TSUSR -- File operation EMT's MACRO V05.04 Friday 18-Dec-87 15:35
Table of contents

.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).
                                        ;
002300  010063  000000G         2$:     MOV     R0,C.SBLK(R3)   ;SET FILE STARTING BLOCK #
002304  016100  000000G                 MOV     FD$LEN(R1),R0   ;GET ALLOCATED SIZE OF FILE
002310  010063  000000G                 MOV     R0,C.LENG(R3)   ;STORE IN CHANNEL BLOCK
002314  010037  000000G                 MOV     R0,UR0          ;RETURN TO USER IN R0
002320  016137  000000G 000000G         MOV     FD$TIM(R1),LSTFDT;SAVE TIME ENTRY FROM EACH LOOKUP
002326  016137  000000G 000000G         MOV     FD$DAT(R1),LSTFDD;ALSO SAVE DATE ENTRY
                                        ;
                                        ;  Allow user to diddle with directory entry if he wants to.
                                        ;
002334  004737  013254'                 CALL    DIDDLE          ;ALLOW DIRECTORY ENTRY DIDDLE
                                        ;
                                        ;  Finished
                                        ;
002340  000137  015172'         3$:     JMP     USRXIT          ;EXIT FROM USR PROCESSING
```

```
                                        .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.
                                ;
                                        .CSECT  TSUSR
000000                          TSUSR:  .RAD50  /USR/
000000  103112
                                ;
                                ;  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
                                        .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$$SZ
                                        .GLOBL  PO$ALC,PRIVCO,PO$BYP,PO$NFR,PO$NFW
                                        .GLOBL  CHKABT,UREGO,NUMDEV,LSTFDT,LSTFDD
                                        .GLOBL  TK3SVL,FD$TIM,SYTIMH,SYTIML,FC$CDX
                                        .GLOBL  QNSPNX,JCDB,DVFLAG,DX$RAL,DX$NMT
                                        .GLOBL  PNAME,CHNNUM,SYINDX,SYUNIT,RUNFLG,AF$BYA
                                        .GLOBL  C.CSW,C.SBLK,C.LENG,C.USED,C.NUMQ,C.DEVQ
                                        .GLOBL  CS$NMX,CS$ENT,SERFLG,$KINIT,NLCHN
                                        .GLOBL  CS$OPN,CHKSD,CS$SPL,CSHCLN,CSHALC
                                        .GLOBL  LSW2,RESDEV,OKFILE,OF$DEV,LSW6
                                        .GLOBL  OF$UNT,OT$RON,OF$FLG,OF$FIL,OF$$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
```

```
                                ;
                                ;  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
                                ;
                                ;  Internal USR error checks.
                                ;  (Numbers correspond to KMON error table offsets)
                                ;
```

```
                                UERR1   =       -16     ;Can't find tentative file entry for file on close
                                UERR2   =       -17     ;File length in chan block not = to len in file entry
                                UERR3   =       -20     ;Highest block # written > file length
                                UERR4   =       -21     ;Empty file entry doesn't follow tentative file entry
                                UERR5   =       -22     ;Not tentative file entry during close
                                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.
                                ;
002050  010246                 USRINI: MOV     R2,-(SP)
002052  010346                         MOV     R3,-(SP)
                                ;
                                ;  Allocate tables within this overlay segment above top of code
                                ;
002054  012702  015254'                 MOV     #USRTOP,R2      ;Point above top of code in this segment
                                ;
                                ;  Allocate file directory cache table
                                ;
002060  010237  000000G                 MOV     R2,CSHHD        ;Set pointer to 1st free directory cache entry
002064  013700  000000G                 MOV     VNFCSH,R0       ;Get # cache entries to allocate
002070  010203                 1$:     MOV     R2,R3           ;Get address of current entry
002072  062703  000000G                 ADD     #FC$$SZ,R3      ;Get address of next entry
002076  020327  140000                  CMP     R3,#140000      ;Will current entry fit in overlay region?
002102  101403                         BLOS    2$              ;Br if yes
002104  162703  000000G                 SUB     #FC$$SZ,R3      ;No -- Backup pointer to current entry
002110  000406                         BR      3$
002112  010362  000000G         2$:     MOV     R3,FC$LNK(R2)   ;Make current entry point to next one
002116  005062  000000G                 CLR     FC$CDX(R2)      ;Say entry is currently free
002122  010302                         MOV     R3,R2           ;Make next entry be current one
002124  077017                         SOB     R0,1$           ;Loop if more to allocate
002126  162703  000000G         3$:     SUB     #FC$$SZ,R3      ;Point to last entry
002132  005063  000000G                 CLR     FC$LNK(R3)      ;Make its forward link zero
                                ;
                                ;  Finished
                                ;
002136  012603                         MOV     (SP)+,R3
002140  012602                         MOV     (SP)+,R2
002142  000207                         RETURN
```

```
                                        .SBTTL  .ENTER
                                ;-------------------------------------------------------------------------------
                                ;  Open a new file.
                                ;
002344  004737  007362'         ENTER:  CALL    USRCOM          ;DO COMMON SETUP
002350  005037  000000G                 CLR     URO             ;FOR NOW, SAY FILE SIZE = 0
                                ;
                                ;  See if we have write access to the file.
                                ;
002354  032763  000000G 000000G         BIT     #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO THE FILE?
002362  001406                         BEQ     4$              ;BR IF YES
002364  004737  015214'                 CALL    ERRNAM          ;SET NAME OF FILE FOR TSKMON ERROR MESSAGE
002370  012700  177763                 MOV     #-15,R0         ;SET ABORT CODE
002374  000137  015144'                 JMP     USRCLS          ;ABORT OPERATION
                                ;
                                ;  See if it is a spooled device.
                                ;
002400  105737  000000G         4$:     TSTB    NSPLDV          ;Are there any spooled devices?
002404  001404                         BEQ     3$              ;Br if not
002406                                 OCALL   CHKSD           ;Are we opening to a spooled device?
002414  103074                         BCC     9$              ;Br if this is a spooled device
                                ;
                                ;  This is not a spooled device.
                                ;  See if this is a magnetic tape.
                                ;
002416  032764  000000G 000000G 3$:     BIT     #DS$NRD,DVSTAT(R4);MAG TAPE TYPE DIRECTORY DEVICE?
002424  001004                         BNE     7$              ;Br if yes
002426  032764  000000G 000000G         BIT     #DX$NRD,DVFLAG(R4);Internal flag set?
002434  001407                         BEQ     5$              ;BR IF NOT
002436  052763  000000G 000000G 7$:     BIS     #CS$ENT,C.CSW(R3);SET FLAG TO CAUSE TO CALL HANDLER ON CLOSE
002444  012702  000000G                 MOV     #DF$ENT,R2      ;GET CODE FOR ENTER
002450  000137  014010'                 JMP     SPLDIR          ;GO DO SPECIAL DEVICE ENTER
                                ;
                                ;  See if this is a file structured device.
                                ;
002454  032764  000000G 000000G 5$:     BIT     #DS$DIR,DVSTAT(R4);FILE STRUCTURED DEVICE?
002462  001451                         BEQ     9$              ;Br if not file structured
                                ;
                                ;  This is a file structured device.
                                ;  Look for a free slot large enough for the file.
                                ;
002464  013702  002044'                 MOV     ASNSIZ,R2       ;WAS SIZE SPECIFIED WITH ASSIGN STATEMENT?
002470  001002                         BNE     6$              ;BR IF YES
002472  013702  000004G                 MOV     EMTBLK+4,R2     ;GET REQUESTED SIZE FROM EMT ARG BLOCK
002476  004737  011324'         6$:     CALL    FNDFRE          ;TRY TO FIND A FREE SLOT
002502  103002                         BCC     1$              ;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.)
002504  000137  015144'                 JMP     USRCLS
                                ;
                                ;  We have found a free slot.
                                ;  Create a tenative file entry.
                                ;
002510  010237  000000G         1$:     MOV     R2,UR0          ;RETURN FILE SIZE IN USER'S R0
002514  004737  012026'                 CALL    ADDENT          ;CREATE A TENTATIVE FILE ENTRY
002520  103004                         BCC     2$              ;BR IF WE WERE SUCCESSFUL
                                ;  Error: Directory overflow -- Return error code 1.
```

.ENTER

```
     58 002522  012700  000001                MOV     #1,R0           ;SET ERROR CODE
     59 002526  000137  015144'                JMP     USRCLS          ;ABORT ENTER
     60                                ;
     61                                ;  Set up info in channel about file.
     62                                ;
     63 002532  013703  0000000     2$:        MOV     CHNADR,R3       ;ADDRESS OF CHANNEL BLOCK
     64 002536  013700  002002'                MOV     CURSEG,R0       ;CURRENT DIRECTORY SEGMENT #
     65 002542  000300                         SWAB    R0              ;POSITION SEG # TO LEFT BYTE
     66 002544  052700  0000000                BIS     #CS$ENT,R0      ;REMEMBER WE ARE DOING AN ENTER
     67 002550  050063  0000000                BIS     R0,C.CSW(R3)    ;SET UP CSW
     68 002554  010263  0000000                MOV     R2,C.LENG(R3)   ;SET UP ALLOCATED LENGTH OF FILE
     69 002560  004737  013420'                CALL    SBCALC          ;CALCULATE STARTING BLOCK # OF FILE
     70 002564  010002                         MOV     R0,R2           ;SAVE STARTING BLOCK NUMBER OF FILE
     71                                ;
     72                                ;  Give user a chance to diddle with directory entry.
     73                                ;
     74 002566  004737  013254'                CALL    DIDDLE          ;ALLOW USER TO DIDDLE WITH DIRECTORY ENTRY
     75                                ;
     76                                ;  Write out the directory segment.
     77                                ;
     78 002572  004737  013706'                CALL    WRTSEG          ;WRITE OUT UPDATED DIRECTORY SEGMENT
     79                                ;
     80                                ;  Set up file starting block number in channel
     81                                ;
     82 002576  010263  0000000                MOV     R2,C.SBLK(R3)   ;SET FILE STARTING BLOCK NUMBER IN CHANNEL
     83 002602  005063  0000000                CLR     C.USED(R3)      ;SAY NO BLOCKS WRITTEN TO FILE YET
     84                                ;
     85                                ;  Finished
     86                                ;
     87 002606  000137  015172'        9$:     JMP     USRXIT
```

```
                                        .SBTTL  .RENAME
                                ;-------------------------------------------------------------------------------
                                ;  Rename a file.
                                ;
002612  004737  007362'        RENAME: CALL    USRCOM          ;DO COMMON SETUP
                                ;
                                ;  See if this is a file structured device.
                                ;
002616  032764  000000G 000000G        BIT     #DS$DIR,DVSTAT(R4);FILE STRUCTURED DEVICE?
002624  001004                        BNE     1$              ;BR IF YES
002626  012700  000002              MOV     #2,R0           ;INVALID OPERATION
002632  000137  015144'              JMP     USRCLS
                                ;
                                ;  Make sure we have write access to old file.
                                ;
002636  032763  000000G 000000G 1$:     BIT     #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO OLD FILE?
002644  001063                        BNE     11$             ;BR IF NOT
                                ;
                                ;  Locate directory entry with old name.
                                ;  (Note: FILSPC currently contains old file name)
                                ;
002646  004737  011242'              CALL    FNDFIL          ;LOOK UP THE ENTRY
002652  103004                        BCC     2$              ;BR IF FOUND
002654  012700  000001              MOV     #1,R0           ;FILE NOT FOUND
002660  000137  015144'              JMP     USRCLS
                                ;  Save info about old file entry.
002664  010146                  2$:     MOV     R1,-(SP)        ;ADDRESS OF ITS DIRECTORY ENTRY
002666  010105                        MOV     R1,R5           ;CARRY ADDR OF DIR ENTRY IN R5 FOR A WHILE
002670  013746  002002'              MOV     CURSEG,-(SP)    ;SEGMENT NUMBER CONTAINING ENTRY
002674  010446                        MOV     R4,-(SP)        ;DEVICE TABLE INDEX
                                ;
                                ;  Remove old file entry from directory cache.
                                ;
002676  004737  014616'              CALL    CSHDEL          ;DELETE FILE ENTRY FROM CACHE TABLE
                                ;
                                ;  Set up new file spec.
                                ;
002702  013701  000020              MOV     EMTBLK+2,R1     ;ADDRESS OF FILE SPEC AREA
002706  042701  000001              BIC     #1,R1           ;MAKE SURE ADDRESS IS EVEN
002712  062701  000010              ADD     #10,R1          ;POINT TO NEW FILE SPEC
002716  004737  007532'              CALL    GETSPC          ;MOVE DEVICE-FILE SPEC TO FILSPC
002722  004737  010212'              CALL    CHKDEV          ;MAKE SURE DEVICE IS LEGAL
002726  103006                        BCC     3$              ;BR IF OK
002730  004737  015214'              CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
002734  012700  177776              MOV     #-2,R0          ;ILLEGAL DEVICE
002740  000137  015144'              JMP     USRCLS
                                ;
                                ;  Make sure user is authorized to access to new file.
                                ;
002744  004737  010566'          3$:     CALL    CHKACC          ;SEE IF USER IS AUTHORIZED FOR ACCESS
002750  032763  000000G 000000G        BIT     #CS$RON,C.CSW(R3);IS USER AUTHORIZED TO WRITE TO NEW FILE?
002756  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.
                                ;
```

