
                                        ;
                                        ;  Now read back in the segment that has the directory entry we are
                                        ;  interested in.
                                        ;
012760  005704                            TST     R4              ;IS ENTRY IN OLD OR NEW DIRECTORY SEGMENT?
012762  001401                            BEQ     9$              ;BR IF IN NEW SEGMENT
012764  010403                            MOV     R4,R3           ;GET NUMBER OF OLD SEGMENT
012766  010300                  9$:       MOV     R3,R0           ;SET AS ARGUMENT TO RDSEG
012770  004737  013772'                   CALL    RDSEG           ;READ IN SEGMENT WITH ENTRY WE WANT
                                        ;
                                        ;  Finished
                                        ;
012774  010501                            MOV     R5,R1           ;GET ADDRESS OF ENTRY OF INTEREST
012776  000241                            CLC                     ;SIGNAL GOOD RETURN
013000  000401                            BR      11$
                                        ;
```

```
                                        ;  Directory overflow
                                        ;
013002  000261                  10$:    SEC                             ;SIGNAL ERROR ON RETURN
013004  012605                  11$:    MOV     (SP)+,R5
013006  012604                          MOV     (SP)+,R4
013010  012603                          MOV     (SP)+,R3
013012  012602                          MOV     (SP)+,R2
013014  000207                          RETURN
```

```
                                        .SBTTL  CONSOL -- Directory segment consolidator
                                ;-------------------------------------------------------------------------
                                ;  CONSOL is called to remove unnecessary entries from the current directory
                                ;  segment.  The unnecessary entries that are removed are:
                                ;   1. Tentative files that are not currently open.
                                ;   2. Multiple consecutive empty entries.
                                ;   3. Empty entries of length 0 that follow a permanent entry.
                                ;
                                ;  Inputs:
                                ;   USRBUF = Current directory segment.
                                ;
013016  010146                  CONSOL: MOV     R1,-(SP)
013020  010246                          MOV     R2,-(SP)
013022  010346                          MOV     R3,-(SP)
013024  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get exclusive access to job context block buffer
                                ;
013026                                  OCALL   GETCXT          ;Get exclusive access to context block buffer
013034  105037  002050'                 CLRB    CNSLXD          ;Say we haven't changed anything yet
                                ;
                                ;  Pass 1:  Convert unopen tentative entries into empty entries.
                                ;
013040  012701  000000G                 MOV     #USRBUF+DH$$SZ,R1;POINT TO 1ST ENTRY
013044  013704  002006'                 MOV     DIRSIZ,R4       ;CARRY DIRECTORY ENTRY SIZE IN R4
013050  160401                          SUB     R4,R1           ;POINT IN FRONT OF 1ST ENTRY FOR NXTDIR
                                ;  Find the next tentative file entry.
013052  012700  000000G         3$:     MOV     #FS$TEN,R0      ;LOOK FOR A TENTATIVE FILE ENTRY
013056  004737  013424'                 CALL    NXTDIR          ;DO SEARCH
013062  103440                          BCS     5$              ;BR IF NO MORE TENTATIVE FILE ENTRIES
                                ;  We found a tentative file entry.
                                ;  R1 points to the entry.
                                ;  See if the entry is open now.
013064  116102  000000G                 MOVB    FD$JOB(R1),R2   ;GET # OF JOB USING ENTRY
013070  001427                          BEQ     1$              ;BR IF NOT TSX JOB
013072  032702  000001                  BIT     #1,R2           ;TSX JOB NUMBER CAN NEVER BE ODD
013076  001024                          BNE     1$              ;BR IF NOT TSX JOB USING CHANNEL
013100  020227  000000G                 CMP     R2,#LSTSL       ;MAKE SURE JOB NUMBER IS NOT TOO BIG
013104  101021                          BHI     1$              ;BR IF NOT TSX JOB
013106  032762  000000G 000000G         BIT     #$DILUP,LSW(R2) ;IS THAT JOB LOGGED ON?
013114  001415                          BEQ     1$              ;BR IF NOT
013116  116103  000000G                 MOVB    FD$CHN(R1),R3   ;GET CHANNEL # ASSOCIATED WITH FILE
013122                                  OCALL   GETCHA          ;GET POINTER TO CHANNEL BLOCK
013130  032760  000000G 000000G         BIT     #CS$OPN,C.CSW(R0);IS THAT CHANNEL OPEN?
013136  001404                          BEQ     1$              ;BR IF NOT
013140  032760  000000G 000000G         BIT     #CS$ENT,C.CSW(R0);IS CHANNEL OPEN FOR CREATING A NEW FILE?
013146  001005                          BNE     2$              ;BR IF YES
                                ;  Convert this tentative file entry to an empty file entry.
013150  012761  000000G 000000G 1$:     MOV     #FS$EMP,FD$STA(R1);SAY THIS IS AN EMPTY FILE ENTRY
013156  105237  002050'                 INCB    CNSLXD          ;REMEMBER WE HAVE CHANGED THE SEGMENT
                                ;  Check next entry.
013162  000733                  2$:     BR      3$
```

```
                                        ;
                                        ;  Pass 2:  Consolidate consecutive empty entries and emtpy entries
                                        ;           of zero length preceded by a permanent file entry.
                                        ;
013164  012701  000000C         5$:     MOV     #USRBUF+DH$$SZ,R1;POINT TO 1ST ENTRY
013170  160401                          SUB     R4,R1           ;POINT IN FRONT OF 1ST ENTRY
013172  012700  000000G         11$:    MOV     #FS$EMP,R0      ;SEARCH FOR NEXT EMTPY FILE ENTRY
013176  004737  013424'                 CALL    NXTDIR
013202  103462                          BCS     12$             ;BR IF THERE ARE NO MORE
                                        ;  We have found an empty file entry.
                                        ;  Combine it with the following entry if it is empty too.
013204  010102                  7$:     MOV     R1,R2           ;GET ADDRESS OF CURRENT ENTRY
013206  060402                          ADD     R4,R2           ;POINT TO NEXT ENTRY
013210  032762  000000G 000000G         BIT     #FS$EMP,FD$STA(R2);IS NEXT ENTRY EMPTY TOO?
013216  001421                          BEQ     8$              ;BR IF NOT
                                        ;  Combine the two entries.
013220  066261  000000G 000000G         ADD     FD$LEN(R2),FD$LEN(R1);COMBINE ALLOCATED SIZES
013226  105237  002050'                 INCB    CNSLXD          ;REMEMBER WE HAVE CHANGED THE SEGMENT
                                        ;  Remove the second empty entry.
013232  010146                          MOV     R1,-(SP)        ;REMEMBER ADDRESS OF 1ST EMPTY ENTRY
013234  012700  000000G                 MOV     #FS$EOS,R0      ;SEARCH FOR END OF SEGMENT MARKER
013240  004737  013424'                 CALL    NXTDIR
                                        ;  R1 now points to end of segment marker.
                                        ;  Move up entries over the one being removed (pointed to by R2).
013244  010200                          MOV     R2,R0           ;GET ADDRESS OF ENTRY TO BE REMOVED
013246  060400                          ADD     R4,R0           ;POINT TO FOLLOWING ENTRY
013250  012022                  6$:     MOV     (R0)+,(R2)+     ;MOVE UP REST OF ENTRIES OVER ONE BEING REMVD
013252  020001                          CMP     R0,R1           ;MOVED END-OF-SEGMENT MARKER YET?
013254  101775                          BLOS    6$              ;BR IF NOT
013256  012601                          MOV     (SP)+,R1        ;GET BACK ADDRESS OF 1ST EMPTY ENTRY
013260  000751                          BR      7$              ;SEE IF THERE ARE MORE EMPTIES TO BE MERGED
                                        ;
                                        ;  This empty is not followed by any other empty entries.
                                        ;  See if it is of zero length and follows a permanent file entry.
                                        ;
013262  005761  000000G         8$:     TST     FD$LEN(R1)      ;IS THIS EMPTY OF ZERO LENGTH?
013266  001027                          BNE     9$              ;BR IF NOT
013270  010102                          MOV     R1,R2           ;GET ADDRESS OF ENTRY
013272  160402                          SUB     R4,R2           ;GET ADDRESS OF PREVIOUS ENTRY
013274  020227  000000C                 CMP     R2,#USRBUF+DH$$SZ;ARE WE 1ST ENTRY?
013300  103422                          BLO     9$              ;BR IF YES
013302  032762  000000G 000000G         BIT     #FS$PRM,FD$STA(R2);IS PREVIOUS ENTRY A PERMANENT FILE?
013310  001416                          BEQ     9$              ;BR IF NOT
                                        ;  Remove a zero length empty entry following a permanent file entry.
013312  010246                          MOV     R2,-(SP)        ;SAVE ADDRESS OF PERMANENT FILE ENTRY
013314  010102                          MOV     R1,R2           ;GET ADDRESS OF EMPTY ENTRY
013316  012700  000000G                 MOV     #FS$EOS,R0      ;SEARCH FOR END OF SEGMENT ENTRY
013322  004737  013424'                 CALL    NXTDIR
013326  010200                          MOV     R2,R0           ;GET ADDRESS OF EMPTY ENTRY
013330  060400                          ADD     R4,R0           ;POINT TO FOLLOWING ENTRY
013332  012022                  10$:    MOV     (R0)+,(R2)+     ;MOVE UP FOLLOWING ENTRIES OVER EMPTY ENTRY
013334  020001                          CMP     R0,R1           ;HAVE WE MOVED END-OF-SEGMENT MARKER YET?
013336  101775                          BLOS    10$             ;BR IF NOT
013340  105237  002050'                 INCB    CNSLXD          ;REMEMBER WE HAVE CHANGED THIS SEGMENT
013344  012601                          MOV     (SP)+,R1        ;GET BACK ADDRESS OF PERMANENT FILE ENTRY
                                        ;  Look for next empty entry.
013346  000711                  9$:     BR      11$
```