```
     58 002760  020426                          CMP     R4,(SP)+        ;COMPARE DEVICE INDEX NUMBERS
     59 002762  001405                          BEQ     4$              ;BR IF OK
     60 002764  022626                  5$:     CMP     (SP)+,(SP)+    ;CLEAN OFF STACK
     61 002766  012700  000002                  MOV     #2,R0           ;INVALID OPERATION
     62 002772  000137  015144'                 JMP     USRCLS
     63 002776  120063  000000G         4$:     CMPB    R0,C.DEVQ(R3)   ;DO UNIT NUMBERS MATCH?
     64 003002  001370                          BNE     5$              ;BR IF DON'T MATCH
     65                                 ;
     66                                 ;  Make sure we have write access to new file.
     67                                 ;
     68 003004  032763  000000G 000000G        BIT     #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO NEW FILE?
     69 003012  001406                          BEQ     10$             ;BR IF YES
     70 003014  004737  015214'         11$:    CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
     71 003020  012700  177763                  MOV     #-15,R0         ;SET ABORT CODE
     72 003024  000137  015144'                 JMP     USRCLS          ;ABORT THE OPERATION
     73                                 ;
     74                                 ;  Delete any existing file with new file name.
     75                                 ;
     76                                 ;  Temporarily mark old file entry as empty so we won't find it when
     77                                 ;  searching for file that has same name as new file.
     78                                 ;  Note that this is safe since the USR is locked and noone else can
     79                                 ;  access the directory.
     80 003030  042765  000000G 000000G 10$:    BIC     #FS$PRM,FD$STA(R5);CLEAR PERMANENT-FILE STATUS FLAG
     81 003036  052765  000000G 000000G        BIS     #FS$EMP,FD$STA(R5);SET EMPTY-FILE STATUS FLAG
     82 003044  023727  002002' 000001        CMP     CURSEG,#1       ;IS THIS DIR ENTRY IN SEGMENT # 1?
     83 003052  001402                          BEQ     12$             ;BR IF YES -- IT WILL NOT BE REREAD BY FNDFIL
     84 003054  004737  013706'                 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 003060  004737  011242'         12$:    CALL    FNDFIL          ;LOOKUP NEW FILE NAME
     88 003064  103442                          BCS     9$              ;BR IF DON'T NEED TO DELETE
     89                                 ;
     90                                 ;  File with new name already exists.
     91                                 ;  See if it is protected.
     92                                 ;
     93 003066  032761  000000G 000000G        BIT     #FS$PRO,FD$STA(R1);IS FILE PROTECTED?
     94 003074  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 003076  012600                          MOV     (SP)+,R0        ;GET DIR SEG # FOR OLD FILE ENTRY
     98 003100  004737  013570'                 CALL    RDSEG           ;READ IN THE SEGMENT
     99 003104  012601                          MOV     (SP)+,R1        ;GET ADDRESS OF DIR ENTRY IN SEGMENT
    100 003106  042761  000000G 000000G        BIC     #FS$EMP,FD$STA(R1);CLEAR EMPTY-FILE FLAG
    101 003114  052761  000000G 000000G        BIS     #FS$PRM,FD$STA(R1);SET PERMANENT-FILE FLAG
    102 003122  004737  013706'                 CALL    WRTSEG          ;REWRITE THE DIR SEGMENT
    103 003126  012700  000003                  MOV     #3,R0           ;RETURN ERROR CODE 3
    104 003132  000137  015144'                 JMP     USRCLS
    105                                 ;  File is not protected.
    106                                 ;  Remove its entry from directory cache.
    107 003136  004737  014616'         13$:    CALL    CSHDEL          ;DELETE FILE ENTRY FROM CACHE TABLE
    108                                 ;  Mark its entry as empty.
    109 003142  012761  000000G 000000G        MOV     #FS$EMP,FD$STA(R1);FILE IS NOW DELETED
    110                                 ;  See if this entry is in same directory segment as the file being renamed.
    111                                 ;  If it is, bypass the consolidation & rewrite.
    112 003150  023716  002002'                 CMP     CURSEG,(SP)     ;IS DELETED FILE ENTRY IN SAME SEG AS RENAMED ENTRY?
    113 003154  001002                          BNE     6$              ;BR IF NOT
    114 003156  012600                          MOV     (SP)+,R0        ;POP SEGMENT NUMBER
```

```
    115 003160  000407                        BR      7$              ;GO FILL IN NEW NAME
    116                                ;  Consolidate & rewrite segment with deleted file entry.
    117 003162  004737  012634'        6$:     CALL    CONSOL          ;CONSOLIDATE DIRECTORY SEGMENT
    118 003166  004737  013706'                CALL    WRTSEG          ;WRITE IT OUT
    119                                ;
    120                                ;  Now change the name in the old file entry.
    121                                ;
    122 003172  012600                 9$:     MOV     (SP)+,R0        ;GET DIRECTORY SEGMENT #
    123 003174  004737  013570'                CALL    RDSEG           ;READ IN THE SEGMENT
    124 003200  012601                 7$:     MOV     (SP)+,R1        ;GET ADDRESS OF ENTRY WITHIN SEGMENT
    125 003202  042761  000000G 000000G       BIC     #FS$EMP,FD$STA(R1);CLEAR EMPTY-FILE FLAG
    126 003210  052761  000000G 000000G       BIS     #FS$PRM,FD$STA(R1);SET PERMANENT-FILE FLAG
    127 003216  010105                         MOV     R1,R5           ;GET ADDRESS OF DIR ENTRY BEING RENAMED
    128 003220  062705  0000000                ADD     #FD$NAM,R5      ;POINT TO FILE NAME AREA IN ENTRY
    129 003224  012702  000002G                MOV     #FILSPC+2,R2    ;POINT TO NEW FILE NAME
    130 003230  012225                         MOV     (R2)+,(R5)+     ;MOVE IN NEW NAME
    131 003232  012225                         MOV     (R2)+,(R5)+
    132 003234  011215                         MOV     (R2),(R5)
    133                                ;
    134                                ;  Give user a chance to diddle with the dir entry if he wishes.
    135                                ;
    136 003236  004737  013254'                CALL    DIDDLE          ;ALLOW USER TO DIDDLE
    137                                ;
    138                                ;  Add new file entry to cache table.
    139                                ;
    140 003242  004737  014466'                CALL    CSHADD          ;ADD NEW FILE ENTRY TO CACHE TABLE
    141                                ;
    142                                ;  Consolidate and rewrite directory segment.
    143                                ;
    144 003246  004737  012634'                CALL    CONSOL          ;CONSOLIDATE DIRECTORY SEGMENT
    145 003252  004737  013706'                CALL    WRTSEG          ;WRITE IT OUT
    146                                ;
    147                                ;  Finished
    148                                ;
    149 003256  005063  0000000                CLR     C.CSW(R3)       ;PURGE THE CHANNEL
    150 003262  000137  015172'                JMP     USRXIT
```

```
                                        .SBTTL  .DELETE
                                ;-------------------------------------------------------------------------------
                                ;  Delete a permanent file entry.
                                ;
003266  004737  007362'         DELETE: CALL    USRCOM          ;DO COMMON SETUP
                                ;
                                ;  See if we have write access to file being deleted.
                                ;
003272  032763  000000G 000000G         BIT     #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO FILE?
003300  001406                        BEQ     2$              ;BR IF YES
003302  004737  015214'               CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
003306  012700  177763                MOV     #-15,R0         ;SET ABORT ERROR CODE
003312  000137  015144'               JMP     USRCLS          ;ABORT THE OPERATION
                                ;
                                ;  See if device is a magnetic tape.
                                ;
003316  032764  000000G 000000G 2$:     BIT     #DS$NRD,DVSTAT(R4);IS THIS A MAG TAPE DEVICE?
003324  001004                        BNE     6$              ;Br if yes
003326  032764  000000G 000000G         BIT     #DX$NRD,DVFLAG(R4);Internal flag set?
003334  001404                        BEQ     5$              ;BR IF NOT
003336  012702  000000G         6$:     MOV     #DF$DEL,R2      ;SET DELETE FUNCTION CODE
003342  000137  014010'               JMP     SPLDIR          ;DO SPECIAL DEVICE DELETE
                                ;
                                ;  See if this is a file structured device.
                                ;
003346  032764  000000G 000000G 5$:     BIT     #DS$DIR,DVSTAT(R4);IS THIS A DIRECTORY STRUCTURED DEVICE?
003354  001004                        BNE     1$              ;BR IF YES
                                ;
                                ;  Delete is invalid on non-file-structured device.
                                ;
003356  012700  000002                MOV     #2,R0           ;RETURN INVALID-OPERATION ERROR CODE
003362  000137  015144'               JMP     USRCLS
                                ;
                                ;  Try to locate directory entry for file being deleted.
                                ;
003366  004737  011242'         1$:     CALL    FNDFIL          ;LOOK UP THE FILE
003372  103425                        BCS     9$              ;BR IF FILE DOES NOT EXIST
                                ;
                                ;  See if file is protected.
                                ;
003374  032761  000000G 000000G         BIT     #FS$PRO,FD$STA(R1);IS FILE PROTECTED?
003402  001404                        BEQ     3$              ;BR IF NOT
003404  012700  000003                MOV     #3,R0           ;RETURN ERROR CODE 3 IF FILE IS PROTECTED
003410  000137  015144'               JMP     USRCLS
                                ;
                                ;  Remove file entry from directory cache.
                                ;
003414  004737  014616'         3$:     CALL    CSHDEL          ;REMOVE FILE ENTRY FROM DIRECTORY CACHE
                                ;
                                ;  File exists.  Mark its entry as empty.
                                ;
003420  012761  000000G 000000G         MOV     #FS$EMP,FD$STA(R1);MARK DIR ENTRY AS EMPTY
                                ;
                                ;  Consolidate and rewrite the directory segment.
                                ;
003426  004737  012634'               CALL    CONSOL          ;CONSOLIDATE THE SEGMENT
003432  004737  013706'               CALL    WRTSEG          ;WRITE SEGMENT TO DISK
```

.DELETE

```
                                ;
                                ;  Finished
                                ;
003436  005063  000000                CLR     C.CSW(R3)       ;PURGE THE CHANNEL
003442  000137  015172'                JMP     USRXIT
                                ;
                                ;  File does not exist.
                                ;
003446  012700  000001          9$:   MOV     #1,R0           ;FILE DOES NOT EXIST
003452  000137  015144'                JMP     USRCLS
```

```
                                        .SBTTL  .CLOSE
                                ;-------------------------------------------------------------------------------
                                ;  Close a channel.
                                ;
003456  013703  0000000         CLOSE:  MOV     CHNADR,R3       ;GET ADDRESS OF CHANNEL BLOCK
003462  032763  0000000 0000000         BIT     #CS$OPN,C.CSW(R3);IS THE CHANNEL OPEN?
003470  001002                         BNE     7$              ;BR IF CHANNEL IS OPEN
003472  000137  004202'                 JMP     10$             ;NOP IF NOT OPEN
                                ;
                                ;  Wait for all I/O to finish on this channel
                                ;
003476  004737  0000000         7$:     CALL    IOWAIT          ;WAIT FOR ALL I/O ON CHANNEL TO FINISH
                                ;
                                ;  See if we are closing a spooled device.
                                ;
003502  032763  0000000 0000000         BIT     #CS$SPL,C.CSW(R3);IS THIS A SPOOLED DEVICE CHANNEL?
003510  001405                         BEQ     1$              ;BR IF NOT
003512                                  OCALL   SDCLOS          ;CLOSE A SPOOLED DEVICE CHANNEL
003520  000137  004176'                 JMP     9$
                                ;
                                ;  See if device being closed is a magnetic tape.
                                ;
003524  016300  0000000         1$:     MOV     C.CSW(R3),R0    ;GET CSW
003530  042700  000000C                 BIC     #^C<CS$NMX>,R0  ;MASK OUT ALL BUT DEVICE TABLE INDEX
003534  032760  0000000 0000000         BIT     #DS$NRD,DVSTAT(R0);IS DEVICE A MAG TAPE?
003542  001004                         BNE     18$             ;Br if yes
003544  032760  0000000 0000000         BIT     #DX$NRD,DVFLAG(R0);Internal flag set?
003552  001404                         BEQ     6$              ;BR IF NOT
003554  012702  0000000         18$:    MOV     #DF$CLS,R2      ;DO SPECIAL DEVICE CLOSE
003560  000137  014010'                 JMP     SPLDIR
                                ;
                                ;  See if we are closing a shared file.
                                ;
003564  005737  0000000         6$:     TST     JCDB            ;Does job have any shared file chans open?
003570  001402                         BEQ     17$             ;BR IF NOT
003572  004777  0000000                 CALL    @SFCLS          ;CHECK FOR CLOSING A SHARED FILE
                                ;
                                ;  See if this file was opened with a .LOOKUP or .ENTER
                                ;
003576  032763  0000000 0000000 17$:    BIT     #CS$ENT,C.CSW(R3);DID WE DO A .ENTER ON THE CHANNEL?
003604  001574                         BEQ     9$              ;BR IF NOT
                                ;
                                ;  We are closing a new file on a directory structured device.
                                ;  Gain exclusive access to USR data base.
                                ;
003606  004737  007770'                 CALL    GETUSR          ;CLAIM EXCLUSIVE ACCESS TO USR
                                ;
                                ;  Set up device # and unit # in FILDVU.
                                ;
003612  116337  0000000 002015'         MOVB    C.DEVQ(R3),FILDVU+1;SET UNIT #
003620  016300  0000000                 MOV     C.CSW(R3),R0    ;GET CHANNEL STATUS WORD
003624  042700  000000C                 BIC     #^C<CS$NMX>,R0  ;MASK OUT ALL BUT DEVICE INDEX #
003630  110037  002014'                 MOVB    R0,FILDVU       ;SET DEVICE INDEX #
                                ;
                                ;  Set up info about this directory
                                ;
003634  005063  0000000                 CLR     C.SBLK(R3)      ;SAY CHANNEL BEING USED FOR DIRECTORY OP
```

.CLOSE

```
     58 003640  042763  000000C 000000G          BIC     #<CS$EOF!CS$ERR>,C.CSW(R3) ;IGNORE PRIOR ERROR
     59 003646  004737  013454'                 CALL    RDSEG1              ;READ SEG1 AND SET DIRSIZ
     60                                 ;
     61                                 ;  Find the tentative file entry.
     62                                 ;
     63 003652  016300  0000000                 MOV     C.CSW(R3),R0        ;GET CHANNEL STATUS WORD
     64 003656  042700  160377                  BIC     #^C<17400>,R0       ;GET DIRECTORY SEGMENT # WITH TENTATIVE ENTRY
     65 003662  000300                          SWAB    R0                  ;RIGHT-JUSTIFY SEG #
     66 003664  004737  013570'                 CALL    RDSEG               ;READ IN THAT SEGMENT
     67 003670  013702  0000000                 MOV     CHNNUM,R2           ;GET OUR CHANNEL #
     68 003674  012700  0000000         2$:     MOV     #FS$TEN,R0          ;FIND NEXT TENTATIVE FILE ENTRY
     69 003700  004737  013166'                 CALL    GETDIR
     70 003704  103540                          BCS     11$                 ;BR IF CAN'T FIND TENTATIVE FILE ENTRY
     71 003706  120261  0000000                 CMPB    R2,FD$CHN(R1)       ;IS THIS ENTRY FOR RIGHT CHANNEL?
     72 003712  001370                          BNE     2$                  ;BR IF NOT
     73 003714  123761  0000000 0000000         CMPB    CORUSR,FD$JOB(R1);IS THIS FILE FOR RIGHT JOB?
     74 003722  001364                          BNE     2$                  ;BR IF NOT
     75                                 ;  Found the tentative file entry.
     76                                 ;  Save the file name in FILSPC.
     77 003724  010102                          MOV     R1,R2               ;ADDRESS OF TENTATIVE FILE ENTRY
     78 003726  062702  0000000                 ADD     #FD$NAM,R2          ;POINT TO NAME IN ENTRY
     79 003732  012704  0000020                 MOV     #FILSPC+2,R4        ;SAVE THE NAME HERE
     80 003736  012224                          MOV     (R2)+,(R4)+         ;SAVE FILE NAME IN FILSPC
     81 003740  012224                          MOV     (R2)+,(R4)+
     82 003742  011214                          MOV     (R2),(R4)
     83                                 ;  Remember the position of the tentative file entry.
     84 003744  010146                          MOV     R1,-(SP)
     85 003746  013746  002002'                 MOV     CURSEG,-(SP)
     86                                 ;
     87                                 ;  Delete any permanent file entry that has the same name as the new file.
     88                                 ;
     89 003752  004737  011242'                 CALL    FNDFIL              ;SEARCH FOR PERM FILE ENTRY WITH THIS NAME
     90 003756  103416                          BCS     5$                  ;BR IF NO PERM FILE ENTRY WITH SAME NAME
     91                                 ;  Delete old file entry from directory cache.
     92 003760  004737  014616'                 CALL    CSHDEL              ;DELETE OLD FILE ENTRY FROM DIRECTORY CACHE
     93                                 ;  Mark old file as deleted.
     94 003764  012761  0000000 0000000        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 003772  023716  002002'                 CMP     CURSEG,(SP)         ;IS OLD SEG SAME AS NEW ONE?
     97 003776  001002                          BNE     3$                  ;BR IF NOT
     98 004000  005726                          TST     (SP)+               ;POP NEW SEG #
     99 004002  000407                          BR      4$
    100 004004  004737  012634'         3$:     CALL    CONSOL              ;CONSOLIDATE THE OLD SEGMENT
    101 004010  004737  013706'                 CALL    WRTSEG              ;WRITE OUT THE OLD SEGMENT
    102 004014  012600                  5$:     MOV     (SP)+,R0            ;GET # OF NEW SEGMENT
    103 004016  004737  013570'                 CALL    RDSEG               ;READ IT IN
    104                                 ;
    105                                 ;  Convert tentative file entry to permanent.
    106                                 ;
    107 004022  012601                  4$:     MOV     (SP)+,R1            ;GET ADDRESS OF TENTATIVE FILE ENTRY
    108 004024  032761  0000000 0000000        BIT     #FS$TEN,FD$STA(R1);MAKE SURE WE'RE POINTING TO TENTATIVE ENTRY
    109 004032  001501                          BEQ     15$                 ;BAD ERROR IF NOT
    110 004034  012761  0000000 0000000        MOV     #FS$PRM,FD$STA(R1);SET ITS TYPE TO PERMANENT FILE
    111 004042  016102  0000000                 MOV     FD$LEN(R1),R2       ;GET ALLOCATED SIZE OF FILE
    112 004046  020263  0000000                 CMP     R2,C.LENG(R3)       ;MAKE SURE FILE LENGTHS AGREE
    113 004052  001060                          BNE     12$                 ;BAD ERROR IF NOT
    114 004054  016300  0000000                 MOV     C.USED(R3),R0       ;GET # BLOCKS ACTUALLY USED IN FILE
```

.CLOSE