```
                                        ;
                                        ;  Finished
                                        ;
013350                          12$:    OCALL   FRECXT          ;Release context block buffer
013356  012604                          MOV     (SP)+,R4
013360  012603                          MOV     (SP)+,R3
013362  012602                          MOV     (SP)+,R2
013364  012601                          MOV     (SP)+,R1
013366  000207                          RETURN
```

```
                                        .SBTTL  GETDIR -- Locate next directory entry
                                ;-----------------------------------------------------------------------
                                ;  GETDIR is called to get the address of the next directory entry of
                                ;  a specified type.  Directory segments are read if necessary to find
                                ;  the next such entry.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of last directory entry.
                                ;   R0 = FS$xxx entry type flags.
                                ;   USRBUF = Current directory segment.
                                ;
                                ;  Outputs:
                                ;   R1 = Address of next directory entry of the specified type.
                                ;   USRBUF = Directory segment that contains the entry.
                                ;   C-flag set if no entry of the specified type can be found.
                                ;
013370                          GETDIR:
                                ;
                                ;  See if we can find entry of specified type in the current directory segment.
                                ;
013370  004737  013424'         1$:     CALL    NXTDIR          ;SEARCH FOR ENTRY OF SPECIFIED TYPE
013374  103012                          BCC     9$              ;BR IF WE FOUND ONE
                                ;
                                ;  Hit end of current directory segment.  See if there are more segments.
                                ;
013376  010046                          MOV     R0,-(SP)
013400  013700  000000C                 MOV     USRBUF+DH$NXT,R0;GET # OF NEXT SEGMENT IN LIST
013404  001404                          BEQ     2$              ;BR IF REACHED END
013406  004737  013772'                 CALL    RDSEG           ;READ IN NEXT SEGMENT
013412  012600                          MOV     (SP)+,R0        ;GET BACK TYPE FLAGS
013414  000765                          BR      1$              ;SEARCH THIS SEGMENT
                                ;
                                ;  No entry of the specified type was found.
                                ;
013416  012600                  2$:     MOV     (SP)+,R0        ;CLEAN OFF STACK
013420  000261                          SEC                     ;SIGNAL FAILURE ON RETURN
                                ;
                                ;  finished
                                ;
013422  000207                  9$:     RETURN
```

```
                                        .SBTTL  NXTDIR -- Locate next directory entry in current segment
                                ;-------------------------------------------------------------------------
                                ;  NXTDIR is called to get the address of the next directory entry of a
                                ;  specified type within the current directory segment.
                                ;
                                ;  Inputs:
                                ;   R0 = FS$xxx directory entry type flags.
                                ;   R1 = Address of last directory entry.
                                ;
                                ;  Outputs:
                                ;   R1 = Address of directory entry found.
                                ;   C-flag set if no entry of specified type could be found.
                                ;
013424  063701  002006'         NXTDIR: ADD     DIRSIZ,R1       ;POINT TO NEXT DIRECTORY ENTRY
013430  030061  000000G                 BIT     R0,FD$STA(R1)   ;IS THIS ENTRY OF THE RIGHT TYPE?
013434  001006                         BNE     1$              ;BR IF YES
013436  032761  000000G 000000G        BIT     #FS$EOS,FD$STA(R1);IS THIS THE END-OF-SEGMENT MARKER?
013444  001767                         BEQ     NXTDIR          ;KEEP SEARCHING IF NOT
                                ;  Hit end of segment.
013446  000261                         SEC                     ;SIGNAL FAILURE ON RETURN
013450  000401                         BR      9$
                                ;  Found entry.
013452  000241                 1$:     CLC                     ;SIGNAL SUCCESS ON RETURN
013454  000207                 9$:     RETURN
```

```
                                        .SBTTL  DIDDLE -- Allow user to access directory entry
                                ;-------------------------------------------------------------------------
                                ;  DIDDLE is called to give the user (especially PIP) a chance to access
                                ;  and possible alter a directory entry.
                                ;  A program requests this feature by storing the address of a routine to
                                ;  be called in the fifth word of the argument block for .ENTER, .LOOKUP,
                                ;  or .RENAME emt's and setting the name argument word odd.
                                ;  The specified routine is called with R1 pointing to the date word of
                                ;  the directory entry.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of the directory entry.
                                ;   EMTBLK = Argument list for the emt.
                                ;
                                ;  Outputs:
                                ;   Directory entry may have been modified by the user.
                                ;
013456  032737  000001  000002G DIDDLE: BIT     #1,EMTBLK+2     ;DOES USER WANT TO DIDDLE WITH THE ENTRY?
013464  001455                         BEQ     9$              ;BR IF NOT
013466  010246                         MOV     R2,-(SP)
013470  010346                         MOV     R3,-(SP)
013472  010446                         MOV     R4,-(SP)
013474  010546                         MOV     R5,-(SP)
013476  010146                         MOV     R1,-(SP)        ;SAVE ADDRESS OF DIR ENTRY ON TOP OF STACK
                                ;
                                ;  User does want to diddle with the directory entry.
                                ;  Move directory entry to user's address space.
                                ;
013500  012703  000300                 MOV     #USRUCA,R3      ;GET ADDRESS OF AREA IN USER'S AREA
013504  012700  000000C                MOV     #FD$$SZ/2,R0    ;GET # WORDS TO MOVE
013510  012146                 1$:     MOV     (R1)+,-(SP)     ;PICK UP WORD FROM DIRECTORY ENTRY
013512  106623                         MTPD    (R3)+           ;MOVE TO AREA IN USER'S SPACE
013514  077003                         SOB     R0,1$           ;MOVE ALL OF DIRECTORY ENTRY
                                ;
                                ;  Use special EMT to get control back from user's routine.
                                ;
013516  013703  000000G                MOV     EMTCAD,R3       ;Get pointer to top of return addr stack
013522  012743  013566'                MOV     #5$,-(R3)       ;Push our return address
013526  010337  000000G                MOV     R3,EMTCAD       ;Save updated stack pointer
013532  106506                         MFPD    SP              ;GET USER'S STACK POINTER
013534  012603                         MOV     (SP)+,R3
013536  012746  000000G                MOV     #CPLEMT,-(SP)   ;PUSH ADDR OF EMT INSTRUCTION ON USER'S STACK
013542  106643                         MTPD    -(R3)
013544  010346                         MOV     R3,-(SP)        ;NOW STORE UPDATED USER STACK POINTER
013546  106606                         MTPD    SP
                                ;
                                ;  Now enter user's routine in user mode.
                                ;  (Use RTI to do this)
                                ;  On entry to user's routine, R1 points to the directory entry.
                                ;
013550  012701  000300G                MOV     #USRUCA+FD$DAT,R1;MAKE R1 POINT TO DATE WORD OF DIRECTORY ENTRY
013554  013746  000000G                MOV     EMTPS,-(SP)     ;PS
013560  013746  000010G                MOV     EMTBLK+10,-(SP) ;PC = Address of user's routine
013564  000002                         RTI                     ;ENTER USER'S ROUTINE
                                ;
                                ;  Return here from user's routine (when EMT is done).
                                ;
```

```
                                        ;  Move directory entry back to directory buffer.
                                        ;
013566  011602                  5$:     MOV     (SP),R2         ;GET ADDRESS WHERE WE SHOULD PUT DIR ENTRY
013570  012703  000300                  MOV     #USRUCA,R3      ;POINT TO AREA IN USER'S AREA WITH DIR ENTRY
013574  012700  000000C                 MOV     #FD$$SZ/2,R0    ;GET # WORDS TO MOVE
013600  106523                  2$:     MFPD    (R3)+           ;GET A WORD FROM USER'S DIRECTORY ENTRY
013602  012622                          MOV     (SP)+,(R2)+     ;MOVE BACK TO OUR INTERNAL AREA
013604  077003                          SOB     R0,2$           ;MOVE ALL OF ENTRY
                                        ;
                                        ;  Finished
                                        ;
013606  012601                          MOV     (SP)+,R1
013610  012605                          MOV     (SP)+,R5
013612  012604                          MOV     (SP)+,R4
013614  012603                          MOV     (SP)+,R3
013616  012602                          MOV     (SP)+,R2
013620  000207                  9$:     RETURN
```