```
115 004060  010061  000000G                 MOV     R0,FD$LEN(R1)   ;SET THIS AS THE ACTUAL FILE LENGTH
116 004064  020002                          CMP     R0,R2           ;MAKE SURE HIGHEST BLOCK NOT ABOVE FILE LENGTH
117 004066  101055                          BHI     13$             ;ERROR IF TOO BIG
118 004070  160002                          SUB     R0,R2           ;GET # BLOCKS LEFT OVER
119 004072  063701  002006'                 ADD     DIRSIZ,R1       ;POINT TO EMPTY FILE ENTRY THAT FOLLOWS
120 004076  032761  000000G 000000G         BIT     #FS$EMP,FD$STA(R1);MAKE SURE THIS REALLY IS AN EMPTY ENTRY
121 004104  001451                          BEQ     14$             ;BAD ERROR IF NOT
122 004106  060261  000000G                 ADD     R2,FD$LEN(R1)   ;ADD RESIDUAL BLOCKS TO EMPTY ENTRY
123                             ;  Set date and time of file creation in file directory entry.
124 004112  163701  002006'                 SUB     DIRSIZ,R1       ;POINT BACK TO NEW DIRECTORY ENTRY
125 004116  005061  000000G                 CLR     FD$TIM(R1)      ;SAY CREATION TIME UNKNOWN
126 004122  005761  000000G                 TST     FD$DAT(R1)      ;DID DIDDLE ROUTINE SET FILE DATE?
127 004126  001013                          BNE     16$             ;BR IF YES
128 004130  013761  000000G 000000G         MOV     SYSDAT,FD$DAT(R1);SET DATA THAT FILE WAS CREATED (CLOSED)
129 004136  013704  000000G                 MOV     SYTIMH,R4       ;GET CURRENT TIME OF DAY (HIGH ORDER)
130 004142  013705  000000G                 MOV     SYTIML,R5       ;GET LOW-ORDER TIME OF DAY
131 004146  071437  000000G                 DIV     TK3SVL,R4       ;CONVERT TO 3-SECOND UNITS
132 004152  010461  000000G                 MOV     R4,FD$TIM(R1)   ;SET TIME OF FILE CREATION
133                             ;  Add entry for new file to directory cache.
134 004156  004737  014466'     16$:        CALL    CSHADD          ;ADD TO CACHE TABLE
135                             ;  Consolidate and rewrite the directory segment.
136 004162  004737  012634'                 CALL    CONSOL          ;CONSOLIDATE THE DIRECTORY
137 004166  004737  013706'                 CALL    WRTSEG          ;WRITE OUT THE SEGMENT
138                             ;
139                             ;  Free the USR
140                             ;
141 004172  004737  010132'                 CALL    FREUSR          ;RELEASE USR DATA BASE
142                             ;
143                             ;  Mark the channel as closed.
144                             ;
145 004176  005063  000000G     9$:         CLR     C.CSW(R3)       ;SAY THE CHANNEL IS CLOSED
146                             ;
147                             ;  Finished
148                             ;
149 004202  000137  000000G     10$:        JMP     EMTXIT
150                             ;
151                             ;  Consistency check errors -- May indicate hardware errors or bugs.
152                             ;
153                             ;  Can't find tentative file during close.
154 004206  012700  177762      11$:        MOV     #UERR1,R0
155 004212  000413                          BR      20$
156                             ;  File size in channel block doesn't agree with size in directory entry.
157 004214  012700  177761      12$:        MOV     #UERR2,R0
158 004220  000410                          BR      20$
159                             ;  Highest block # written is greater than file length.
160 004222  012700  177760      13$:        MOV     #UERR3,R0
161 004226  000405                          BR      20$
162                             ;  Empty file entry does not follow tentative file entry.
163 004230  012700  177757      14$:        MOV     #UERR4,R0
164 004234  000402                          BR      20$
165                             ;  Not pointing to tentative file entry during close.
166 004236  012700  177756      15$:        MOV     #UERR5,R0
167 004242  000137  015202'     20$:        JMP     UABORT          ;GIVE FATAL ABORT
```

```
                                .SBTTL  .DSTATUS
                        ;-------------------------------------------------------------------------
                        ;  .DSTATUS -- Determine device status.
                        ;
004246  004737  007770' DSTAT:  CALL    GETUSR          ;GET EXCLUSIVE ACCESS TO USR DATA BASE
                        ;  Move device name to FILSPC buffer and perform any assigns.
004252  013701  000000G        MOV     URO,R1          ;POINT TO DEVICE SPEC
004256  004737  007532'        CALL    GETSPC          ;MOVE TO FILSPC BUFFER
                        ;  Look up device in perm name table.
004262  004737  010212'        CALL    CHKDEV          ;LOOK UP THE DEVICE NAME
004266  103003                 BCC     1$              ;BR IF FOUND
                        ;  Invalid device name
004270  005000                 CLR     R0              ;SET ERROR CODE
004272  000137  015156'        JMP     USRERR
                        ;
                        ;  Valid device.
                        ;  Return info about it.
                        ;  (R4 now has device table index)
                        ;
004276  010005          1$:     MOV     R0,R5           ;CARRY UNIT NUMBER IN R5
004300  013700  000004G        MOV     EMTBLK+4,R0     ;GET POINTER TO USER'S RESULT AREA
004304  010037  000000G        MOV     R0,URO          ;RETURN POINTER TO RESULT IN R0
004310  004737  000000G        CALL    VALADW          ;VALIDATE THE ADDRESS
004314  016446  000000G        MOV     DVSTAT(R4),-(SP);DEVICE STATUS WORD
004320  106620                 MTPD    (R0)+
004322  016446  000000G        MOV     HANSIZ(R4),-(SP);HANDLER SIZE
004326  106620                 MTPD    (R0)+
004330  016446  000000G        MOV     HANENT(R4),-(SP);HANDLER ENTRY POINT
004334  106620                 MTPD    (R0)+
004336  016446  000000G        MOV     DEVSIZ(R4),-(SP);DEVICE SIZE
004342  106620                 MTPD    (R0)+
                        ;
                        ;  If this is a logical disk, return file size as device size
                        ;
004344  020437  000000G        CMP     R4,LDDEVX       ;IS THIS A LOGICAL DISK?
004350  001004                 BNE     2$              ;BR IF NOT
004352  006305                 ASL     R5              ;CONVERT UNIT # TO WORD TABLE INDEX
004354  016546  000000G        MOV     LDSIZE(R5),-(SP);GET LOGICAL DISK SIZE
004360  106640                 MTPD    -(R0)           ;PASS TO USER
                        ;
                        ;  Finished
                        ;
004362  000137  015172' 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.
                                ;
004366  004737  007770'         FETCH:  CALL    GETUSR          ;Get exclusive access to USR data base
                                ;
                                ;  Move device name to FILSPC buffer and perform any assigns
                                ;
004372  013701  0000000             MOV     URO,R1          ;Point to device spec
004376  004737  007532'             CALL    GETSPC          ;Move to filspc buffer
                                ;
                                ;  Look up device in perm name table
                                ;
004402  004737  010212'             CALL    CHKDEV          ;Look up the device name
004406  103003                      BCC     1$              ;Br if found
                                ;
                                ;  Invalid device
                                ;
004410  005000                      CLR     R0              ;Set error code = 0
004412  000137  015156'             JMP     USRERR          ;Return with error
                                ;
                                ;  The device name is valid.
                                ;  Return handler load address to user in R0.
                                ;
004416  013700  0000040         1$:     MOV     EMTBLK+4,R0     ;Get user's load address
004422  005200                      INC     R0              ;Round up
004424  042700  000001              BIC     #1,R0           ;And force to even address
004430  010037  0000000             MOV     R0,URO          ;And return as top of handler
004434  000137  015172'             JMP     USRXIT          ;Exit from EMT
```

```
                                        .SBTTL  .SFDAT, .SFTIM, & .FPROT
                                ;-------------------------------------------------------------------------
                                ;  .SFDAT  --  Set date in file entry
                                ;
004440  004737  004704'         SFDATE: CALL    SFCOM           ;LOCATE FILE ENTRY
004444  004737  004754'                 CALL    SFWRIT          ;MAKE SURE WE HAVE WRITE ACCESS
004450  013700  0000040                MOV     EMTBLK+4,R0     ;GET SPECIFIED DATE VALUE
004454  001002                          BNE     1$              ;BR IF DATE SPECIFIED
004456  013700  0000000                MOV     SYSDAT,R0       ;USE CURRENT DATE
004462  016105  0000000         1$:     MOV     FD$DAT(R1),R5   ;SAVE OLD FILE DATE
004466  010061  0000000                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
                                ;
004472  004737  014466'         SFEXIT: CALL    CSHADD          ;ADD ENTRY TO CACHE
                                ;
                                ;  Rewrite directory segment
                                ;
004476  004737  013706'                 CALL WRTSEG             ;REWRITE DIRECTORY SEGMENT
                                ;
                                ;  Finished
                                ;
004502  005063  0000000                 CLR     C.CSW(R3)       ;SAY CHANNEL IS CLOSED
004506  000137  015172'                 JMP     USRXIT          ;EXIT

                                ;-------------------------------------------------------------------------
                                ;  .SFTIM -- Set file time in file entry
                                ;
004512  004737  004704'         SFTIME: CALL    SFCOM           ;LOCATE FILE ENTRY
004516  004737  004754'                 CALL    SFWRIT          ;MAKE SURE WE HAVE WRITE ACCESS
004522  013761  0000040 0000000        MOV     EMTBLK+4,FD$TIM(R1) ;SET TIME VALUE IN DIRECTORY ENTRY
004530  000760                        BR      SFEXIT          ;REWRITE DIRECTORY SEGMENT

                                ;-------------------------------------------------------------------------
                                ;  .FPROT -- Set file protection
                                ;
004532  004737  004704'         SFPROT: CALL    SFCOM           ;DO COMMON SETUP
004536  004737  004754'                 CALL    SFWRIT          ;MAKE SURE WE HAVE WRITE ACCESS
004542  113700  0000040                MOVB    EMTBLK+4,R0     ;GET PROTECT/UNPROTECT FLAG
004546  001407                          BEQ     2$              ;BR IF UNPROTECT WANTED
004550  020027  000001                 CMP     R0,#1           ;VALUE MUST BE 0 OR 1
004554  001410                          BEQ     3$              ;BR IF PROTECT WANTED
004556  012700  000003                 MOV     #3,R0           ;ERROR IF NOT 0 OR 1
004562  000137  015144'                 JMP     USRCLS
004566  042761  0000000 0000000 2$:     BIC     #FS$PRO,FD$STA(R1) ;UNPROTECT THE FILE
```