```
      1                                             .SBTTL  SBCALC -- Calculate starting block # for a file entry
      2                                     ;-------------------------------------------------------------------
      3                                     ;  SBCALC is called to calculate the starting block number of a file.
      4                                     ;
      5                                     ;  Inputs:
      6                                     ;   R1 = Address of directory entry for file.
      7                                     ;   USRBUF = Directory segment containing entry.
      8                                     ;
      9                                     ;  Outputs:
     10                                     ;   R0 = Starting block number of the file.
     11                                     ;
     12 013622  010246                      SBCALC: MOV     R2,-(SP)
     13 013624  013700  000000C                     MOV     USRBUF+DH$BLK,R0;GET BLOCK # OF 1ST FILE IN THIS DIR SEGMENT
     14 013630  012702  000000C                     MOV     #USRBUF+DH$$SZ,R2;POINT TO 1ST DIR ENTRY IN THIS SEGMENT
     15 013634  020201                      1$:     CMP     R2,R1           ;HAVE WE REACHED OUR ENTRY?
     16 013636  001405                              BEQ     2$              ;BR IF YES
     17 013640  066200  000000G                     ADD     FD$LEN(R2),R0   ;ADD SPACE ALLOCATED TO THAT FILE ENTRY
     18 013644  063702  002006'                     ADD     DIRSIZ,R2       ;POINT TO NEXT ENTRY
     19 013650  000771                              BR      1$
     20 013652  012602                      2$:     MOV     (SP)+,R2
     21 013654  000207                              RETURN
```

```
                                        .SBTTL  RDSEG1 -- Read 1st directory segment
                                ;-------------------------------------------------------------------
                                ;  RDSEG1 is called to read the first directory segment on a device
                                ;  and set up parameters about the directory.
                                ;
                                ;  Inputs:
                                ;   CHNNUM = Number of channel we are working on.
                                ;
                                ;  Outputs:
                                ;   R1 = Pointer to dummy directory entry in front of 1st real one (for NXTDIR).
                                ;   USRBUF = 1st directory segment.
                                ;   CURSEG = 0
                                ;   DIRHIS = Number of highest directory segment in use.
                                ;   DIRSIZ = Number of bytes per directory entry.
                                ;
013656                          RDSEG1:
                                ;
                                ;  Read in directory segment # 1
                                ;
013656  012700  000001                  MOV     #1,R0           ;SAY WE WANT SEG # 1
013662  004737  013772'                 CALL    RDSEG           ;READ THE SEGMENT IN
                                ;
                                ;  Set up parameters about the directory
                                ;
013666  012701  000002'                 MOV     #USRBUF,R1      ;POINT TO BUFFER WITH DIRECTORY SEGMENT
013672  016137  000000G 002004'         MOV     DH$HIS(R1),DIRHIS;HIGHEST SEGMENT # USED
013700  016137  000000G 002010'         MOV     DH$NSG(R1),DIRNSG;NUMBER OF DIRECTORY SEGMENTS AVAILABLE
013706  001425                  BEQ     5$              ;BR IF INVALID DIRECTORY
013710  023727  002010' 000037        CMP     DIRNSG,#31.     ;CHECK TO SEE IF VALID
013716  101021                  BHI     5$              ;BR IF INVALID
013720  016100  000000G                 MOV     DH$NEB(R1),R0   ;# EXTRA BYTES PER DIRECTORY ENTRY
013724  020027  001000                  CMP     R0,#512.        ;CHECK IF REASONABLE
013730  101014                  BHI     5$              ;BR IF INVALID
013732  062700  000000G                 ADD     #FD$OPT,R0      ;# STANDARD BYTES PER DIRECTORY ENTRY
013736  010037  002006'                 MOV     R0,DIRSIZ       ;TOTAL # BYTES PER DIRECTORY ENTRY
013742  026127  000000G 000000G        CMP     DH$BLK(R1),#DH$$BS;MAKE SURE BASE BLOCK # IS REASONABLE
013750  103404                  BLO     5$              ;BR IF INVALID
                                ;
                                ;  Point R1 in front of 1st directory entry so NXTDIR will get 1st real entry.
                                ;
013752  062701  000000G                 ADD     #DH$$SZ,R1      ;POINT TO 1ST REAL DIRECTORY ENTRY
013756  160001                  SUB     R0,R1           ;POINT BACK 1 ENTRY
                                ;
                                ;  Finished
                                ;
013760  000207                  RETURN
                                ;
                                ;  Invalid directory
                                ;
013762  012700  177755          5$:     MOV     #UERR6,R0       ;FATAL ERROR
013766  000137  015356'                 JMP     USRCLS
```

```
                                        .SBTTL  RDSEG  -- Read a directory segment
                                ;-------------------------------------------------------------------------
                                ;  RDSEG is called to read a directory segment.
                                ;
                                ;  Inputs:
                                ;   R0 = Number of directory segment to be read.
                                ;   CHNNUM = Number of current channel being used.
                                ;
                                ;  Outputs:
                                ;   R1 = Address of dummy directory in front of first real one.
                                ;   USRBUF = Segment read in.
                                ;   CURSEG = Number of segment read in.
                                ;
014002  020037  002002'         RDSEG:  CMP     R0,CURSEG       ;IS DESIRED SEGMENT ALREADY IN BUFFER?
014006  001437                          BEQ     1$              ;BR IF YES
014010  010001                          MOV     R0,R1           ;GET SEG #
                                ;  Convert segment # to absolute block #.
014012  010137  002002'         2$:     MOV     R1,CURSEG       ;SAVE CURRENT SEGMENT #
014016  005301                          DEC     R1              ;1ST DIRECTORY SEGMENT IS # 1
014020  006301                          ASL     R1              ;CVT SEGMENT # TO BLOCK #
014022  062701  000000G                 ADD     #DH$$BS,R1      ;BASE BLOCK # OF 1ST SEGMENT
                                ;  Read in the segment.
014026                                  .READW  #USREMT,CHNNUM,#USRBUF,#512.,R1
014070  103006                          BCC     1$              ;BR IF NO ERROR
014072  004737  015426'                 CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
014076  012700  177775                  MOV     #-3,R0          ;DIRECTORY READ ERROR
014102  000137  015356'                 JMP     USRCLS
                                ;  Set R1 to point to dummy directory entry in front of first real one.
014106  012701  000000C         1$:     MOV     #USRBUF+DH$$SZ,R1;POINT TO FIRST REAL ENTRY IN SEGMENT
014112  163701  002006'                 SUB     DIRSIZ,R1       ;POINT IN FRONT OF FIRST ONE
014116  000207                          RETURN
```

```
                                        .SBTTL  WRTSEG -- Write directory segment
                                ;-------------------------------------------------------------------------
                                ;  WRTSEG writes the current directory segment to disk.
                                ;
                                ;  Inputs:
                                ;   USRBUF = Current segment.
                                ;   CURSEG = Number of current segment in USRBUF.
                                ;
014110  010146          WRTSEG: MOV     R1,-(SP)
014112  013701  002002'          MOV     CURSEG,R1       ;GET CURRENT SEGMENT #
014116  005301                  DEC     R1              ;1ST SEGMENT IS # 1
014120  006301                  ASL     R1              ;CVT SEGMENT # TO BLOCK #
014122  062701  000000G          ADD     #DH$$BS,R1      ;BASE BLOCK # OF 1ST SEGMENT
014126                          .WRITW  #USREMT,CHNNUM,#USRBUF,#512.,R1
014170  103006                  BCC     1$              ;BR IF WRITE OK
014172  004737  015426'          CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
014176  012700  177775          MOV     #-3,R0          ;DIRECTORY WRITE ERROR
014202  000137  015356'          JMP     USRCLS          ;ABORT OPERATION
014206  012601          1$:     MOV     (SP)+,R1
014210  000207                  RETURN
```

```
                                        .SBTTL  SPLDIR -- Directory operations for special devices
                                ;-----------------------------------------------------------------------
                                ;  SPLDIR is jumped to from the normal USR directory routines to perform
                                ;  a directory operation on a special type device such as a magnetic
                                ;  tape or cassette.  SPLDIR releases the USR, performs the operations
                                ;  and then returns from the emt.
                                ;
                                ;  Inputs:
                                ;   R2 = Operation function code (DF$xxx).
                                ;   R3 = Address of CSW for channel.
                                ;   FILSPC = File spec.
                                ;
014212                          SPLDIR:
                                ;
                                ;  Free the USR module.
                                ;
014212  004737  010146'                 CALL    FREUSR          ;FREE THE USR DATA BASE FOR OTHERS TO USE
                                ;
                                ;  Obtain exclusive access to special device data base.
                                ;
014216  004737  014610'                 CALL    GETSPD          ;GAIN EXCLUSIVE ACCESS TO SPD DATA BASE
                                ;
                                ;  Copy the untranslated device/file specification from the user's area
                                ;  to a system buffer.
                                ;
014222  013701  000002G                 MOV     EMTBLK+2,R1     ;Point to file spec in user's area
014226  042701  000001                  BIC     #1,R1           ;Make sure address is even
014232  010246                          MOV     R2,-(SP)        ;Save operation function code
014234  012702  000000G                 MOV     #SPFLDV,R2      ;Point to buffer where name is to be stored
014240  004737  007720'                 CALL    CPYSPC          ;Copy file spec to SPFLDV
014244  012602                          MOV     (SP)+,R2        ;Restore operation function code
014246  005037  000000G                 CLR     SPSIZE          ;HANDLER RETURNS FILE SIZE HERE
014252  005037  000000G                 CLR     SPUSR           ;HANDLER RETURNS ERROR CODE HERE
014256  042763  000000C 000000G         BIC     #<CS$EOF!CS$ERR>,C.CSW(R3) ;IGNORE PRIOR ERROR
                                ;
                                ;  Build an I/O queue entry to do the operation.
                                ;
014264  004737  000000G                 CALL    GETQ            ;GET A FREE I/O QUEUE ENTRY
014270  010361  000000G                 MOV     R3,Q.UCSW(R1)   ;SET ADDRESS OF CHANNEL CSW
014274  013761  000000G 000000G         MOV     @#KPAR6,Q.PA6(R1);Save mapping for context block
014302  010105                          MOV     R1,R5           ;Get address of Q element
014304  062705  000000G                 ADD     #Q.ICSW,R5      ;Point to internal CSW cell
014310  010561  000000G                 MOV     R5,Q.CSW(R1)    ;Set up pointer to internal CSW cell
014314  010300                          MOV     R3,R0           ;Get pointer to original channel block
014316  012025                          MOV     (R0)+,(R5)+     ;Copy original channel block to internal one
014320  012025                          MOV     (R0)+,(R5)+
014322  012025                          MOV     (R0)+,(R5)+
014324  012025                          MOV     (R0)+,(R5)+
014326  011015                          MOV     (R0),(R5)
014330  016300  000000G                 MOV     C.CSW(R3),R0    ;Get the channel status word
014334  042700  177701                  BIC     #^C76,R0        ;Isolate the device index number
014340  110061  000000G                 MOVB    R0,Q.DEVX(R1)   ;store the device index number
014344  013700  000006G                 MOV     EMTBLK+6,R0     ;GET FILE SEQUENCE # FOR .ENTER
014350  020227  000000G                 CMP     R2,#DF$ENT      ;ARE WE DOING A .ENTER?
014354  001402                          BEQ     6$              ;BR IF YES
014356  013700  000004G                 MOV     EMTBLK+4,R0     ;GET SEQUENCE # FOR .LOOKUP, .DELETE, ETC.
014362  010061  000000G         6$:     MOV     R0,Q.BLKN(R1)   ;SET SEQUENCE # IN Q.BLKN FIELD
```

```
     58 014366  013761  000004G 000000G          MOV     EMTBLK+4,Q.WCNT(R1);SET FILE ENTER SIZE IN Q.WCNT FIELD
     59 014374  110261  000000G                  MOVB    R2,Q.FUNC(R1)    ;SET OPERATION FUN CODE (LOOKUP, ENTER, ETC.)
     60 014400  116361  000000G 000000G          MOVB    C.DEVQ(R3),Q.UNIT(R1);SET UNIT NUMBER
     61 014406  113700  000000G                  MOVB    CORUSR,R0        ;Get current job index number
     62 014412  006200                          ASR     R0               ;Convert to job number
     63 014414  110061  000000G                  MOVB    R0,Q.JOB(R1)     ;Set job number in queue element
     64 014420  072027  000003                   ASH     #3,R0            ;Position for Q.JNUM field
     65 014424  042700  177407                   BIC     #^C<370>,R0      ;Clear all but job number field
     66 014430  150061  000000G                  BISB    R0,Q.JNUM(R1)    ;Set job # in queue element
     67 014434  012704  000000G                  MOV     #SPFLNM,R4       ;GET ADDRESS OF FILE SPEC BUFFER
     68 014440  005005                          CLR     R5               ;SET FOR SHIFT
     69 014442  073427  177772                   ASHC    #-6,R4           ;SHIFT LOW-ORDER 6 BITS INTO R5
     70 014446  000241                          CLC
     71 014450  006005                          ROR     R5               ;RIGHT JUSTIFY LOW-ORDER 6 BITS IN R5
     72 014452  072527  177767                   ASH     #-9.,R5
     73 014456  062705  000000G                  ADD     #VPAR6,R5        ;BIAS ADDRESS TO BE IN PAR6 RANGE
     74 014462  010561  000000G                  MOV     R5,Q.BUFF(R1)    ;SET THIS AS VIRTUAL BUFFER ADDRESS
     75 014466  010461  000000G                  MOV     R4,Q.PAR(R1)     ;SET PAR6 BASE OFFSET
     76 014472  005061  000000G                  CLR     Q.COMP(R1)       ;NO COMPLETION ROUTINE
     77 014476  013761  000000G 000000G          MOV     CHNNUM,Q.CHAN(R1);SET USER'S CHANNEL NUMBER
     78                                ;
     79                                ;  Initiate the I/O operation.
     80                                ;
     81 014504  004737  000000G                       CALL    QIO              ;QUEUE THE I/O OPERATION
     82                                ;
     83                                ;  Wait for I/O to finish.
     84                                ;
     85 014510                                .WAIT   CHNNUM           ;WAIT FOR OPERATION TO FINISH
     86 014520  103003                        BCC     3$               ;BR IF NO I/O ERROR
     87                                ;  Directory I/O error.
     88 014522  012737  177775  000000G          MOV     #-3,SPUSR        ;SET I/O ERROR CODE
     89                                ;
     90                                ;  Return handler reported file size to user in R0.
     91                                ;
     92 014530  013737  000000G 000000G 3$:     MOV     SPSIZE,URO       ;RETURN HDLR REPORTED FILE SIZE IN USER'S R0
     93 014536  013705  000000G                 MOV     SPUSR,R5         ;GET HANDLER REPORTED ERROR CODE
     94                                ;
     95                                ;  Release special device data base area.
     96                                ;
     97 014542  004737  014652'                 CALL    FRESPD           ;FREE SPD AREA
     98                                ;
     99                                ;  See if we should purge the channel.
    100                                ;
    101 014546  020227  000000G                 CMP     R2,#DF$LOK       ;DOING A LOOKUP?
    102 014552  001405                          BEQ     4$               ;BR IF YES
    103 014554  020227  000000G                 CMP     R2,#DF$ENT       ;DOINT AN ENTER?
    104 014560  001402                          BEQ     4$               ;BR IF YES
    105 014562  005063  000000G                 CLR     C.CSW(R3)        ;PURGE THE CHANNEL
    106                               ;
    107                               ;  See if handler reported an error
    108                               ;
    109 014566  010500                4$:       MOV     R5,R0            ;DID HANDLER RETURN AN ERROR CODE?
    110 014570  001002                          BNE     5$               ;BR IF YES
    111 014572  000137  000000G                 JMP     EMTXIT           ;NORMAL EMT EXIT
    112 014576                        5$:
    113                               ;;        CLR     C.CSW(R3)        ;PURGE CHANNEL IF ERROR OCCURED
    114                               ;; This change (from CLR to BIC) is implemented to allow "some" information
```

```
                                        ;; to be returned by .CSTAT on failed lookups. RT-11 actually maintains
                                        ;; a valid C.CSW word on this type of error so their .CSTAT always works.
014576  042763  000000G 000000G        BIC     #CS$OPN,C.CSW(R3) ;SPECIAL PURGE FOR NON-RT DIR DEVICES
014604  000137  000000G                JMP     SETERR            ;RETURN ERROR CODE
```

```
                                        .SBTTL  GETSPD -- Gain exclusive access to special device data base
                                ;-------------------------------------------------------------------------
                                ;  GETSPD is called to gain exclusive access to the special device data base.
                                ;  If the data base is currently in use by another job, our job is suspended
                                ;  until the data base becomes free.
                                ;
014610  113700  000000G        GETSPD: MOVB    SPDJOB,R0       ;IS SPD DATA BASE FREE NOW?
014614  001412                        BEQ     1$              ;BR IF YES
014616  120037  000000G                CMPB    R0,CORUSR       ;HAVE WE ALREADY GOT IT?
014622  001412                        BEQ     2$              ;BR IF YES
                                ;
                                ;  Someone else owns SPD data base.
                                ;  Suspend our job until it becomes free.
                                ;
014624  012700  000000G                MOV     #S$QSPD,R0      ;QUEUE FOR SPD DATA BASE
014630  004737  000000G                CALL    QNSPNX          ;PUT US AT TAIL OF QUEUE FOR IT
014634  004737  000000G                CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
014640  000763                        BR      GETSPD          ;NOW TRY TO GET SPD
                                ;
                                ;  We can get SPD data base now.
                                ;
014642  113737  000000G 000000G 1$:    MOVB    CORUSR,SPDJOB   ;CLAIM SPD DATA BASE FOR US
014650  000207                2$:     RETURN
                                        .SBTTL  FRESPD -- Free special device data base
                                ;-------------------------------------------------------------------------
                                ;  Free the special device data base and restart any user who is waiting
                                ;  for it.
                                ;
014652  123737  000000G 000000G FRESPD: CMPB    CORUSR,SPDJOB   ;DO WE CURRENTLY OWN SPD?
014660  001006                        BNE     1$              ;BR IF NOT
014662  105037  000000G                CLRB    SPDJOB          ;FREE IT
                                ;
                                ;  Restart any job that is waiting for SPD data base.
                                ;
014666  012700  000000G                MOV     #S$QSPD,R0      ;GET SPD WAIT QUEUE STATE CODE
014672  004737  000000G                CALL    UREGO           ;RESTART ANY WAITING JOB
                                ;
                                ;  Finished
                                ;
014676  000207                1$:     RETURN
```