```
     58 004574  000736                          BR      SFEXIT          ;GO EXIT
     59 004576  052761  0000000 0000000 3$:     BIS     #FS$PRO,FD$STA(R1) ;PROTECT THE FILE
     60 004604  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.
                                ;
004606  004737  004704'        GFINFO: CALL    SFCOM           ;LOCATE FILE ENTRY
004612  013700  000004G               MOV     EMTBLK+4,R0     ;GET ADDRESS OF USER'S INFO BUFFER
004616  004737  000000G               CALL    VALADW          ;MAKE SURE ADDRESS IS VALID
                                ;  File size
004622  016146  000000G               MOV     FD$LEN(R1),-(SP);FILE SIZE
004626  106620                       MTPD    (R0)+
                                ;  Protected/Unprotected status
004630  005046                       CLR     -(SP)           ;ASSUME FILE IS NOT PROTECTED
004632  032761  000000G 000000G      BIT     #FS$PRO,FD$STA(R1);IS FILE PROTECTED?
004640  001401                       BEQ     1$              ;BR IF NOT
004642  005216                       INC     (SP)            ;RETURN 1 IF FILE IS PROTECTED
004644  106620                  1$:  MTPD    (R0)+
                                ;  Creation date
004646  016146  000000G               MOV     FD$DAT(R1),-(SP);CREATION DATE
004652  106620                       MTPD    (R0)+
                                ;  Creation time
004654  016146  000000G               MOV     FD$TIM(R1),-(SP);CREATION TIME
004660  106620                       MTPD    (R0)+
                                ;  Starting block number
004662  010002                       MOV     R0,R2           ;SAVE BUFFER POINTER
004664  004737  013420'               CALL    SBCALC          ;CALCULATE STARTING BLOCK NUMBER
004670  010046                       MOV     R0,-(SP)
004672  106622                       MTPD    (R2)+           ;RETURN STARTING BLOCK NUMBER
                                ;
                                ;  Finished
                                ;
004674  005063  000000G               CLR     C.CSW(R3)       ;SAY CHANNEL IS CLOSED
004700  000137  015172'               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.
                                ;
004704  004737  007362'         SFCOM:  CALL    USRCOM          ;DO USR ENTRY SETUP
                                ;
                                ;  Make sure this is a file structured device and we have write
                                ;  access to it.
                                ;
004710  032764  0000000 000000G         BIT     #DS$DIR,DVSTAT(R4) ;IS THIS A FILE STRUCTURED DEVICE?
004716  001004                        BNE     1$              ;BR IF YES
004720  012700  000002                MOV     #2,R0           ;RETURN ERROR CODE 2 IF NOT
004724  000137  015144'                JMP     USRCLS
                                ;
                                ;  Locate directory entry for the file
                                ;
004730  004737  011242'         1$:     CALL    FNDFIL          ;LOCATE THE DIRECTORY ENTRY
004734  103004                        BCC     3$              ;BR IF ENTRY FOR FILE FOUND
004736  012700  000001                MOV     #1,R0           ;ERROR 1 IF CAN'T FIND ENTRY
004742  000137  015144'                JMP     USRCLS
                                ;
                                ;  Remove directory entry from cache
                                ;
004746  004737  014616'         3$:     CALL    CSHDEL          ;REMOVE ENTRY FROM CACHE
                                ;
                                ;  Finished
                                ;
004752  000207                        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.
                                ;
004754  032763  0000000 000000G SFWRIT: BIT     #CS$RON,C.CSW(R3) ;DO WE HAVE WRITE ACCESS TO THE FILE?
004762  001406                        BEQ     1$              ;BR IF YES
004764  004737  015214'                CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
004770  012700  177763                MOV     #-15,R0         ;ERROR IF NOT
004774  000137  015144'                JMP     USRCLS
005000  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.
                ;
005002          ALCEMT:
                ;
                ;  Determine if this is an Allocate, Deallocate, or test allocation EMT
                ;
005002  113700  0000000         MOVB    EMTBLK,R0       ;Get sub-function code
005006  001410                  BEQ     10$             ;0==>Allocate
005010  120027  000001          CMPB    R0,#1           ;1==>Deallocate
005014  001475                  BEQ     DLCEMT
005016  120027  000002          CMPB    R0,#2           ;2==>Test allocation
005022  001567                  BEQ     TLCEMT
005024  000137  0000000         JMP     BADEMT          ;Invalid sub-function code
                ;
                ;  Allocate a device
                ;
005030  032737  0000000 0000000 10$:    BIT     #P0$ALC,PRIVC0  ;Are we authorized to allocate devices?
005036  001004                  BNE     11$             ;Br if yes
005040  012700  000005          MOV     #5,R0           ;Error code 5 if not authorized
005044  000137  0000000         JMP     SETERR
                ;
                ;  Do common allocate setup
                ;
005050  004737  005622'  11$:    CALL    ALCCOM          ;Do common setup
                ;
                ;  See if the device being allocated is currently allocated to another job
                ;
005054  004737  005422'         CALL    ALCTST          ;Test for allocation to another job
                ;
                ;  Device is not currently allocated.
                ;  Check to see if the device is already allocated by our job or a
                ;  related job.
                ;
005060  012702  0000000         MOV     #ALCTBL,R2      ;Point to start of allocation table
005064  013700  002014'         MOV     FILDVU,R0       ;Get device # and index #
005070  020062  0000000 4$:     CMP     R0,AD$DVU(R2)   ;Search for device in allocation table
005074  001406                  BEQ     5$              ;Br if found it
005076  062702  0000000         ADD     #AD$$SZ,R2      ;Point to next entry
005102  020227  0000000         CMP     R2,#ALCEND      ;Checked all entries?
005106  103770                  BLO     4$              ;Br if not
005110  000410                  BR      7$              ;Br if not currently in allocation table
                ;
                ;  Device is already in allocation table
                ;
005112  116200  0000000 5$:     MOVB    AD$JOB(R2),R0   ;Get # of job that owns device
005116  120160  0000000         CMPB    R1,LNPRIM(R0)   ;Is primary job reallocating device?
005122  001030                  BNE     9$              ;Br if not
005124  110162  0000000         MOVB    R1,AD$JOB(R2)   ;Let primary job take over device
005130  000425                  BR      9$
                ;
                ;  Device is not currently allocated.
                ;  Search for a free entry in the allocation table.
                ;
005132  012702  0000000 7$:     MOV     #ALCTBL,R2      ;Point to start of allocation table
```

```
     58 005136  105762  0000000          2$:     TSTB    AD$JOB(R2)          ;Is this entry free?
     59 005142  001411                           BEQ     3$                  ;Br if yes
     60 005144  062702  0000000                  ADD     #AD$$SZ,R2          ;Point to next table entry
     61 005150  020227  0000000                  CMP     R2,#ALCEND          ;Checked all entries?
     62 005154  103770                           BLO     2$                  ;Loop if more to check
     63 005156  012700  000003                   MOV     #3,R0               ;Error 3 -- Allocation table is full
     64 005162  000137  015156'                  JMP     USRERR              ;Error abort
     65                                     ;
     66                                     ;  Found a free entry, set it up for this device
     67                                     ;
     68 005166  013762  002014' 0000000  3$:     MOV     FILDVU,AD$DVU(R2);Store device and unit numbers
     69 005174  110162  0000000                  MOVB    R1,AD$JOB(R2)       ;Store job number
     70 005200  105062  0000000                  CLRB    AD$FLG(R2)          ;No flags yet
     71                                     ;
     72                                     ;  Finished
     73                                     ;
     74 005204  000137  015172'          9$:     JMP     USRXIT
```

```
                                        .SBTTL  DLCEMT -- Deallocate a device
                                ;-------------------------------------------------------------------------
                                ;  Deallocate a device.
                                ;
005210  032737  000000@ 000000@ DLCEMT: BIT     #PO$ALC,PRIVCO  ;Are we authorized to allocate devices?
005216  001004                         BNE     11$             ;Br if yes
005220  012700  000005                 MOV     #5,R0           ;Error code 5 if not authorized
005224  000137  000000@                 JMP     SETERR
005230  004737  005622'         11$:    CALL    ALCCOM          ;Do common setup
005234  012702  000000@                 MOV     #ALCTBL,R2      ;Point to start of allocation table
                                ;
                                ;  If the device name is null, deallocate all devices allocated by
                                ;  this user.
                                ;
005240  013700  000000@                 MOV     FILSPC,R0       ;Is device name null?
005244  001015                         BNE     2$              ;Br if not
005246  120162  000000@         1$:     CMPB    R1,AD$JOB(R2)   ;Is this entry an allocation for our job?
005252  001004                         BNE     3$              ;Br if not
005254  105062  000000@                 CLRB    AD$JOB(R2)      ;Say no job using this entry
005260  005062  000000@                 CLR     AD$DVU(R2)      ;Say entry is free
005264  062702  000000@         3$:     ADD     #AD$$SZ,R2      ;Point to next allocation table entry
005270  020227  000000@                 CMP     R2,#ALCEND      ;Checked all entries?
005274  103764                         BLO     1$              ;Loop if more to check
005276  000437                         BR      9$
                                ;
                                ;  The device name is not null.
                                ;  Deallocate the specified device.
                                ;
005300  105737  002014'         2$:     TSTB    FILDVU          ;Trying to deallocate TT?
005304  001434                         BEQ     9$              ;Ignore if yes
005306  013700  002014'                 MOV     FILDVU,R0       ;Get device/unit ID for device being dealloc
005312  020062  000000@         7$:     CMP     R0,AD$DVU(R2)   ;Search for specified device in alloc table
005316  001406                         BEQ     5$              ;Br if found it
005320  062702  000000@                 ADD     #AD$$SZ,R2      ;Point to next entry in table
005324  020227  000000@                 CMP     R2,#ALCEND      ;Checked all entries?
005330  103770                         BLO     7$              ;Keep looking if not
005332  000421                         BR      9$              ;Device was not allocated by anyone
005334  116200  000000@         5$:     MOVB    AD$JOB(R2),R0   ;Get # of job that owns this device
005340  126061  000000@ 000000@         CMPB    LNPRIM(R0),LNPRIM(R1) ;Is device owned by us?
005346  001407                         BEQ     6$              ;Br if yes
005350  006200                         ASR     R0              ;Convert job index # to job #
005352  010037  000000@                 MOV     R0,UR0          ;Return owner job # in R0
005356  012700  000001                 MOV     #1,R0           ;Error 1 -- Device is allocated to someone
005362  000137  015156'                 JMP     USRERR
005366  005062  000000@         6$:     CLR     AD$DVU(R2)      ;Say this table entry is free
005372  105062  000000@                 CLRB    AD$JOB(R2)
                                ;
                                ;  Finished
                                ;
005376  000137  015172'         9$:     JMP     USRXIT
```

```
      1                                       .SBTTL  TLCEMT -- Check to see if a device is allocated
      2                                 ;-----------------------------------------------------------------------
      3                                 ;  Check to see if a device is allocated by another user.
      4                                 ;
      5 005402  005037  0000000         TLCEMT: CLR     URO             ;Clear user's RO in case ALCCOM aborts
      6 005406  004737  005622'                CALL    ALCCOM          ;Do common setup
      7                                 ;
      8                                 ;  See if device is currently allocated
      9                                 ;
     10 005412  004737  005422'                CALL    ALCTST          ;See if device is allocated to another user
     11                                 ;
     12                                 ;  Finished -- Device is not allocated to any other user
     13                                 ;
     14 005416  000137  015172'                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.
                ;
005422  010246          ALCTST: MOV     R2,-(SP)
005424  010346                  MOV     R3,-(SP)
                ;
                ;  Clear returned value for R0 in case device is not in use or allocated
                ;
005426  005037  000000G         CLR     UR0             ;Say no use of device found yet
                ;
                ;  See if the device is legal
                ;
005432  005737  000000G         TST     FILSPC          ;Don't allow null device
005436  001413                  BEQ     11$             ;Error if name null
005440  105737  002014'          TSTB    FILDVU          ;Trying to allocate TT?
005444  001410                  BEQ     11$             ;Br if yes
005446  123737  002014' 000000G 6$:     CMPB    FILDVU,SYINDX   ;Is this the system device?
005454  001010                  BNE     12$             ;Br if not
005456  123737  002015' 000001G         CMPB    FILDVU+1,SYUNIT+1;Is this the system unit?
005464  001004                  BNE     12$             ;Br if not
005466  012700  000002          11$:    MOV     #2,R0           ;Invalid device if null
005472  000137  015156'                  JMP     USRERR
                ;
                ;  See if the device is currently allocated to any user
                ;
005476  012702  000000G         12$:    MOV     #ALCTBL,R2      ;Point to start of allocation table
005502  023762  002014' 000000G 1$:     CMP     FILDVU,AD$DVU(R2);Search for entry for this device and unit
005510  001423                  BEQ     5$              ;Br if found it
005512  062702  000000G                 ADD     #AD$$SZ,R2      ;Point to next entry
005516  020227  000000G                 CMP     R2,#ALCEND      ;Searched all entries?
005522  103767                  BLO     1$              ;Loop if not
                ;
                ;  Device is not currently allocated.
                ;  See if any other job has a channel open to this device.
                ;
005524  004737  005742'                 CALL    CHKUSE          ;See if device is being used by another job
005530  103007                  BCC     13$             ;Br if no other jobs have device open
005532  006200                  ASR     R0              ;Convert job index # to job #
005534  010037  000000G                 MOV     R0,UR0          ;Return in R0
005540  012700  000004                  MOV     #4,R0           ;Error 4 -- Device in use by another job
005544  000137  015156'                  JMP     USRERR
005550  006200          13$:    ASR     R0              ;Get # of job that is using the device
005552  010037  000000G                 MOV     R0,UR0          ;Return in R0
005556  000416                  BR      9$
                ;
```

```
                                        ;  The device is allocated.
                                        ;  See if it is allocated to our job or to another job.
                                        ;
005560  116203  0000000         5$:     MOVB    AD$JOB(R2),R3   ;Get # of job to which dev is allocated
005564  010337  0000000                 MOV     R3,UR0          ;Return job # to user in R0
005570  006237  0000000                 ASR     UR0
005574  126361  0000000 0000000         CMPB    LNPRIM(R3),LNPRIM(R1) ;Is dev allocated to our family?
005602  001404                          BEQ     9$              ;Br if yes
005604  012700  000001                  MOV     #1,R0           ;Error 1 -- Device is allocated to another job
005610  000137  015156'                 JMP     USRERR
                                        ;
                                        ;  Device is not allocated to another user
                                        ;
005614  012603                  9$:     MOV     (SP)+,R3
005616  012602                          MOV     (SP)+,R2
005620  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.
                ;
005622  010146  ALCCOM: MOV     R1,-(SP)
                ;
                ;  Gain exclusive access to USR data base
                ;
005624  004737  007770'         CALL    GETUSR          ;Get exclusive use of USR
                ;
                ;  Move file spec to FILSPC and apply any assigns
                ;
005630  013701  000002G         MOV     EMTBLK+2,R1     ;Get pointer to device spec argument
005634  004737  007532'         CALL    GETSPC          ;Setup FILSPC
                ;
                ;  Check to see if the device is valid
                ;
005640  005737  000000G         TST     FILSPC          ;Is the device null?
005644  001430                  BEQ     2$              ;Treat this as a special valid device
005646  004737  010212'         CALL    CHKDEV          ;See if device is valid
005652  103016                  BCC     1$              ;Br if CHKDEV says it is valid
005654  020437  000000G         CMP     R4,CLDEVX       ;Is this a CL unit?
005660  001004                  BNE     3$              ;Br if not
005662  020027  000000G         CMP     R0,#CLTOTL      ;Valid CL unit number?
005666  103417                  BLO     2$              ;Br if yes
005670  000412                  BR      8$              ;Invalid device
005672  020437  000000G 3$:     CMP     R4,C1DEVX       ;Is this a C1 unit?
005676  001007                  BNE     8$              ;Br if not
005700  020027  177770G         CMP     R0,#CLTOTL-8.   ;Is this a valid C1 unit number?
005704  002410                  BLT     2$              ;Br if yes
005706  000403                  BR      8$              ;Br if not
005710  020437  000000G 1$:     CMP     R4,LDDEVX       ;Is this a logical disk?
005714  001004                  BNE     2$              ;Br if not -- Don't allow alloc of LD
                ;
                ;  This is an invalid device
                ;
005716  012700  000002  8$:     MOV     #2,R0           ;Error 2 -- Invalid device
005722  000137  015156'         JMP     USRERR
                ;
                ;  This is a valid device, Build the FILDVU word
                ;
005726  110437  002014' 2$:     MOVB    R4,FILDVU       ;Set device index number
005732  110037  002015'         MOVB    R0,FILDVU+1     ;Set unit number
                ;
                ;  Finished
                ;
005736  012601                  MOV     (SP)+,R1
005740  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).
                                ;
005742  010246                  CHKUSE: MOV     R2,-(SP)
005744  010346                          MOV     R3,-(SP)
005746  010446                          MOV     R4,-(SP)
005750  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get exclusive access to job context block buffer
                                ;
005752                                  OCALL   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.
                                ;
005760  005004                          CLR     R4              ;Have not found any job accessing device
005762  013705  000000G                 MOV     CSHDEV,R5       ;Point to start of mount table
005766  005765  000000G         6$:     TST     CD$DVU(R5)      ;Is this mount entry in use?
005772  001430                          BEQ     7$              ;Br if not
005774  026537  000000G 002014'         CMP     CD$DVU(R5),FILDVU ;Does this match device of interest?
006002  001024                          BNE     7$              ;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.
                                ;
006004  012702  000000G                 MOV     #LSTSL,R2       ;Get index # of last job
006010  020201                  8$:     CMP     R2,R1           ;Are we checking current job?
006012  001415                          BEQ     10$             ;Br if yes -- It is ok
006014  010203                          MOV     R2,R3           ;Get index number
006016  010246                          MOV     R2,-(SP)        ;Preserve R2
006020  004737  015112'                 CALL    CDJFLG          ;Get pointer to flag for this job
006024  010200                          MOV     R2,R0           ;Get pointer to byte with flag bit
006026  012602                          MOV     (SP)+,R2        ;Recover job number
006030  130310                          BITB    R3,(R0)         ;Does this job have device mounted?
006032  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.
                                ;
006034  126261  000000G 000000G         CMPB    LNPRIM(R2),LNPRIM(R1) ;Is this job allowed to access the dev?
006042  001073                          BNE     5$              ;Br if not
006044  010204                          MOV     R2,R4           ;Remember # of job accessing device
006046  162702  000002          10$:    SUB     #2,R2           ;More jobs to check?
```