```
                        .SBTTL  CSHADD -- Add file entry to directory cache
                ;-----------------------------------------------------------------------
                ;  CSHADD is called to add a new file entry to the directory cache.
                ;
                ;  Inputs:
                ;   R1 = Pointer to directory entry for file in USRBUF.
                ;   FILDVU = Device # / unit #
                ;
014700  010246  CSHADD: MOV     R2,-(SP)
014702  010346         MOV     R3,-(SP)
014704  010446         MOV     R4,-(SP)
014706  010546         MOV     R5,-(SP)
                ;
                ;  See if we are caching file entries for this device
                ;
014710  004737  015210' CALL    CSHTST          ;ARE WE TO CACHE ENTRIES FOR THIS DEVICE?
014714  103440         BCS     9$              ;BR IF NOT
                ;
                ;  We are caching entries for this device.
                ;  Find least recently used entry in cache table (or 1st unused entry).
                ;
014716  012702  000000C        MOV     #CSHHD-FC$LNK,R2;DUMMY POINTER TO PHANTOM FIRST ENTRY
014722  010203  1$:     MOV     R2,R3           ;REMEMBER ADDRESS OF PREVIOUS ENTRY
014724  016202  000000G        MOV     FC$LNK(R2),R2   ;CHAIN FORWARD TO NEXT ENTRY IN LIST
014730  005762  000000G        TST     FC$CDX(R2)      ;IS THIS ENTRY UNUSED?
014734  001403                 BEQ     2$              ;BR IF YES
014736  005762  000000G        TST     FC$LNK(R2)      ;IS THIS THE LAST ENTRY IN LIST?
014742  001367                 BNE     1$              ;BR IF NOT
                ;
                ;  Relink cache entry at head of chain (it is most recently used).
                ;
014744  016263  000000G 000000G 2$:     MOV     FC$LNK(R2),FC$LNK(R3);RELINK CHAIN AROUND US
014752  013762  000000G 000000G        MOV     CSHHD,FC$LNK(R2);NOW LINK US AT FRONT OF LIST
014760  010237  000000G                MOV     R2,CSHHD
                ;
                ;  Copy info from file directory entry to cache entry.
                ;
014764  010203                 MOV     R2,R3           ;GET ADDRESS OF CACHE ENTRY
014766  010104                 MOV     R1,R4           ;GET ADDRESS OF DIRECTORY ENTRY
014770  012700  000000G        MOV     #FD$$SZ,R0      ;GET # BYTES TO MOVE
014774  006200                 ASR     R0              ;GET # WORDS TO MOVE
014776  012423         3$:     MOV     (R4)+,(R3)+     ;COPY DIRECTORY INFO TO CACHE ENTRY
015000  077002                 SOB     R0,3$
                ;
                ;  Store pointer to cached-device table entry into file cache entry
                ;
015002  010562  000000G        MOV     R5,FC$CDX(R2)   ;REMEMBER WHICH DEVICE FILE IS ASSOC WITH
                ;
                ;  Store starting block number of file
                ;
015006  004737  013622' CALL    SBCALC          ;CALCULATE STARTING BLOCK # OF FILE
015012  010062  000000G        MOV     R0,FC$SBL(R2)   ;SAVE STARTING BLOCK #
                ;
                ;  Finished
                ;
015016  012605         9$:     MOV     (SP)+,R5
015020  012604                 MOV     (SP)+,R4
```

```
     58 015022  012603                          MOV     (SP)+,R3
     59 015024  012602                          MOV     (SP)+,R2
     60 015026  000207                          RETURN
```

```
                                        .SBTTL  CSHDEL -- Remove a file entry from the directory cache
                                ;-------------------------------------------------------------------------------
                                ;  CSHDEL is called to remove an individual file entry from the cache table.
                                ;
                                ;  Inputs:
                                ;   R1 = Pointer to directory entry for file to be removed.
                                ;   FILDVU = Device # / unit # of device.
                                ;
015030  010146                 CSHDEL: MOV     R1,-(SP)
                                ;
                                ;  Search for file entry in cache table.
                                ;
015032  062701  000000G                 ADD     #FD$NAM,R1      ;POINT TO FILE NAME IN DIRECTORY ENTRY
015036  004737  015054'                 CALL    CSHCHK          ;SEE IF ENTRY IS IN DIRECTORY CACHE
015042  103402                         BCS     4$              ;BR IF FILE ENTRY IS NOT IN CACHE
                                ;
                                ;  Found entry in cache table.
                                ;  Mark it as deleted.
                                ;
015044  005061  000000G                 CLR     FC$CDX(R1)      ;MARK ENTRY AS DELETED
                                ;
                                ;  Finished
                                ;
015050  012601                 4$:     MOV     (SP)+,R1
015052  000207                         RETURN
```

```
                                        .SBTTL  CSHCHK -- Search for file entry in directory cache
                                ;-------------------------------------------------------------------
                                ;  CSHCHK is called to try to locate an entry for a file in
                                ;  the directory cache.  If it is found the file is set as the most
                                ;  recently used.
                                ;
                                ;  Inputs:
                                ;   R1 = Pointer to file name (3 words, rad50: name, name, extension)
                                ;   FILDVU = Device & unit #.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared if file entry is found.
                                ;   R1 = Pointer to cache table entry for file
                                ;   (same format as directory entry).
                                ;   R0 = Starting block # of file.
                                ;
015054  010246                  CSHCHK: MOV     R2,-(SP)
015056  010346                          MOV     R3,-(SP)
015060  010546                          MOV     R5,-(SP)
015062  010103                          MOV     R1,R3           ;CARRY FILE NAME POINTER IN R3
                                ;
                                ;  See if this device is being cached and get pointer to device table entry
                                ;
015064  004737  015210'                 CALL    CSHTST          ;SEE IF THIS DEVICE IS BEING CACHED
015070  103442                          BCS     4$              ;BR IF DEVICE IS NOT BEING CACHED
                                ;
                                ;  Search for file entry in cache table.
                                ;
015072  012701  000000C                 MOV     #CSHHD-FC$LNK,R1;GET POINTER TO DUMMY STARTING POINT
015076  010102                  1$:     MOV     R1,R2           ;REMEMBER ADDRESS OF PREVIOUS ENTRY
015100  016101  000000G                 MOV     FC$LNK(R1),R1   ;GET ADDRESS OF NEXT ENTRY IN LIST
015104  020561  000000G                 CMP     R5,FC$CDX(R1)   ;IS FILE ON CACHED DEVICE?
015110  001012                          BNE     2$              ;BR IF NOT
015112  010300                          MOV     R3,R0           ;POINT TO FILE NAME
015114  022061  000000G                 CMP     (R0)+,FD$NAM(R1);COMPARE FILE NAMES
015120  001006                          BNE     2$
015122  022061  000002G                 CMP     (R0)+,FD$NAM+2(R1)
015126  001003                          BNE     2$
015130  021061  000004G                 CMP     (R0),FD$NAM+4(R1)
015134  001404                          BEQ     3$              ;BR IF FOUND ENTRY FOR FILE
015136  005761  000000G          2$:     TST     FC$LNK(R1)      ;IS THIS LAST ENTRY IN LIST?
015142  001355                          BNE     1$              ;BR IF NOT
015144  000414                          BR      4$              ;FILE ENTRY IS NOT IN CACHE TABLE
                                ;
                                ;  Found entry -- Relink at head of list (it is most recently used).
                                ;
015146  016162  000000G 000000G 3$:     MOV     FC$LNK(R1),FC$LNK(R2);RELINK LIST AROUND US
015154  013761  000000G 000000G         MOV     CSHHD,FC$LNK(R1);LINK US IN AT HEAD OF LIST
015162  010137  000000G                 MOV     R1,CSHHD
                                ;
                                ;  Return with R1 = address of cache entry, R0 = Starting block #.
                                ;
015166  016100  000000G                 MOV     FC$SBL(R1),R0   ;GET STARTING BLOCK # FOR FILE
015172  000241                          CLC                     ;SIGNAL SUCCESSFUL RETURN
015174  000401                          BR      5$
                                ;
                                ;  Can't find file entry in cache.
```

```
                                        ;
015176  000261                  4$:     SEC                     ;SIGNAL UNSUCCESSFUL RETURN
                                        ;
                                        ;  Return
                                        ;
015200  012605                  5$:     MOV     (SP)+,R5
015202  012603                          MOV     (SP)+,R3
015204  012602                          MOV     (SP)+,R2
015206  000207                          RETURN
```

```
                                        .SBTTL  CSHTST -- Determine if device is to be cached
                                ;-------------------------------------------------------------------------
                                ;  CSHTST is called to determine if file entries for a particular
                                ;  device & unit are to be stored in the directory cache.
                                ;
                                ;  Inputs:
                                ;   FILDVU = Device & unit #
                                ;
                                ;  Outputs:
                                ;   C-flag cleared if device is to be cached.
                                ;   R5 = Address of device entry in cached device table.
                                ;
015210  010346                  CSHTST: MOV     R3,-(SP)
015212  010446                          MOV     R4,-(SP)
                                ;
                                ;  If this file is on a logical disk, set up information about the LD.
                                ;
015214  004737  015266'                 CALL    GETDVU          ;GET INFO ABOUT PHYSICAL DEVICE AND UNIT
                                ;
                                ;  Search for device information in table of cached devices
                                ;
015220  013705  000000G                 MOV     CSHDEV,R5       ;POINT TO TABLE OF MOUNTED DEVICES
015224  020365  000000G          2$:    CMP     R3,CD$DVU(R5)   ;DO DEVICE # AND UNIT #'S MATCH?
015230  001003                          BNE     5$              ;BR IF NOT
015232  020465  000000G                 CMP     R4,CD$BAS(R5)   ;DOES BASE BLOCK # OF DISK MATCH?
015236  001407                          BEQ     3$              ;BR IF YES -- THIS DEVICE IS CACHED
015240  062705  000000G          5$:    ADD     #CD$$SZ,R5      ;POINT TO NEXT TABLE ENTRY
015244  020537  000000G                 CMP     R5,CSHDVN       ;REACHED END OF TABLE?
015250  103765                          BLO     2$              ;BR IF NOT
015252  000261                          SEC                     ;SIGNAL THAT DEVICE IS NOT TO BE CACHED
015254  000401                          BR      4$
015256  000241                  3$:     CLC                     ;SIGNAL THAT DEVICE IS TO BE CACHED
                                ;
                                ;  Finished
                                ;
015260  012604                  4$:     MOV     (SP)+,R4
015262  012603                          MOV     (SP)+,R3
015264  000207                          RETURN
```

```
                                        .SBTTL  GETDVU -- Get device # & unit # info
                                ;-------------------------------------------------------------------
                                ;  GETDVU is called to get the device number, unit number, and starting
                                ;  block number of a logical disk.
                                ;
                                ;  Inputs:
                                ;   FILDVU = Logical device number and unit number.
                                ;
                                ;  Outputs:
                                ;   R3 = Physical device number and unit number.
                                ;   R4 = Starting block number on physical disk of logical disk base.
                                ;        (Zero if this is not a logical disk)
                                ;
015266  013703  002014'         GETDVU: MOV     FILDVU,R3       ;GET DEVICE # AND UNIT #
015272  005004                         CLR     R4              ;ASSUME DEVICE IS NOT A LOGICAL DISK
015274  120337  000000G                 CMPB    R3,LDDEVX       ;IS THIS DEVICE A LOGICAL DISK?
015300  001010                         BNE     4$              ;BR IF NOT
015302  000303                         SWAB    R3              ;GET LOGICAL UNIT # TO LOW ORDER BYTE
015304  042703  177770                 BIC     #^C7,R3         ;MASK OUT ALL BUT UNIT #
015310  006303                         ASL     R3              ;CONVERT TO WORD TABLE INDEX
015312  016304  000000G                 MOV     LDBASE(R3),R4   ;GET STARTING POSITION OF LOGICAL DISK
015316  016303  000000G                 MOV     LDPDEV(R3),R3   ;GET PHYSICAL DEVICE # AND UNIT #
                                ;
                                ;  Finished
                                ;
015322  000207                 4$:     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:
                                ;   R3 = Job index number of job whose flag is to be identified.
                                ;   R5 = Pointer to cached device table entry.
                                ;
                                ;  Outputs:
                                ;   R2 = Address of byte that contains mount-flag.
                                ;   R3 = Mount-flag bit positioned correctly within byte.
                                ;
015324  006203                  CDJFLG: ASR     R3              ;Convert job index to index by 1
015326  005303                          DEC     R3              ;Make base job number 0
015330  005002                          CLR     R2              ;Clear for divide
015332  071227  000010                  DIV     #8.,R2          ;Divide by 8 jobs per byte
015336  060502                          ADD     R5,R2           ;Get address of byte within
015340  062702  000000G                 ADD     #CD$JOB,R2      ; cached-device table entry
015344  012700  000001                  MOV     #1,R0           ;Get a mount flag
015350  072003                          ASH     R3,R0           ;Position flag according to job number
015352  010003                          MOV     R0,R3           ;Return flag in R3
015354  000207                          RETURN
```

```
                                        .SBTTL  USRXIT -- Exit from USR
                                ;-------------------------------------------------------------------------
                                ;  USRXIT is the exit point for EMT processing within the USR.
                                ;  USRCLS sets an I/O error code, purges the channel then aborts the emt.
                                ;  USRERR sets an I/O error code but does not purge the channel.
                                ;
                                ;  Inputs:
                                ;   R0 = error code [USRERR, and USRCLS only]
                                ;
015356  013703  000000G         USRCLS: MOV     CHNADR,R3       ;GET ADDRESS OF CURRENT CHANNEL
015362  001402                          BEQ     USRERR          ;BR IF NONE
015364  005063  000000G                 CLR     C.CSW(R3)       ;PURGE THE CHANNEL
015370  010046                  USRERR: MOV     R0,-(SP)        ;SAVE ERROR CODE
015372  004737  010146'                 CALL    FREUSR          ;FREE USR DATA BASE IF WE HAVE IT
015376  012600                          MOV     (SP)+,R0        ;GET BACK ERROR CODE
015400  000137  000000G                 JMP     SETERR          ;RETURN ERROR CODE FOR EMT
                                ;
                                ;  Normal exit from USR
                                ;
015404  004737  010146'         USRXIT: CALL    FREUSR          ;FREE USR MODULE
015410  000137  000000G                 JMP     EMTXIT          ;EXIT FROM EMT
                                ;
                                ;  Fatal abort due to internal consistency check.
                                ;
                                ;  Inputs:
                                ;   R0 = Abort code.
                                ;
015414  010005                  UABORT: MOV     R0,R5           ;GET ERROR CODE FOR ABORT
015416  013704  000000G                 MOV     EMTADR,R4       ;GET ADDRESS OF ABORTED EMT
015422  000137  000000G                 JMP     ABORT           ;GIVE A FATAL ABORT
```

```
                                        .SBTTL  ERRNAM -- Set name of file spec for error message
                                ;-------------------------------------------------------------------------
                                ;  ERRNAM is called to move the current device/file specification to a
                                ;  cell that will cause the spec to be printed with an error message
                                ;  when the job reenters TSKMON.
                                ;
                                ;  Inputs:
                                ;   FILSPC = Device/file specification in RAD50 form.
                                ;
                                ;  Outputs:
                                ;   ERRSPC = Device/file specification saved for TSKMON.
                                ;
015426                          ERRNAM:
                                ;
                                ;  Don't set file spec for error message if a .SERR has been done
                                ;
015426  105737  000000G                TSTB    SERFLG                  ;Has a .SERR been done?
015432  001014                         BNE     9$                      ;Br if yes
                                ;
                                ;  Move file spec to holding area
                                ;
015434  013737  000000G 000000G        MOV     FILSPC,ERRSPC           ;Device name
015442  013737  000002G 000002G        MOV     FILSPC+2,ERRSPC+2       ;File name part 1
015450  013737  000004G 000004G        MOV     FILSPC+4,ERRSPC+4       ;File name part 2
015456  013737  000006G 000006G        MOV     FILSPC+6,ERRSPC+6       ;Extension
                                ;
                                ;  Finished
                                ;
015464  000207                 9$:     RETURN
                                ;
                                ;  Top of TSUSR module
                                ;
015466                          USRTOP:
        000001                          .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  10520 Words  ( 42 Pages)
Size of core pool:  18176 Words  ( 71 Pages)
Operating  system:  RT-11

Elapsed time:  00:01:40.08
,LP:TSUSR=DK:TSUSR/C/N:SYM
```

Cross reference table (CREF V05.05)

```
$DILUP    1-55      41-40
$DUPRN    1-71      34-131
$KINIT    1-41      19-77
$NOIN     1-62      34-49
$UCLRN    1-47      34-41
...V1    48-23      48-23     49-14     49-14     50-85     50-85
...V2    48-23      48-23     48-23#    48-23#    49-14     49-14     49-14#    49-14#    50-85     50-85#
ABORT     1-63      58-30
ACCMCH   34-94      34-113    35-15#
AD$$SZ    1-51      14-41     14-60     15-21     15-34     17-41     33-24
AD$DVU    1-51      14-39     14-68*    15-20*    15-32     15-45*    17-39     33-22
AD$FLG    1-51      14-70*
AD$JOB    1-51      14-48     14-51*    14-58     14-69*    15-17     15-19*    15-38     15-46*    17-61     33-41
ADDENT    5-55      38-16#
AF$BYA    1-39      29-26
ALCCOM   14-27      15-9      16-6      18-11#
ALCEMT    1-25      14-6#
ALCEND    1-51      14-42     14-61     15-22     15-35     17-42     33-25
ALCTBL    1-51      14-37     14-57     15-10     17-38     33-20
ALCTST   14-31      16-10     17-16#
ASNEND    1-33      29-32
ASNSIZ    2-48#      5-42     29-15*    29-50*
ASNTBL    1-32      29-28
AT$$SZ    1-34      29-31
AT$DEV    1-34      29-39
AT$EXT    1-34      29-44
AT$FIL    1-34      29-40     29-43
AT$LOG    1-34      29-29
AT$SIZ    1-34      29-47
BADEMT    1-56      14-16
C.CSW     1-40       5-10      5-30*     5-67*     6-16      6-51      6-68      6-149*    7-9       7-61*     8-6       8-16
          8-23       8-40      8-51      8-58*     8-63      8-145*    11-35*    12-36*    13-45     19-87     19-94     28-35
         28-61*     28-62*    34-32*    34-74*    34-147*   41-44     41-46     50-34*    50-50     50-105*   50-117*   58-12*
C.DEVQ    1-40       6-63      8-50      19-96     27-18     28-60*    50-60
C.LENG    1-40       4-65*     5-68*      8-112    28-64*
C.NUMQ    1-40      28-66*
C.SBLK    1-40       4-63*     5-82*     8-57*     28-63*
C.USED    1-40       5-83*     8-114     28-65*
C1DEVX    1-62      18-33     32-79
CD$$SZ    1-72      19-62     20-35     22-27     26-25     55-27
CD$$UB    1-73      20-66     24-21
CD$BAS    1-72      20-49*    24-29     55-25
CD$DVU    1-72      19-32     19-34     20-33     20-48*    22-20     24-31     24-43     24-48*    55-23
CD$JOB    1-73      20-65     24-20     57-19
CD$NAM    1-73      20-45*    20-46*    20-58*    20-59*
CD$TOP    1-72      20-47*    20-54*
CDJFLG   19-45      20-73     22-15     24-14     26-19     26-23     57-14#
CHKABT    1-36      31-25     51-17
CHKACC    6-50      28-74     34-16#
CHKALC   28-70      33-10#
CHKDEV    6-42       9-10     10-16     18-26     25-24     28-48     32-14#
CHKSD     1-42       4-14      5-20
CHKUSE   17-48      19-17#
CHNADR    1-32       5-63      8-5       28-27     58-10
CHNNUM    1-39       8-67     38-49     48-23     48-23     49-14     49-14     50-77     50-85     50-85
CHNSIZ    1-60
```

Cross reference table (CREF V05.05)