```
                        BGT     8$              ;Br if yes
                ;
                ;  Check next mount table entry
                ;
006054  062705  000000G 7$:    ADD     #CD$$SZ,R5      ;Point to next mount table entry
006060  020537  000000G        CMP     R5,CSHDVN       ;Have we checked all mount table entries?
006064  103740                BLO     6$              ;Br if not
                ;
                ;  Finished checking all mount table entries
                ;
006066  010405                MOV     R4,R5           ;Get # of any job accessing device
                ;
                ;  Begin loop to cycle through all jobs to see if any job has
                ;  a channel opened to this device.
                ;
006070  012702  000002        MOV     #2,R2           ;Get index # of 1st job
                ;
                ;  Ignore this job if it is not logged on.
                ;
006074  032762  000000G 000000G 1$:    BIT     #$KINIT,LSW(R2) ;Is this job logged on?
006102  001440                BEQ     2$              ;Br if not
                ;
                ;  Begin loop to cycle through all channels for this job
                ;
006104  005003                CLR     R3              ;Start with channel # 0
006106                  3$:     GCALL   GETCHA          ;Get address of this channel
                ;
                ;  Ignore this channel if it is not open
                ;
006114  032760  000000G 000000G        BIT     #CS$OPN,C.CSW(R0) ;Is this channel open?
006122  001424                BEQ     4$              ;Br if not
                ;
                ;  Get the device index number and the channel number out of the
                ;  channel block.
                ;
006124  010004                MOV     R0,R4           ;Get addr of channel block to R4
006126  016400  000000G        MOV     C.CSW(R4),R0    ;Get CSW
006132  042700  177701        BIC     #^C76,R0        ;Extract device index number
006136  116404  000000G        MOVB    C.DEVQ(R4),R4   ;Get unit number
006142  042704  177770        BIC     #^C7,R4         ;Clear all but unit number
006146  120037  002014'       CMPB    R0,FILDVU       ;Does it match the device of interest?
006152  001010                BNE     4$              ;Br if not
006154  120437  002015'       CMPB    R4,FILDVU+1     ;Does unit number match?
006160  001005                BNE     4$              ;Br if not
                ;
                ;  We found a job that has a channel open to the device
                ;
006162  126261  000000G 000000G        CMPB    LNPRIM(R2),LNPRIM(R1) ;Is this job allowed to access the dev?
006170  001020                BNE     5$              ;Br if not
006172  010205                MOV     R2,R5           ;Remember # of friendly job
                ;
                ;  Check next channel (including Command file, log file, spool control
                ;  and SAV file load channels)
                ;
006174  005203                4$:     INC     R3              ;Advance the channel number
006176  020327  000000G        CMP     R3,#NLCHN       ;Checked all of user's channels?
006202  103741                BLO     3$              ;Loop if more to check
```

```
                                        ;
                                        ;  Check next job
                                        ;
006204  062702  000002         2$:     ADD     #2,R2           ;Get # of next job
006210  020227  0000000                 CMP     R2,#LSTSL       ;Checked all jobs?
006214  101727                          BLOS    1$              ;Loop if more to check
                                        ;
                                        ;  Finished checking all jobs
                                        ;  See if a friendly job is using the device
                                        ;
006216                                  OCALL   FRECXT          ;Free context block buffer
006224  010500                          MOV     R5,R0           ;Get # of any job accessing the device
006226  000241                          CLC                     ;Set flag saying device is free
006230  000405                          BR      9$
                                        ;
                                        ;  We found a job accessing the channel who is in conflict with testing job
                                        ;
006232                          5$:     OCALL   FRECXT          ;Free context block buffer
006240  010200                          MOV     R2,R0           ;Return job index number in R0
006242  000261                          SEC                     ;Set flag saying that device is in use
                                        ;
                                        ;  Finished
                                        ;
006244  012605                  9$:     MOV     (SP)+,R5
006246  012604                          MOV     (SP)+,R4
006250  012603                          MOV     (SP)+,R3
006252  012602                          MOV     (SP)+,R2
006254  000207                          RETURN
```

```
                                        .SBTTL  MOUNT  -- Mount a new file structure
                                ;-------------------------------------------------------------------
                                ;  The MOUNT EMT is used to introduce a new file structure to the system.
                                ;
006256  004737  007120'         MOUNT:  CALL    MNTCOM          ;DO COMMON SETUP
                                ;
                                ;  First dismount the device to clear the directory file entries
                                ;
006262  004737  006732'                 CALL    DMTSUB          ;DISMOUNT THE DEVICE
                                ;
                                ;  See if this device is mounted by other users.
                                ;
006266  004737  014776'                 CALL    CSHTST          ;SEARCH FOR DEVICE IN MOUNT TABLE
006272  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.
                                ;
006274  113705  002014'                 MOVB    FILDVU,R5       ;GET DEVICE INDEX NUMBER
006300  016500  000000G                 MOV     DVSTAT(R5),R0   ;GET STATUS FLAGS FOR THIS DEVICE
006304  032700  000000G                 BIT     #DS$DIR,R0      ;IS THIS A DIRECTORY STRUCTURED DEVICE?
006310  001503                          BEQ     3$              ;BR IF NOT -- DON'T CACHE
006312  032700  000000G                 BIT     #DS$NRD,R0      ;NON RT-11 DIRECTORY STRUCTURE (MAG TAPE)?
006316  001100                          BNE     3$              ;BR IF YES -- DON'T CACHE
006320  032765  000000G 000000G         BIT     #DX$NRD,DVFLAG(R5);Internal non-RT-dir flag set?
006326  001074                          BNE     3$              ;Br if yes
006330  032765  000000G 000000G         BIT     #DX$NMT,DVFLAG(R5);Is it flagged for no mount?
006336  001070                          BNE     3$              ;Br if yes
                                ;
                                ;  Look for a free entry in the mount table.
                                ;
006340  013705  000000G                 MOV     CSHDEV,R5       ;POINT TO TABLE OF STRUCTURES
006344  005765  000000G         1$:     TST     CD$DVU(R5)      ;SEARCH FOR A FREE SLOT IN THE TABLE
006350  001411                          BEQ     2$              ;BR IF FOUND ONE
006352  062705  000000G                 ADD     #CD$$SZ,R5      ;POINT TO NEXT TABLE ENTRY
006356  020537  000000G                 CMP     R5,CSHDVN       ;HIT END OF TABLE?
006362  103770                          BLO     1$              ;BR IF NOT
006364  012700  000001                  MOV     #1,R0           ;NO FREE SLOTS IN TABLE
006370  000137  015156'                 JMP     USRERR
                                ;
                                ;  We found a free entry in the mount table.
                                ;  Set up the entry for this device.
                                ;
006374  004737  015054'         2$:     CALL    GETDVU          ;GET PHYSICAL DEVICE # AND UNIT #
006400  005065  000000G                 CLR     CD$NAM(R5)      ;CLEAR FILE NAME (ASSUME NOT LOGICAL DISK)
006404  005065  000002G                 CLR     CD$NAM+2(R5)
006410  012765  177777  000000G         MOV     #177777,CD$TOP(R5);ASSUME NOT LOGICAL DISK FOR TOP BLOCK
006416  010365  000000G                 MOV     R3,CD$DVU(R5)   ;STORE PHYSICAL DEVICE # AND UNIT #
006422  010465  000000G                 MOV     R4,CD$BAS(R5)   ;STORE BASE BLOCK # IF LOGICAL DISK
006426  001420                          BEQ     6$              ;BR IF NOT A LOGICAL DISK
006430  113703  002015'                 MOVB    FILDVU+1,R3     ;THIS IS A LOGICAL DISK, GET LD UNIT #
006434  006303                          ASL     R3              ;CONVERT UNIT # TO WORD TABLE INDEX
006436  066304  000000G                 ADD     LDSIZE(R3),R4   ;GET # OF BLOCK ABOVE TOP OF LOGICAL DISK
006442  010465  000000G                 MOV     R4,CD$TOP(R5)   ;SET TOP BLOCK NUMBER OF LOGICAL DISK
006446  072327  000002                  ASH     #2,R3           ;TIMES 8 BYTES PER ENTRY
006452  062703  000000G                 ADD     #LDNAME,R3      ;POINT TO ENTRY FOR NAME OF THIS LD FILE
006456  005723                          TST     (R3)+           ;SKIP OVER DEVICE NAME
```

```
     58 006460  012365  0000000                  MOV     (R3)+,CD$NAM(R5);STORE 1ST 3 CHARS OF FILE NAME
     59 006464  011365  0000020                  MOV     (R3),CD$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 006470  010503                  6$:     MOV     R5,R3           ;POINT TO SET OF MOUNT-FLAGS FOR
     65 006472  062703  0000000                  ADD     #CD$JOB,R3      ;  ALL JOBS
     66 006476  012700  0000000                  MOV     #CD$$UB,R0      ;GET # BYTES USED FOR MOUNT FLAGS
     67 006502  105023                  5$:     CLRB    (R3)+           ;SAY DEVICE NOT MOUNTED BY ANY JOBS
     68 006504  077002                          SOB     R0,5$
     69                                 ;
     70                                 ;  Set mount flag for our job
     71                                 ;
     72 006506  113703  0000000         4$:     MOVB    CORUSR,R3       ;Get our job index number
     73 006512  004737  015112'                 CALL    CDJFLG          ;GET MOUNT FLAG FOR OUR JOB
     74 006516  150312                          BISB    R3,(R2)         ;SET MOUNT FLAG FOR OUR JOB
     75 006520  000137  015172'         3$:     JMP     USRXIT
```

```
                        .SBTTL  DISMNT -- Dismount a file structure
                ;-----------------------------------------------------------------------
                ;  The DISMNT EMT is used to remove a file structure from the system tables.
                ;
006524          DISMNT:
                ;
                ;  Determine if this is a request to dismount a specific file structure
                ;  or a request to clean out the directory cache.
                ;
006524  113700  000000G         MOVB    EMTBLK,R0       ;Get sub-function code
006530  120027  000001          CMPB    R0,#1           ;Clean out directory cache?
006534  001430                  BEQ     CSHDMT          ;Br if yes
006536  120027  000002          CMPB    R0,#2           ;Dismount all devices for job?
006542  001442                  BEQ     DMTALL          ;Br if yes
006544  120027  000003          CMPB    R0,#3           ;Dismount a LD structure?
006550  001414                  BEQ     1$              ;Br if yes
006552  120027  000004          CMPB    R0,#4           ;Get information about a LD structure?
006556  001413                  BEQ     2$              ;Br if yes
006560  120027  000005          CMPB    R0,#5           ;Dismount all LD structures?
006564  001412                  BEQ     3$              ;Br if yes
                ;
                ;  This is a request to dismount a file structure
                ;
006566  004737  007120'         CALL    MNTCOM          ;Do common setup
                ;
                ;  Call DMTSUB subroutine to do real dismount work.
                ;
006572  004737  006732'         CALL    DMTSUB          ;Do the dismount
                ;
                ;  Finished
                ;
006576  000137  015172'         JMP     USRXIT
                ;
                ;  Dismount a LD structure
                ;
006602  000137  000000G 1$:     JMP     LDDEMT          ;Dismount a LD
                ;
                ;  Get information about a LD structure
                ;
006606  000137  000000G 2$:     JMP     LDIEMT          ;Get info about a LD
                ;
                ;  Dismount all LD structures
                ;
006612  000137  000000G 3$:     JMP     LDAEMT          ;Dismount all LD's

                ;-----------------------------------------------------------------------
                ;  CSHDMT removes all entries from the directory cache.
                ;
006616  004737  007770' CSHDMT: CALL    GETUSR          ;GET EXCLUSIVE USE OF USR
006622  013700  000000G         MOV     CSHHD,R0        ;POINT TO 1ST CACHED DIRECTORY ENTRY
006626  005060  000000G 1$:     CLR     FC$CDX(R0)      ;SAY ENTRY IS EMPTY
006632  005060  000000G         CLR     FD$NAM(R0)
006636  016000  000000G         MOV     FC$LNK(R0),R0   ;LINK TO NEXT ENTRY
006642  001371                  BNE     1$              ;BRANCH BACK IF ANOTHER TO FREE
006644  000137  015172'         JMP     USRXIT          ;FINISHED
```

```
                                .SBTTL  DMTALL -- Dismount all mounted devices for job
                        ;-------------------------------------------------------------------------------
                        ;  Dismount all devices for this job.
                        ;
006650                  DMTALL:
                        ;
                        ;  Gain exclusive access to USR data base
                        ;
006650  004737  007770'         CALL    GETUSR          ;Get exclusive access to USR
                        ;
                        ;  Get flag bit that identifies our job in mount table entries
                        ;
006654  113703  000000G         MOVB    CORUSR,R3       ;Get our job index number
006660  005005                  CLR     R5              ;Say no entry to point to yet
006662  004737  015112'         CALL    CDJFLG          ;Get flag bit for our job
                        ;
                        ;  Locate each device this is mounted by our job
                        ;
006666  013705  000000G         MOV     CSHDEV,R5       ;Point to 1st cache device entry
006672  005765  000000G 1$:     TST     CD$DVU(R5)      ;Is this entry in use?
006676  001406                  BEQ     2$              ;Br if not
006700  010500                  MOV     R5,R0           ;Get address of entry
006702  060200                  ADD     R2,R0           ;Point to byte with our job flag bit
006704  130310                  BITB    R3,(R0)         ;Is this device mounted by our job?
006706  001402                  BEQ     2$              ;Br if not
006710  004737  006752'         CALL    DMTDEV          ;Dismount this entry
006714  062705  000000G 2$:     ADD     #CD$$SZ,R5      ;Point to next mount table entry
006720  020537  000000G         CMP     R5,CSHDVN       ;Checked all entries?
006724  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
                        ;
006726  000137  015172'         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.
                                ;
006732  010546                 DMTSUB: MOV     R5,-(SP)
                                ;
                                ;  Locate entry for device in cached-device table.
                                ;
006734  004737  014776'                 CALL    CSHTST          ;Locate entry for device in cached dev table
006740  103402                          BCS     9$              ;Br if device is not mounted
                                ;
                                ;  Device is mounted, dismount it.
                                ;
006742  004737  006752'                 CALL    DMTDEV          ;Dismount the device
                                ;
                                ;  Finished
                                ;
006746  012605                 9$:     MOV     (SP)+,R5
006750  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.
                                ;
006752  010246                  DMTDEV: MOV     R2,-(SP)
006754  010346                          MOV     R3,-(SP)
                                ;
                                ;  Reset mount flag for our job.
                                ;
006756  113703  0000000                 MOVB    CORUSR,R3       ;Get our job index number
006762  004737  015112'                 CALL    CDJFLG          ;GET MOUNT FLAG FOR OUR JOB
006766  140312                          BICB    R3,(R2)         ;RESET MOUNT FLAG FOR OUR JOB
                                ;
                                ;  See if any other users still have device mounted.
                                ;
006770  010503                          MOV     R5,R3           ;Point to bytes with mount flags
006772  062703  0000000                 ADD     #CD$JOB,R3
006776  012700  0000000                 MOV     #CD$$UB,R0      ;GET # BYTES WITH MOUNT FLAGS
007002  105723                  1$:     TSTB    (R3)+           ;ANY OTHER JOBS HAVE THIS DEVICE MOUNTED?
007004  001042                          BNE     9$              ;BR IF YES
007006  077003                          SOB     R0,1$
                                ;
                                ;  No other jobs have this device mounted.
                                ;  Check if this is the SY disk.  If so, don't dismount it.
                                ;
007010  005765  0000000                 TST     CD$BAS(R5)      ;IS THIS A LOGICAL DISK?
007014  001011                          BNE     2$              ;BR IF YES
007016  016500  0000000                 MOV     CD$DVU(R5),R0   ;GET DEVICE # AND UNIT #
007022  120037  0000000                 CMPB    R0,SYINDX       ;IS THIS SY DEVICE?
007026  001004                          BNE     2$              ;BR IF NOT
007030  000300                          SWAB    R0              ;GET UNIT # TO LOW-ORDER BYTE
007032  120037  0000010                 CMPB    R0,SYUNIT+1     ;IS THIS SY UNIT?
007036  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.
                                ;
007040  005737  0000000          2$:     TST     CSHALC          ;Is data caching genned into system?
007044  001404                          BEQ     5$              ;Br if not
007046  016503  0000000                 MOV     CD$DVU(R5),R3   ;Get device and unit number for CSHCLN
007052  004777  0000000                 CALL    @CSHCLN         ;Clean out the data cache
                                ;
                                ;  Say device is no longer mounted
                                ;
007056  005065  0000000          5$:     CLR     CD$DVU(R5)      ;SAY DEVICE IS NOT MOUNTED
                                ;
                                ;  Remove file entries for this device from directory cache.
                                ;
007062  013700  0000000                 MOV     CSHHD,R0        ;POINT TO FIRST CACHED DIRECTORY ENTRY
007066  020560  0000000          3$:     CMP     R5,FC$CDX(R0)   ;IS FILE ON THIS DEVICE?
007072  001004                          BNE     4$              ;BR IF NOT
007074  005060  0000000                 CLR     FC$CDX(R0)      ;REMOVE ENTRY FROM CACHED DIRECTORY
007100  005060  0000000                 CLR     FD$NAM(R0)
007104  016000  0000000          4$:     MOV     FC$LNK(R0),R0   ;CHAIN TO NEXT ENTRY
```