```
CKADEV    2-52#    34-84*   34-133   35-25
CKAUNT    2-53#    34-86*   34-135   35-29
CL$LIX    1-46     32-83
CLCLOS    1-69
CLDEVX    1-62     18-28    32-77
CLOSE     1-24      8-5#
CLRDIR    1-25     27-12#
CLTOTL    1-56     18-30    18-35
CLZERR    1-73      8-149
CNSLXD    2-54#    37-108   41-20*   41-50*   42-18*   42-54*
CONSOL    6-117     6-144    7-56     8-100    8-136    37-31    41-12#
CORUSR    1-33      8-73    20-72    22-13    24-13    26-22    31-14    31-26    31-33    31-58    33-42    34-22
         34-130    38-48    50-61    51-9     51-22    51-30
CPLEMT    1-49     45-42
CPYMNT    1-27     26-9#
CPYSPC   29-20     30-10#   50-30
CS$ENT    1-41      5-30     5-66     8-40    41-46
CS$EOF    1-45      8-58    50-34
CS$ERR    1-45      8-58    50-34
CS$NMX    1-41      8-24     8-52
CS$OPN    1-42      8-6     19-87    28-35    28-62    41-44    50-117
CS$RON    1-45      5-10     6-16     6-51     6-68     7-9     13-45    34-32    34-74    34-147
CS$SPL    1-42      8-16
CSHADD    4-56      6-140    8-134    11-27    52-9#
CSHALC    1-42     24-41
CSHCHK    4-42     53-14    54-17#
CSHCLN    1-42     24-44
CSHDEL    6-34      6-107    7-48     8-92    13-32    53-9#
CSHDEV    1-64     19-31    20-32    22-19    26-17    55-22
CSHDMT   21-12     21-49#
CSHDVN    1-64     19-63    20-36    22-28    26-26    55-28
CSHHD     1-63      3-15*   21-50    24-52    27-29    52-22    52-33    52-34*   54-29    54-48    54-49*
CSHTST   20-13     23-12    27-24    52-16    54-24    55-13#
CURCP     1-61     31-39
CURSEG    2-31#     5-64     6-29     6-82     6-112    8-85     8-96    31-46*   37-77    37-88    37-94    37-104
         37-106    37-161   40-27    40-80*   48-14    48-18*   49-10
DELETE    1-24      7-5#
DEVSIZ    1-59      9-30
DF$CLS    1-68      8-29
DF$DEL    1-68      7-21
DF$ENT    1-68      5-31    50-54    50-103
DF$LOK    1-68      4-24    50-101
DH$$BS    1-53     47-36    48-21    49-13
DH$$SZ    1-53     38-72    40-34    40-44    40-63    40-84    41-24    42-5     42-40    46-14    47-41    48-29
DH$BLK    1-52     40-43    40-81*   46-13    47-36
DH$HIS    1-52     40-96*   47-26
DH$NEB    1-52     47-31
DH$NSG    1-52     47-27
DH$NXT    1-52     37-114   40-70    40-73*   40-79*   43-27
DIDDLE    4-72      5-74     6-136   45-18#
DIRHIS    2-32#    40-25    40-71    40-97*   47-26*
DIRNSG    2-34#    40-25    47-27*   47-29
DIRSIZ    2-33#     8-119    8-124   37-35    38-27    38-61    38-70    38-71    38-76    38-78    39-27    39-34
         40-39     40-46    40-52    41-25    44-14    46-18    47-35*   48-30
DISMNT    1-27     21-5#
DLCEMT   14-13     15-5#
```

Cross reference table (CREF V05.05)

```
DMTALL  21-14   22-5#
DMTDEV  22-26   23-17   24-8#
DMTSUB  20-9    21-28   23-8#
DS$DIR   1-47    4-29    5-36    6-9     7-26   13-18   20-21   34-68
DS$NRD   1-46    4-20    5-26    7-17    8-25   20-23
DSTAT    1-25    9-5#
DVFLAG   1-38    4-22    5-28    7-19    8-27   20-25   20-27   33-32
DVSTAT   1-32    4-20    4-29    5-26    5-36    6-9     7-17    7-26    8-25    9-24    13-18   20-20
        34-68
DX$NMT   1-38   20-27
DX$NRD   1-46    4-22    5-28    7-19    8-27   20-25
DX$RAL   1-38   33-32
EMTADR   1-63   58-29
EMTBLK   1-32    5-44    6-38    9-21   10-27   11-7    11-43   11-51   12-11   14-10   18-19   21-10
        25-19   28-42   45-18   45-53   50-26   50-53   50-56   50-58
EMTCAD   1-49   45-37   45-39*
EMTPS    1-59    1-71   30-22   45-52
EMTXIT   1-54    8-155  50-111  58-21
ENTER    1-24    5-5#
ERRNAM   5-12    6-44    6-70    7-11   13-47   25-26   28-50   33-34   33-45   34-140  48-25   49-16
        59-13#
ERRSPC   1-45   59-22*  59-23*  59-24*  59-25*
EXCJOB   1-31   31-21   31-22*  31-28*
FC$$SZ   1-56    3-18    3-21    3-27
FC$CDX   1-37    3-24*  21-51*  24-53   24-55*  27-30   27-32*  52-25   52-47*  53-20*  54-32
FC$LNK   1-63    3-23*   3-28*  21-53   24-57   27-34   52-22   52-24   52-27   52-32   52-32*  52-33*
        54-29   54-31   54-41   54-47   54-47*  54-48*
FC$SBL   1-64   52-52*  54-53
FD$$SZ   1-64   45-30   45-62   52-40
FD$CHN   1-48    8-71   38-49*  41-42
FD$DAT   1-49    4-68    8-126   8-128*  11-10   11-11*  12-23   38-50*  45-51
FD$JOB   1-48    8-73   38-48*  41-34
FD$LEN   1-48    4-64    8-111   8-115*  8-122*  12-14   37-63   38-43   38-44*  38-63*  40-45   40-51
        42-17   42-17*  42-36   46-17
FD$NAM   1-48    6-128   8-78   21-52*  24-56*  27-33*  36-29   37-50   38-52   53-13   54-35   54-37
        54-39
FD$OPT   1-49   47-34
FD$STA   1-48    6-80*   6-81*   6-93    6-100*  6-101*  6-109*  6-125*  6-126*  7-41    7-52*   8-94*
        8-108   8-110*  8-120   11-57*  11-59*  12-18   37-36   38-47*  38-74   39-19   39-39*  40-36
        40-49   41-49*  42-14   42-42   44-15   44-17
FD$TIM   1-37    4-67    8-125*  8-132*  11-43*  12-26
FETCH    1-27   10-7#
FILBLK   2-35#
FILDVU   2-36#   8-50*   8-53*  14-38   14-68   15-29   15-31   17-27   17-29   17-31   17-39   18-48*
        18-49*  19-34   19-98   19-100  20-19   20-51   25-32*  25-33*  27-13   27-17*  27-20*  27-39*
        28-55*  28-56*  33-15   33-21   33-31   56-14
FILSPC   1-32    4-35    4-41    6-129   8-79   15-15   17-25   18-24   29-19   29-24   29-38   32-19
        34-55   34-66   35-36   36-30   37-51   38-53   59-22   59-23   59-24   59-25
FNDFIL   4-48    6-22    6-87    7-36    8-89   13-25   36-15#
FNDFRE   5-45   37-17#
FRECXT   1-57   19-125  19-132  42-61
FRESPD   1-27   50-97   51-30#
FREUSR   1-25    8-141  31-57#  50-17   58-14   58-20
FS$EMP   1-50    6-81    6-100   6-109   6-125    7-52    8-94    8-120   37-37   37-39   39-39   41-49
        42-7    42-14
FS$EOS   1-50   37-37   37-41   39-18   40-36   40-69   42-21   42-47   44-17
```

Cross reference table (CREF V05.05)

```
FS$PRM    1-50     6-80     6-101    6-126    8-110    36-24    37-37    40-49    42-42
FS$PRO    1-61     6-93     7-41    11-57    11-59    12-18    37-47
FS$TEN    1-50     8-68     8-108    38-47    38-74    40-49    41-28
GETCHA    1-55    19-83    41-43
GETCXT    1-57    19-24    41-19
GETDIR    8-69    36-25    43-17#
GETDVU   20-44    55-18    56-14#
GETQ      1-70    50-38
GETSPC    6-41     9-8     10-12    18-20    25-20    28-44    29-13#
GETSPD   50-21    51-7#    51-18
GETUSR    1-23     8-46     9-5      10-7     18-15    21-49    22-9     25-15    28-31    31-7#
GFINFO    1-26    12-10#
HANENT    1-59     9-28
HANSIZ    1-59     9-26
INSERT   38-62    38-77    39-14#
INTPRI    1-69    31-20    31-34    31-66    31-73
IOWAIT    1-57     8-12
JCDB      1-38     8-34
KMONCE    1-31    34-60
KPAR6     1-33    50-40
L1OFF     2-43#   37-76*   37-85    37-87*   37-131   37-147
L1SEG     2-44#   37-77*   37-84    37-88*   37-104   37-130   37-146
L1SIZ     2-42#   37-22*   37-71    37-75*   37-81    37-83    37-86*   37-122   37-129   37-135
L2OFF     2-46#   37-85*   37-93*   37-143
L2SEG     2-47#   37-84*   37-94*   37-106   37-142
L2SIZ     2-45#   37-23*   37-83*   37-90    37-92*   37-140   37-144
LD$RON    1-53    34-30
LDAEMT    1-33    21-44
LDBASE    1-72    56-21
LDDEMT    1-32    21-36
LDDEVX    1-62     9-35    18-38    32-65    34-26    56-16
LDFLAG    1-53    34-30
LDIEMT    1-32    21-40
LDNAME    1-73    20-56
LDPDEV    1-69    32-69    56-22
LDSIZE    1-55     9-38    20-53
LNPRIM    1-56    14-49    15-39    15-39    17-64    17-64    19-54    19-54    19-105   19-105   33-43    33-43
LOOKUP    1-24     4-5#
LSTFDD    1-36     4-68*
LSTFDT    1-36     4-67*
LSTSL     1-60    19-40    19-119   41-38
LSW       1-55    19-77    41-40
LSW2      1-43
LSW3      1-62    34-49
LSW5      1-45
LSW6      1-43    34-131
LSW7      1-47    34-41
MAXLD     1-33
MNTCOM   20-5     21-24    25-10#
MOUNT     1-27    20-5#
NLCHN     1-41    19-113
NSPLDV    1-73     4-12     5-18
NUMDEV    1-36    32-49
NXTDIR   38-24    39-21    41-29    42-8     42-22    42-48    43-21    44-14#   44-18
ODTBAS    1-71    30-24
OF$$SZ    1-44    34-101   34-118
```

Cross reference table (CREF V05.05)

```
OF$DEV    1-43     35-23
OF$FIL    1-44     35-18     35-35
OF$FLG    1-44     34-96     34-115
OF$UNT    1-44     35-27
OKFAND    1-43     34-92     34-102
OKFEND    1-45     34-111
OKFILE    1-43     34-91
OKFNND    1-43     34-110
OT$RON    1-44     34-96     34-115
OVRHC     1-69      4-14      5-20      8-18      19-24     19-83     19-125    19-132    41-19     41-43     42-61
PO$ALC    1-35     14-20     15-5
PO$BYP    1-35     34-36
PO$NFR    1-35     34-72
PO$NFW    1-35     34-70
PO$SYS    1-46     34-47
PNAME     1-39     32-50
PR7       1-69     31-11     31-57
PRIVCO    1-35      1-46     14-20     15-5      34-36     34-47     34-70     34-72
PSW       1-69     31-11*    31-20*    31-34*    31-57*    31-66*    31-73*
Q.BLKN    1-66     50-57*
Q.BUFF    1-66     50-74*
Q.CHAN    1-67     50-77*
Q.COMP    1-66     50-76*
Q.CSW     1-65     50-43*
Q.DEVX    1-65     50-52*
Q.FUNC    1-65     50-59*
Q.ICSW    1-67     50-42
Q.JNUM    1-64     50-66*
Q.JOB     1-65     50-63*
Q.PA6     1-68     50-40*
Q.PAR     1-66     50-75*
Q.UCSW    1-67     50-39*
Q.UNIT    1-65     50-60*
Q.WCNT    1-66     50-58*
QIO       1-70     50-81
QNSPNX    1-38     31-24     51-16
R$CHN     1-61
R$XCHN    1-61
R50DK     2-38#    32-37
R50SY     2-37#    32-39
R50SYS    2-39#    34-56
R50TSX    2-40#    34-58
RDSEG     6-98      6-123     8-66      8-103     37-116    37-164    40-107    43-29     47-21     48-14#
RDSEG1    8-59     36-20     37-27     40-95     47-16#
RENAME    1-24      6-5#
RESDEV    1-43     34-78
RUNFLG    1-39     29-26
S$QSPD    1-67     51-15     51-36
S$QUSR    1-33     31-23     31-67
SBCALC    5-69     12-30     36-38     46-12#    52-51
SDCLOS    1-47      8-18
SERFLG    1-41     59-17
SETERR    1-54      8-151    14-23     15-8      50-118    58-16
SFCLS     1-47      8-36
SFCOM    11-5      11-41     11-49     12-10     13-13#
SFDATE    1-26     11-5#
```

Cross reference table (CREF V05.05)

```
SFEXIT  11-27#    11-44     11-58     11-60
SFPROT    1-26    11-49#
SFTIME    1-26    11-41#
SFWRIT  11-6      11-42     11-50     13-45#
SPDJOB    1-70    51-7      51-22*    51-30     51-32*
SPFLDV    1-70    50-29
SPFLNM    1-70    50-67
SPLDIR    4-25     5-32      7-22      8-30     50-13#
SPLIT   38-37     40-17#
SPSIZE    1-70    50-32*    50-92
SPUSR     1-67    50-33*    50-88*    50-93
SYINDX    1-39    17-29     24-32     32-41     34-51     34-133
SYSCHN    1-54     1-60
SYSDAT    1-55     8-128    11-9
SYTIMH    1-37     8-129
SYTIML    1-37     8-130
SYUNIT    1-39    17-31     24-35     32-44     34-53     34-135
TK3SVL    1-37     8-131
TLCEMT  14-15     16-5#
TSUSR     1-15#    1-23
UABORT    8-173   58-28#
UACHKW    1-58    30-15
UERR1     2-59#    8-160
UERR2     2-60#    8-163
UERR3     2-61#    8-166
UERR4     2-62#    8-169
UERR5     2-63#    8-172
UERR6     2-64#   47-50
UMODE     1-71    30-22
URO       1-54     4-6*      4-66*     5-6*      5-54*     9-7       9-22*     10-11     10-30*    15-42*    16-5*     17-21*
         17-51*   17-55*    17-62*    17-63*    50-92*
UREGO     1-36    31-68     51-37
USRBUF    2-30#   37-114    38-31     38-72     40-34     40-43     40-44     40-63     40-70     40-73*    40-79*    40-81*
         40-84    40-85     40-96*    41-24     42-5      42-40     43-27     46-13     46-14     47-25     48-23     48-29
         49-14
USRCLS    4-51     5-14      5-49      5-59      6-12      6-25      6-46      6-62      6-72      6-104     7-13      7-32
          7-44     7-67     11-56     13-21     13-28     13-49     30-18     33-36     33-47     34-142    47-51     48-27
         49-18    58-10#
USRCNT    2-41#   31-35*    31-60*
USRCOM    4-5      5-5       6-5       7-5      13-13     28-26#
USREMT    1-63    48-23     49-14
USRERR    9-14    10-22     14-64     15-44     17-34     17-53     17-67     18-44     20-39     25-28     28-38     28-52
         31-42    58-11     58-13#
USRINI    1-23     3-6#
USRJOB    1-70    31-12     31-33*    31-58     31-62*
USRTOP    1-23     3-11     59-33#
USRUCA    1-25     2-26#    45-29     45-51     45-61
USRXIT    4-76     5-87      6-150     7-62      9-43      10-31     11-36     12-37     14-74     15-50     16-14     20-75
         21-32    21-55     22-41     58-20#
VALADB    1-58
VALADW    1-58     9-23     12-12
VMXFIL    1-47    37-152    37-154    37-156
VNFCSH    1-50     3-16
VPAR6     1-66    50-73
WLDNAM    1-33    35-40
WRTSEG    5-78     6-84      6-102     6-118     6-145     7-57      8-101     8-137     11-31     37-110    40-74     40-91
         40-98    49-9#
```

Cross reference table (CREF V05.05)

...CM1	48-23	49-14									
...CM2	48-23	48-23	48-23	48-23	49-14	49-14	49-14	49-14			
...CM5	48-23	49-14									
...CM7	48-23	49-14									
.READW	1-19#	48-23									
.WAIT	1-19#	50-85									
.WRITW	1-19#	49-14									
DISABL	2-4#	31-11	31-57								
ENABL	2-8#	31-20	31-34	31-66	31-73						
OCALL	2-15#	4-14	5-20	8-18	19-24	19-83	19-125	19-132	41-19	41-43	42-61