```
     58 007110  001366                          BNE     3$              ;BR IF THERE IS ANOTHER
     59                                ;
     60                                ;  Finished
     61                                ;
     62 007112  012603                 9$:     MOV     (SP)+,R3
     63 007114  012602                         MOV     (SP)+,R2
     64 007116  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 #.
                ;
007120  010146  MNTCOM: MOV     R1,-(SP)
007122  010446          MOV     R4,-(SP)
                ;
                ;  Gain exclusive access to USR data base.
                ;
007124  004737  007770'         CALL    GETUSR          ;GET EXCLUSIVE USE OF USR
                ;
                ;  Move file spec to FILSPC and perform any assigns.
                ;
007130  013701  000002G         MOV     EMTBLK+2,R1     ;POINT TO DEVICE SPEC ARG
007134  004737  007532'         CALL    GETSPC          ;SET UP FILSPC
                ;
                ;  Check to see if device is legal.
                ;
007140  004737  010212'         CALL    CHKDEV          ;LOOK UP THE DEVICE NAME
007144  103006                  BCC     1$              ;BR IF DEVICE IS OK
007146  004737  015214'         CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
007152  012700  177776          MOV     #-2,R0          ;INVALID DEVICE NAME
007156  000137  015156'         JMP     USRERR
                ;
                ;  Build FILDVU word.
                ;
007162  110437  002014'  1$:     MOVB    R4,FILDVU       ;SET DEVICE INDEX #
007166  110037  002015'         MOVB    R0,FILDVU+1     ;SET UNIT #
                ;
                ;  Finished
                ;
007172  012604          MOV     (SP)+,R4
007174  012601          MOV     (SP)+,R1
007176  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.
                                ;
007200  010246                  CPYMNT: MOV     R2,-(SP)
007202  010346                          MOV     R3,-(SP)
007204  010446                          MOV     R4,-(SP)
007206  010546                          MOV     R5,-(SP)
007210  010204                          MOV     R2,R4           ;Save job index in R4
                                ;
                                ;  Begin to search through mount table for mounts for other job
                                ;
007212  013705  0000000         1$:     MOV     CSHDEV,R5       ;Point to start of mount table
007216  010403                          MOV     R4,R3           ;Get # of job we are copying from
007220  004737  015112'                  CALL    CDJFLG          ;Get job flag for other job
007224  130312                          BITB    R3,(R2)         ;Is this entry mounted by other job?
007226  001405                          BEQ     2$              ;Br if not
007230  113703  0000000                 MOVB    CORUSR,R3       ;Get our job index #
007234  004737  015112'                  CALL    CDJFLG          ;Get mount bit for our job
007240  150312                          BISB    R3,(R2)         ;Say device is mounted by our job
007242  062705  0000000         2$:     ADD     #CD$$SZ,R5      ;Point to next mount table entry
007246  020537  0000000                 CMP     R5,CSHDVN       ;Checked all entries?
007252  103761                          BLO     1$              ;Br if not
                                ;
                                ;  Finished
                                ;
007254  012605                          MOV     (SP)+,R5
007256  012604                          MOV     (SP)+,R4
007260  012603                          MOV     (SP)+,R3
007262  012602                          MOV     (SP)+,R2
007264  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
                                ;
007266  010546                  CLRDIR: MOV     R5,-(SP)
007270  013746  002014'                 MOV     FILDVU,-(SP)    ;Save current device/file index info
                                ;
                                ;  Set up device and unit numbers in FILDVU
                                ;
007274  110237  002014'                 MOVB    R2,FILDVU       ;Set device index number
007300  116300  0000000                 MOVB    C.DEVQ(R3),R0   ;Get device unit number
007304  042700  177770                  BIC     #^C<7>,R0       ;Clear out all but unit number
007310  110037  002015'                 MOVB    R0,FILDVU+1     ;Set unit number
                                ;
                                ;  See if this device is being cached
                                ;
007314  004737  014776'                 CALL    CSHTST          ;Is this device being cached?
007320  103414                          BCS     9$              ;Br if not
                                ;
                                ;  Clean out directory cache entries for this device
                                ;
007322  013700  0000000                 MOV     CSHHD,R0        ;Point to 1st cached dir entry
007326  020560  0000000         1$:     CMP     R5,FC$CDX(R0)   ;Is file on this device?
007332  001004                          BNE     2$              ;Br if not
007334  005060  0000000                 CLR     FC$CDX(R0)      ;Remove entry from directory cache
007340  005060  0000000                 CLR     FD$NAM(R0)
007344  016000  0000000         2$:     MOV     FC$LNK(R0),R0   ;Chain to next cache entry
007350  001366                          BNE     1$              ;Loop if more to check
                                ;
                                ;  Finished
                                ;
007352  012637  002014'         9$:     MOV     (SP)+,FILDVU
007356  012605                          MOV     (SP)+,R5
007360  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.
                                ;
007362  010146                  USRCOM: MOV     R1,-(SP)
007364  013703  0000000                 MOV     CHNADR,R3       ;GET ADDRESS OF CURRENT CHANNEL AREA
                                ;
                                ;  Claim USR data base for us
                                ;
007370  004737  007770'                 CALL    GETUSR          ;GAIN EXCLUSIVE ACCESS TO USR DATA BASE
                                ;
                                ;  Make sure channel is closed now.
                                ;
007374  032763  0000000 0000000         BIT     #CS$OPN,C.CSW(R3);IS CHANNEL OPEN NOW?
007402  001403                          BEQ     2$              ;BR IF NOT OPEN NOW
007404  005000                          CLR     R0              ;RETURN ERROR CODE OF 0
007406  000137  015156'                 JMP     USRERR
                                ;
                                ;  Move file spec to FILSPC and apply any assigns.
                                ;
007412  013701  0000020         2$:     MOV     EMTBLK+2,R1     ;GET ADDRESS OF FILE SPEC IN USER'S AREA
007416  042701  000001                  BIC     #1,R1           ;MAKE SURE ADDRESS IS EVEN
007422  004737  007532'                 CALL    GETSPC          ;GET FILE SPEC TO FILSPC
                                ;
                                ;  Now check to see if the device is legal.
                                ;
007426  004737  010212'                 CALL    CHKDEV          ;IS THE DEVICE LEGAL?
007432  103006                          BCC     1$              ;BR IF RECOGNIZED DEVICE
007434  004737  015214'                 CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
007440  012700  177776                  MOV     #-2,R0          ;INVALID DEVICE
007444  000137  015156'                 JMP     USRERR
                                ;  (At this point R4 has device table index and R0 has device unit #)
                                ;  Set up FILDVU word.
007450  110437  002014'         1$:     MOVB    R4,FILDVU       ;SET DEVICE #
007454  110037  002015'                 MOVB    R0,FILDVU+1     ;SET UNIT #
                                ;
```

```
                                        ;  Set up information in channel.
                                        ;
007460  110063  0000000                       MOVB    R0,C.DEVQ(R3)   ;SET DEVICE UNIT #
007464  010463  0000000                       MOV     R4,C.CSW(R3)    ;SET DEVICE TABLE INDEX # IN CSW
007470  052763  0000000 0000000               BIS     #CS$OPN,C.CSW(R3);MARK CHANNEL AS OPEN
007476  005063  0000000                       CLR     C.SBLK(R3)      ;STARTING BLOCK # OF FILE
007502  005063  0000000                       CLR     C.LENG(R3)      ;ALLOCATED LENGTH OF FILE
007506  005063  0000000                       CLR     C.USED(R3)      ;NO BLOCKS WRITTEN TO FILE YET
007512  105063  0000000                       CLRB    C.NUMQ(R3)      ;NO I/O OPERATIONS QUEUED ON CHANNEL YET
                                        ;
                                        ;  See if this device is allocated to some other user
                                        ;
007516  004737  010432'                       CALL    CHKALC          ;Check for allocation problems
                                        ;
                                        ;  See if we are authorized to access this device and file.
                                        ;
007522  004737  010566'                       CALL    CHKACC          ;CHECK FOR ACCESS AUTHORIZATION
                                        ;
                                        ;  Return successfully
                                        ;
007526  012601                              MOV     (SP)+,R1
007530  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)
                                ;
007532  010146                  GETSPC: MOV     R1,-(SP)
007534  010246                          MOV     R2,-(SP)
007536  005037  002044'                 CLR     ASNSIZ          ;NO ASSIGN SIZE YET
                                ;
                                ;  Copy the file specification from the user's area into FILSPC.
                                ;
007542  012702  000000G                 MOV     #FILSPC,R2      ;Copy spec into FILSPC
007546  004737  007704'                 CALL    CPYSPC          ;Copy file spec from user's area
                                ;
                                ;  Apply any assignments to this file spec.
                                ;
007552  013700  000000G                 MOV     FILSPC,R0       ;GET DEVICE NAME
007556  001447                          BEQ     4$              ;BR IF NULL NAME
007560  032737  000000G 000000G         BIT     #AF$BYA,RUNFLG  ;Should we bypass logical assignment?
007566  001043                          BNE     4$              ;Br if yes
007570  012702  000000G                 MOV     #ASNTBL,R2      ;POINT TO START OF ASSIGN TABLE
007574  020062  000000G         2$:     CMP     R0,AT$LOG(R2)   ;SEARCH FOR DEVICE NAME IN ASSIGN TABLE
007600  001406                          BEQ     3$              ;BR IF FOUND
007602  062702  000000G                 ADD     #AT$$SZ,R2      ;TRY NEXT ENTRY
007606  020227  000000G                 CMP     R2,#ASNEND      ;DONE ALL?
007612  103770                          BLO     2$              ;BR IF NOT
007614  000430                          BR      4$              ;NAME WAS NOT ASSIGNED
                                ;
                                ;  Name was assigned.  Substitute assigned name.
                                ;
007616  012701  000000G         3$:     MOV     #FILSPC,R1      ;POINT TO AREA WITH FILE SPEC
007622  016211  000000G                 MOV     AT$DEV(R2),(R1) ;PUT IN REAL DEVICE NAME
007626  016200  000000G                 MOV     AT$FIL(R2),R0   ;WAS FILE NAME ASSIGNED?
007632  001421                          BEQ     4$              ;BR IF NOT
007634  010061  000002                  MOV     R0,2(R1)        ;PUT IN FILE NAME
007640  016261  000020 000004           MOV     AT$FIL+2(R2),4(R1)
007646  016200  000000G                 MOV     AT$EXT(R2),R0   ;WAS FILE EXTENSION ASSIGNED?
007652  001402                          BEQ     8$              ;BR IF NOT
007654  010061  000006                  MOV     R0,6(R1)        ;PUT IN FILE EXT
007660  016200  000000G         8$:     MOV     AT$SIZ(R2),R0   ;WAS FILE SIZE ASSIGNED?
007664  001404                          BEQ     4$              ;BR IF NOT
007666  010061  000010                  MOV     R0,8.(R1)       ;PUT IN FILE SIZE
007672  010037  002044'                 MOV     R0,ASNSIZ       ;RETURN ASSIGN SIZE IN ASNSIZ
                                ;
                                ;  finished
                                ;
007676  012602                  4$:     MOV     (SP)+,R2
007700  012601                          MOV     (SP)+,R1
007702  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.
                                ;
007704                          CPYSPC:
                                ;
                                ;  Make sure address of file spec is legal
                                ;
007704  010100                          MOV     R1,R0           ;GET ADDRESS OF FILE SPEC BUFFER
007706  004737  0000000                 CALL    UACHKW          ;MAKE SURE ADDRESS IS LEGAL
007712  103004                          BCC     5$              ;BR IF ADDRESS IS OK
007714  012700  177766                  MOV     #-12,R0         ;GET ERROR ABORT CODE
007720  000137  015144'                 JMP     USRCLS          ;ABORT THE JOB
                                ;
                                ;  Copy file specification from user's area to FILSPC.
                                ;
007724  032737  0000000 0000000 5$:     BIT     #UMODE,EMTPS    ;WAS EMT DONE IN USER OR KERNEL MODE?
007732  001410                          BEQ     6$              ;BR IF KERNEL MODE
007734  013700  0000000                 MOV     ODTBAS,R0       ;GET THE USER'S HIGH MEMORY LIMIT
007740  160100                          SUB     R1,R0           ;SUBTRACT ADDRESS OF FILE SPEC AREA
007742  006200                          ASR     R0              ;CONVERT TO NUMBER OF WORDS
007744  001410                          BEQ     9$              ;BR IF NOTHING TO MOVE
007746  020027  000004                  CMP     R0,#4.          ;COMPARE WITH SIZE OF FULL FILE SPEC
007752  101402                          BLOS    1$              ;BR IF WE MUST MOVE LESS BECAUSE OF MEMORY TOP
007754  012700  000004          6$:     MOV     #4.,R0          ;# WORDS TO MOVE
007760  106521                  1$:     MFPD    (R1)+           ;GET A WORD FROM USER'S AREA
007762  012622                          MOV     (SP)+,(R2)+     ;MOVE TO FILSPC
007764  077003                          SOB     R0,1$
                                ;
                                ;  Finished
                                ;
007766  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.
                                ;
007770  010546                  GETUSR: MOV     R5,-(SP)
                                ;
                                ;  See if we can access the USR
                                ;
007772                          4$:     DISABL                  ;;;DISABLE INTERRUPTS
010000  113700  0000000G               MOVB    USRJOB,R0       ;;;IS USR AVAILABLE NOW?
010004  001426                         BEQ     1$              ;;;BR IF YES
010006  120037  0000000G               CMPB    R0,CORUSR       ;;;DO WE ALREADY OWN USR?
010012  001426                         BEQ     2$              ;;;BR IF YES
                                ;
                                ;  Someone else owns the USR now.
                                ;  Suspend our job until we can get it.
                                ;
010014                                  ENABL                   ;ENABLE INTERRUPTS
010022  113705  0000000G               MOVB    EXCJOB,R5       ;REMEMBER IF WE HAVE EXCLUSIVE SYS USE
010026  105037  0000000G               CLRB    EXCJOB          ;ENABLE OTHER JOBS TO RUN
010032  012700  0000000G               MOV     #S$QUSR,R0      ;PUT US IN QUEUE FOR USR
010036  004737  0000000G               CALL    QNSPNX          ;SUSPEND JOB AND WAIT FOR USR
010042  004737  0000000G               CALL    CHKABT          ;WERE WE ABORTED WHILE ASLEEP?
010046  020537  0000000G               CMP     R5,CORUSR       ;DO WE WANT EXCLUSIVE ACCESS TO SYSTEM?
010052  001347                         BNE     4$              ;LOOP IF NOT
010054  110537  0000000G               MOVB    R5,EXCJOB       ;GET EXCLUSIVE ACCESS TO THE SYSTEM
010060  000744                         BR      4$              ;GO TRY TO GET USR NOW
                                ;
                                ;  We can get USR now.
                                ;
010062  113737  0000000G 0000000G 1$:      MOVB    CORUSR,USRJOB   ;;;CLAIM USR FOR US
010070                          2$:     ENABL                   ;ENABLE INTERRUPTS
010076  005237  002026'                 INC     USRCNT          ;REMEMBER # TIMES WE CLAIMED USR
                                ;
                                ;  Make sure USR is not being called from a completion routine.
                                ;
010102  105737  0000000G               TSTB    CURCP           ;ARE WE IN A COMPLETION ROUTINE NOW?
010106  001404                         BEQ     3$              ;BR IF NOT
010110  012700  177765                 MOV     #-13,R0         ;ABORT IF IN COMPLETION ROUTINE
010114  000137  015156'                 JMP     USRERR
                                ;
                                ;  Initialize USR data base.
                                ;
010120  012737  177777  002002' 3$:     MOV     #-1,CURSEG      ;NO DIR SEG IN BUFFER NOW
                                ;
                                ;  Finished
                                ;
010126  012605                          MOV     (SP)+,R5
010130  000207                          RETURN

                                        .SBTTL  FREUSR -- Free the USR
                                ;-------------------------------------------------------------------
                                ;  FREUSR is called to release our ownership of the USR data base.
                                ;
010132                          FREUSR: DISABL                  ;;;DISABLE INTERRUPTS
```

```
     58 010140  123737  0000000 0000000          CMPB    CORUSR,USRJOB   ;;;DO WE OWN USR?
     59 010146  001015                           BNE     1$              ;;;BR IF NOT
     60 010150  005337  002026'                  DEC     USRCNT          ;;;IS THIS LAST UNLOCK OF USR?
     61 010154  002012                           BGE     1$              ;;;BR IF NOT
     62 010156  105037  0000000                  CLRB    USRJOB          ;;;SAY USR IS FREE
     63                                 ;
     64                                 ;  Restart any jobs waiting for USR
     65                                 ;
     66 010162                                   ENABL                   ;ENABLE INTERRUPTS
     67 010170  012700  0000000                  MOV     #S$QUSR,R0      ;QUEUE OF WAITING JOBS
     68 010174  004737  0000000                  CALL    UREGO           ;RESTART WAITING JOBS
     69 010200  000207                           RETURN
     70                                 ;
     71                                 ;  Finished
     72                                 ;
     73 010202                          1$:      ENABL                   ;ENABLE INTERRUPTS
     74 010210  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.
                ;
010212  010146  CHKDEV: MOV     R1,-(SP)
010214  010246         MOV     R2,-(SP)
                ;
                ;  Get device name and split off unit number.
                ;
010216  013701  0000000      MOV     FILSPC,R1       ;GET DEVICE NAME (RAD50)
010222  005000               CLR     R0              ;SET FOR DIVIDE
010224  071027  000050       DIV     #50,R0          ;SPLIT OFF LOW-ORDER RAD50 CHARACTER
010230  012702  177777       MOV     #-1,R2          ;ASSUME NO UNIT NUMBER SPECIFIED
010234  005701               TST     R1              ;WAS A UNIT NUMBER SPECIFIED?
010236  001406               BEQ     6$              ;BR IF NOT
010240  162701  000036       SUB     #36,R1          ;CONVERT RAD50 DIGIT TO BINARY VALUE
010244  010102               MOV     R1,R2           ;GET BINARY VALUE OF UNIT NUMBER
010246  020227  000007       CMP     R2,#7           ;RESTRICT UNIT NUMBER TO RANGE 0-7
010252  101063               BHI     8$              ;BR IF INVALID UNIT NUMBER
010254  070027  000050  6$:  MUL     #50,R0          ;GET DEVICE NAME WITHOUT UNIT NUMBER
                ;
                ;  The rad50 device name less unit number is now in R1.
                ;  R2 has the binary value of the unit number or -1 if no unit number
                ;  was specified.
                ;
                ;  Translate "SY:" and "DK:" to physical device.
                ;
010260  020137  002020'       CMP     R1,R50DK        ;IS DEVICE NAME "DK"?
010264  001403               BEQ     2$              ;BR IF YES
010266  020137  002016'       CMP     R1,R50SY        ;IS DEVICE NAME "SY"?
010272  001007               BNE     3$              ;BR IF NOT
010274  013704  0000000  2$:  MOV     SYINDX,R4       ;GET SY DEVICE INDEX NUMBER
010300  005702               TST     R2              ;DID USER SPECIFY A UNIT NUMBER?
010302  002014               BGE     7$              ;BR IF YES
010304  113702  0000010      MOVB    SYUNIT+1,R2     ;GET SYSTEM DEVICE UNIT NUMBER
010310  000411               BR      7$
                ;
                ;  Look up device name in permanent device name table.
                ;
010312  013704  0000000  3$:  MOV     NUMDEV,R4       ;GET INDEX NUMBER OF LAST DEVICE
010316  020164  0000000  5$:  CMP     R1,PNAME(R4)    ;SEARCH FOR NAME IN TABLE
010322  001404               BEQ     7$              ;BR IF FOUND
010324  162704  000002       SUB     #2,R4           ;TRY NEXT ENTRY
010330  002372               BGE     5$              ;LOOP IF MORE TO CHECK
010332  000433               BR      8$              ;Invalid device
                ;
                ;  Found device name.  Translate no unit number into # 0.
                ;
```

```
     58 010334  010200                7$:     MOV     R2,R0           ;GET UNIT NUMBER VALUE
     59 010336  002001                        BGE     1$              ;BR IF UNIT NUMBER WAS SPECIFIED
     60 010340  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 010342  020437  0000000       1$:     CMP     R4,LDDEVX       ;IS THIS A LOGICAL DISK?
     66 010346  001006                        BNE     11$             ;BR IF NOT
     67 010350  010002                        MOV     R0,R2           ;GET UNIT NUMBER
     68 010352  006302                        ASL     R2              ;CONVERT TO WORD TABLE INDEX
     69 010354  005762  0000000               TST     LDPDEV(R2)      ;IS UNIT MAPPED TO A FILE?
     70 010360  001420                        BEQ     8$              ;BR IF NOT
     71 010362  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 010364  010002                11$:    MOV     R0,R2           ;Get unit number
     77 010366  020437  0000000               CMP     R4,CLDEVX       ;Is this a CL unit?
     78 010372  001405                        BEQ     12$             ;Br if yes
     79 010374  020437  0000000               CMP     R4,C1DEVX       ;Is this a C1 unit?
     80 010400  001006                        BNE     9$              ;Br if not
     81 010402  062702  000010                ADD     #8.,R2          ;Bias unit number by 8 for C1
     82 010406  006302                12$:    ASL     R2              ;Convert to word table index
     83 010410  005762  0000000               TST     CL$LIX(R2)      ;Is this unit assigned to some line?
     84 010414  001402                        BEQ     8$              ;Br if not
     85                                ;
     86                                ;  This device access is ok
     87                                ;
     88 010416  000241                9$:     CLC                     ;SIGNAL GOOD RETURN
     89 010420  000401                        BR      10$
     90                                ;
     91                                ;  Invalid device
     92                                ;
     93 010422  000261                8$:     SEC                     ;SIGNAL INVALID DEVICE NAME
     94                                ;
     95                                ;  Finished -- Return
     96                                ;
     97 010424  012602                10$:    MOV     (SP)+,R2
     98 010426  012601                        MOV     (SP)+,R1
     99 010430  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.
                        ;
010432  010046         CHKALC: MOV     R0,-(SP)
010434  010246                 MOV     R2,-(SP)
                        ;
                        ;  Don't do allocation test for TT
                        ;
010436  105737  002014'         TSTB    FILDVU          ;Device # 0 ==> TT
010442  001446                 BEQ     9$              ;Br if this device is TT
                        ;
                        ;  See if this device is in the allocation table.
                        ;
010444  012702  000000G        MOV     #ALCTBL,R2      ;Point to start of allocation table
010450  013700  002014'         MOV     FILDVU,R0       ;Get device/unit id
010454  020062  000000G  1$:    CMP     R0,AD$DVU(R2)   ;Is this entry for this device?
010460  001421                 BEQ     2$              ;Br if yes
010462  062702  000000G        ADD     #AD$$SZ,R2      ;Point to next table entry
010466  020227  000000G        CMP     R2,#ALCEND      ;Have we checked all entries?
010472  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.
                        ;
010474  113700  002014'         MOVB    FILDVU,R0       ;Get the device index number
010500  032760  000000G 000000G BIT     #DX$RAL,DVFLAG(R0);Is allocation required before access?
010506  001424                 BEQ     9$              ;Br if not
010510  004737  015214'         CALL    ERRNAM          ;Set file spec for Kmon error message
010514  012700  177751         MOV     #-27,R0         ;Set error code
010520  000137  015144'         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.
                        ;
010524  116202  000000G  2$:    MOVB    AD$JOB(R2),R2   ;Get # of job that owns the device
010530  113700  000000G        MOVB    CORUSR,R0       ;Get current job index number
010534  126260  000000G 000000G CMPB    LNPRIM(R2),LNPRIM(R0) ;Do we own the device?
010542  001406                 BEQ     9$              ;Br if yes
010544  004737  015214'         CALL    ERRNAM          ;Set file spec for Kmon error message
010550  012700  177752         MOV     #-26,R0         ;Set error code
010554  000137  015144'         JMP     USRCLS          ;Error return
                        ;
                        ;  Device access is ok
                        ;
010560  012602           9$:    MOV     (SP)+,R2
010562  012600                 MOV     (SP)+,R0
010564  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$RON set in channel CSW if read-only access authorized.
                                ;
010566  010046                 CHKACC: MOV     R0,-(SP)
010570  010146                         MOV     R1,-(SP)
010572  010246                         MOV     R2,-(SP)
010574  010346                         MOV     R3,-(SP)
010576  010446                         MOV     R4,-(SP)
010600  010546                         MOV     R5,-(SP)
010602  113701  0000000                MOVB    CORUSR,R1       ;GET JOB INDEX NUMBER
                                ;
                                ;  See if we are accessing a logical disk with NOWRITE attribute set.
                                ;
010606  020437  0000000                CMP     R4,LDDEVX       ;IS THIS A LOGICAL DISK?
010612  001011                         BNE     13$             ;BR IF NOT
010614  010002                         MOV     R0,R2           ;GET UNIT NUMBER
010616  006302                         ASL     R2              ;CONVERT TO WORD TABLE INDEX
010620  032762  0000000 0000000        BIT     #LD$RON,LDFLAG(R2) ;WAS /NOWRITE SPECIFIED FOR DISK?
010626  001403                         BEQ     13$             ;BR IF NOT
010630  052763  0000000 0000000        BIS     #CS$RON,C.CSW(R3);SAY READ-ONLY ACCESS ALLOWED
                                ;
                                ;  Allow access to all devices if user has BYPASS privilege
                                ;
010636  032737  0000000 0000000 13$:   BIT     #PO$BYP,PRIVCO  ;Does user have BYPASS privilege?
010644  001004                         BNE     19$             ;Br if yes
                                ;
                                ;  See if TSXUCL is trying to access its command file
                                ;
010646  032761  0000000 0000000        BIT     #$UCLRN,LSW7(R1);IS TSXUCL RUNNING?
010654  001402                         BEQ     16$             ;Br if not
010656  000137  011224'         19$:   JMP     9$              ;Allow full file access
                                ;
                                ;  See if a non-privileged user is trying to access a SYS or TSX file on SY:
                                ;
010662  032737  0000000 0000000 16$:   BIT     #PO$SYS,PRIVCO  ;IS THIS A PRIVILEGED USER?
010670  001022                         BNE     1$              ;BR IF PRIVILEGED USER
010672  032761  0000000 0000000        BIT     #$NOIN,LSW3(R1) ;ARE WE RUNNING IN START-UP COMMAND FILE?
010700  001016                         BNE     1$              ;BR IF YES -- GIVE FULL ACCESS DURING STARTUP
010702  120437  0000000                CMPB    R4,SYINDX       ;Is device = SY?
010706  001013                         BNE     1$              ;Br if not SY
010710  120037  0000010                CMPB    R0,SYUNIT+1     ;Is unit = SY?
010714  001010                         BNE     1$              ;Br if not
010716  013702  0000060                MOV     FILSPC+6,R2     ;GET FILE EXTENSION
010722  020237  002022'                CMP     R2,R50SYS       ;IS EXTENSION "SYS"?
010726  001525                         BEQ     8$              ;BR IF YES -- DISALLOW ACCESS
```

```
 58 010730  020237  002024'                    CMP     R2,R50TSX       ;IS EXTENSION "TSX"?
 59 010734  001522                             BEQ     8$              ;BR IF YES -- DISALLOW ACCESS
 60                                 ;
 61                                 ;  See if this is a non-file-structured lookup to a file-structured device
 62                                 ;
 63 010736  005737  000002@        1$:     TST     FILSPC+2        ;Non-file-structured lookup?
 64 010742  001017                         BNE     15$             ;Br if not
 65 010744  032764  000000@ 000000@        BIT     #DS$DIR,DVSTAT(R4) ;Is this a directory structured device?
 66 010752  001413                         BEQ     15$             ;Br if not
 67 010754  032737  000000@ 000000@        BIT     #PO$NFW,PRIVCO  ;May user do NFS open and then write?
 68 010762  001007                         BNE     15$             ;Br if he has write access
 69 010764  032737  000000@ 000000@        BIT     #PO$NFR,PRIVCO  ;May user do NFS open and read?
 70 010772  001503                         BEQ     8$              ;Br if not -- No access allowed
 71 010774  052763  000000@ 000000@        BIS     #CS$RON,C.CSW(R3) ;Force read-only access
 72                                 ;
 73                                 ;  See if ACCESS command was used to restrict access to any devices or files.
 74                                 ;
 75 011002  005737  000000@        15$:    TST     RESDEV          ;ANY RESTRICTED DEVICES AND FILES?
 76 011006  001506                         BEQ     9$              ;BR IF NOT
 77                                 ;
 78                                 ;  There are restricted devices and files.
 79                                 ;  See if access is authorized.
 80                                 ;
 81 011010  110037  002047'                MOVB    R0,CKAUNT       ;SAVE DEVICE UNIT #
 82 011014  110437  002046'                MOVB    R4,CKADEV       ;SAVE DEVICE INDEX NUMBER
 83 011020  005002                         CLR     R2              ;SET NO-ACCESS STATE
 84 011022  012705  000000@                MOV     #OKFILE,R5      ;POINT TO START OF ALLOWED DEVICE SPECS
 85                                 ;  Check next authorized file spec.
 86 011026  010504                  6$:    MOV     R5,R4           ;GET ADDRESS OF FILE SPEC AREA
 87                                 ;  Check device number and unit number.
 88 011030  116400  000000@                MOVB    OF$DEV(R4),R0   ;GET DEV INDEX NUMBER
 89 011034  001432                         BEQ     4$              ;BR IF THIS ENTRY IS NOT USED
 90 011036  002407                         BLT     2$              ;BR IF WILDCARD
 91 011040  120037  002046'                CMPB    R0,CKADEV       ;DOES IT MATCH THAT OF FILE SPEC BEING OPENED?
 92 011044  001026                         BNE     4$              ;BR IF NOT
 93 011046  123764  002047' 000000@       CMPB    CKAUNT,OF$UNT(R4);COMPARE DEVICE UNIT NUMBERS
 94 011054  001022                         BNE     4$              ;BR IF MISMATCH
 95                                 ;  Check file names.
 96 011056  062704  000000@        2$:     ADD     #OF$FIL,R4      ;POINT TO FILE NAME
 97 011062  012701  000002@                MOV     #FILSPC+2,R1    ;POINT TO FILE NAME IN SPEC BEING OPENED
 98 011066  012700  000003                 MOV     #3,R0           ;GET # WORDS TO COMPARE
 99 011072  022421                  5$:    CMP     (R4)+,(R1)+     ;COMPARE FILE NAMES
100 011074  001404                         BEQ     3$              ;BR IF OK
101 011076  026427  177776  000000@        CMP     -2(R4),#WLDNAM  ;WILD CARD IN NAME?
102 011104  001006                         BNE     4$              ;BR IF NOT
103 011106  077007                  3$:    SOB     R0,5$           ;KEEP COMPARING
104                                ;  Names matched. -- See if user is restricted to read-only access.
105 011110  032765  000000@ 000000@        BIT     #OT$RON,OF$FLG(R5);IS USER RESTRICTED TO READ-ONLY ACCESS?
106 011116  001442                         BEQ     9$              ;BR IF NOT
107                                ;  Read-only access -- Remember that and keep checking for full access.
108 011120  005202                         INC     R2              ;SET READ-ONLY ACCESS STATE
109                                ;  Check next authorized file spec.
110 011122  062705  000000@        4$:     ADD     #OF$$SZ,R5      ;POINT TO NEXT AUTHORIZED FILE SPEC
111 011126  020527  000000@                CMP     R5,#OKFEND      ;CHECKED ALL?
112 011132  103735                         BLO     6$              ;BR IF NOT
113                                ;
114                                ;  Reached end of authorized file spec list without getting full access.
```

```
                                        ;
                                        ;  Allow implicit access to some devices and files.
                                        ;  Allow full access to TT.
                                        ;
011134  105737  002046'                       TSTB    CKADEV          ;TT DEVICE INDEX IS ALWAYS ZERO
011140  001431                               BEQ     9$              ;BR IF TT
                                        ;
                                        ;  See if we were granted read-only access.
                                        ;
011142  005702                               TST     R2              ;WERE WE GRANTED READ-ONLY ACCESS TO FILE?
011144  001024                               BNE     7$              ;BR IF YES
                                        ;
                                        ;  Allow DUP to access SY: on a read-only basis
                                        ;
011146  113701  000000G                      MOVB    CORUSR,R1       ;GET JOB INDEX NUMBER
011152  032761  000000G 000000G              BIT     #$DUPRN,LSW6(R1);IS DUP RUNNING?
011160  001410                               BEQ     8$              ;BR IF NOT
011162  123737  002046' 000000G              CMPB    CKADEV,SYINDX   ;IS DEVICE = SY?
011170  001004                               BNE     8$              ;BR IF NOT
011172  123737  002047' 000001G              CMPB    CKAUNT,SYUNIT+1 ;IS UNIT = SY?
011200  001406                               BEQ     7$              ;BR IF SY -- ALLOW READ ONLY ACCESS
                                        ;
                                        ;  User is not allowed to access this device or file.
                                        ;
011202  004737  015214'                 8$:   CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
011206  012700  177764                       MOV     #-14,R0         ;SET ERROR CODE
011212  000137  015144'                       JMP     USRCLS          ;ABORT
                                        ;
                                        ;  User is allowed read-only access to this device or file.
                                        ;  Set CS$RON flag in CSW.
                                        ;
011216  052763  000000G 000000G        7$:   BIS     #CS$RON,C.CSW(R3);SET READ-ONLY ACCESS FLAG IN CSW
                                        ;
                                        ;  User is allowed full access to file.
                                        ;
011224  012605                         9$:   MOV     (SP)+,R5
011226  012604                               MOV     (SP)+,R4
011230  012603                               MOV     (SP)+,R3
011232  012602                               MOV     (SP)+,R2
011234  012601                               MOV     (SP)+,R1
011236  012600                               MOV     (SP)+,R0
011240  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.
                        ;
011242  010246          FNDFIL: MOV     R2,-(SP)
011244  010346                  MOV     R3,-(SP)
                        ;
                        ;  Read in directory segment number 1 and set up info about directory.
                        ;
011246  004737  013454'         CALL    RDSEG1          ;READ IN FIRST DIRECTORY SEGMENT
                        ;
                        ;  Search for permanent file entry that matches our name.
                        ;
011252  012700  0000000 2$:     MOV     #FS$PRM,R0      ;LOCATE NEXT PERM FILE ENTRY
011256  004737  013166'         CALL    @ETDIR
011262  103415                  BCS     3$              ;BR IF HIT END OF DIRECTORY
                        ;  Compare file names.
011264  010103                  MOV     R1,R3           ;GET ADDRESS OF DIRECTORY ENTRY
011266  062703  0000006         ADD     #FD$NAM,R3      ;POINT TO FILE NAME IN DIR ENTRY
011272  012702  0000020         MOV     #FILSPC+2,R2    ;POINT TO NAME WE WANT TO FIND
011276  012700  000003          MOV     #3,R0           ;COMPARE 3 WORDS
011302  022223          1$:     CMP     (R2)+,(R3)+     ;DO NAMES MATCH?
011304  001362                  BNE     2$              ;BR IF NOT
011306  077003                  SOB     R0,1$           ;CHECK ALL OF NAME
                        ;
                        ;  Found the file entry -- Calculate starting block number.
                        ;
011310  004737  013420'         CALL    SBCALC          ;CALCULATE STARTING BLOCK # FOR ENTRY
                        ;
                        ;  Finished
                        ;
011314  000241          4$:     CLC                     ;SIGNAL GOOD RETURN
011316  012603          3$:     MOV     (SP)+,R3
011320  012602                  MOV     (SP)+,R2
011322  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.
                                ;
011324  010346                 FNDFRE: MOV     R3,-(SP)
011326  010446                         MOV     R4,-(SP)
                                ;
                                ;  Initialize tables that hold info about 2 largest free slots.
                                ;
011330  005037  002030'                 CLR     L1SIZ           ;SIZE OF LARGEST FREE SLOT
011334  005037  002036'                 CLR     L2SIZ           ;SIZE OF 2ND LARGEST FREE SLOT
                                ;
                                ;  Read in 1st directory segment and set up info about directory parameters.
                                ;
011340  004737  013454'                 CALL    RDSEG1          ;READ IN 1ST DIRECTORY SEGMENT
                                ;
                                ;  Consolidate entries in directory segment.
                                ;
011344  004737  012634'         7$:     CALL    CONSOL          ;CONSOLIDATE DIRECTORY SEGMENT
                                ;
                                ;  Find next directory entry for a permanent file or a free slot.
                                ;
011350  063701  002006'         4$:     ADD     DIRSIZ,R1       ;POINT TO NEXT DIRECTORY ENTRY
011354  016100  000000G                 MOV     FD$STA(R1),R0   ;PICK UP STATUS WORD FROM DIRECTORY ENTRY
011360  032700  000000C                 BIT     #FS$PRM+FS$EMP+FS$EOS,R0;PERM FILE, EMPTY SLOT OR END OF SEGMENT?
011364  001771                         BEQ     4$              ;BR IF NONE OF ABOVE
011366  032700  000000G                 BIT     #FS$EMP,R0      ;IS THIS AN EMPTY SPACE ENTRY?
011372  001024                         BNE     13$             ;BR IF YES
011374  032700  000000G                 BIT     #FS$EOS,R0      ;IS THIS THE END OF SEGMENT MARKER ENTRY?
011400  001106                         BNE     1$              ;BR IF YES
                                ;
                                ;  Found permanent file entry.  See if this is a protected file entry
                                ;  with the same name as the file being entered.
                                ;
011402  032700  000000G                 BIT     #FS$PRO,R0      ;IS THIS FILE PROTECTED?
011406  001760                         BEQ     4$              ;BR IF NOT
011410  010103                         MOV     R1,R3           ;POINT TO NAME IN DIR ENTRY
011412  062703  000000G                 ADD     #FD$NAM,R3
011416  012704  000020G                 MOV     #FILSPC+2,R4    ;POINT TO NAME OF FILE BEING ENTERED
011422  012700  000003                 MOV     #3,R0           ;GET # WORDS TO COMPARE
011426  022324                 14$:    CMP     (R3)+,(R4)+     ;COMPARE FILE NAMES
011430  001347                         BNE     4$              ;BR IF DIFFERENT
011432  077003                         SOB     R0,14$          ;COMPARE ALL OF NAMES
                                ;  Error: Protected file with same name.
                                ;  Return error code 3.
```

```
     58 011434  012700  000003                    MOV     #3,R0           ;SET ERROR CODE
     59 011440  000137  012012'                    JMP     24$             ;RETURN ERROR STATUS
     60                                 ;
     61                                 ;  We found a free slot.  Check its size.
     62                                 ;
     63 011444  016100  000000@         13$:     MOV     FD$LEN(R1),R0   ;GET SIZE OF FREE SLOT
     64 011450  005702                            TST     R2              ;WHAT KIND OF ALLOCATION IS BEING REQUESTED?
     65 011452  001422                            BEQ     2$              ;BR IF HE WANTS DEFAULT ALGORITHM
     66 011454  020227  177777                    CMP     R2,#-1          ;DOES HE WANT LARGEST FREE SLOT?
     67 011460  001417                            BEQ     2$              ;BR IF YES
     68                                 ;  See if free slot is large enough to satisfy specific request.
     69 011462  020002                            CMP     R0,R2           ;IS FREE SLOT LARGE ENOUGH?
     70 011464  103731                            BLO     4$              ;BR IF NOT
     71 011466  013703  002030'                    MOV     L1SIZ,R3        ;GET SIZE OF BEST FIT SO FAR
     72 011472  001402                            BEQ     15$             ;BR IF THIS IS 1ST SLOT THAT WILL FIT
     73 011474  020003                            CMP     R0,R3           ;IS NEW SLOT A BETTER FIT THAN LAST ONE?
     74 011476  103324                            BHIS    4$              ;BR IF NOT
     75 011500  010037  002030'         15$:     MOV     R0,L1SIZ        ;REMEMBER SIZE OF BEST FIT FOUND SO FAR
     76 011504  010137  002032'                    MOV     R1,L1OFF        ;OFFSET OF DIR ENTRY WITHIN SEGMENT
     77 011510  013737  002002' 002034'           MOV     CURSEG,L1SEG    ;# OF DIR SEGMENT WITH ENTRY
     78 011516  000714                            BR      4$
     79                                 ;  We have a requested size of 0 or -1.
     80                                 ;  Remember two largest free slots.
     81 011520  020037  002030'         2$:      CMP     R0,L1SIZ        ;IS NEW SLOT LARGER THAN LARGEST WE'VE SEEN?
     82 011524  101421                            BLOS    5$              ;BR IF NOT
     83 011526  013737  002030' 002036'           MOV     L1SIZ,L2SIZ     ;MOVE INFO INTO 2ND LARGEST ENTRY
     84 011534  013737  002034' 002042'           MOV     L1SEG,L2SEG
     85 011542  013737  002032' 002040'           MOV     L1OFF,L2OFF
     86 011550  010037  002030'                    MOV     R0,L1SIZ        ;SAVE SIZE OF NEW LARGEST ENTRY
     87 011554  010137  002032'                    MOV     R1,L1OFF        ;SAVE ADDRESS OF DIR ENTRY
     88 011560  013737  002002' 002034'           MOV     CURSEG,L1SEG    ;SAVE # OF SEGMENT WITH ENTRY
     89 011566  000670                            BR      4$              ;KEEP LOOKING FOR LARGER FREE SLOTS
     90 011570  020037  002036'         5$:      CMP     R0,L2SIZ        ;IS NEW SLOT LARGER THAN 2ND LARGEST SO FAR?
     91 011574  101665                            BLOS    4$              ;BR IF NOT
     92 011576  010037  002036'                    MOV     R0,L2SIZ        ;SAVE NEW 2ND LARGEST SIZE
     93 011602  010137  002040'                    MOV     R1,L2OFF        ;SAVE ADDRESS OF DIR ENTRY
     94 011606  013737  002002' 002042'           MOV     CURSEG,L2SEG    ;SAVE DIR SEGMENT #
     95 011614  000655                            BR      4$              ;KEEP LOOKING
     96                                 ;
     97                                 ;  There are no more free slots in the current directory segment.
     98                                 ;  See if there are more segments to check.
     99                                 ;
    100 011616  013700  000000C         1$:      MOV     USRBUF+DH$NXT,R0;GET # OF NEXT SEGMENT
    101 011622  001403                            BEQ     6$              ;BR IF THERE ARE NO MORE SEGMENTS
    102 011624  004737  013570'                    CALL    RDSEG           ;READ IN NEXT SEGMENT
    103 011630  000645                            BR      7$              ;SEARCH THIS SEGMENT
    104                                 ;
    105                                 ;  We have reached the end of the last active directory segment.
    106                                 ;  See if we were able to satisfy request.
    107                                 ;
    108 011632  005737  002030'         6$:      TST     L1SIZ           ;DID WE FIND ANY ENTRY THAT WOULD DO?
    109 011636  001463                            BEQ     9$              ;BR IF NOT
    110 011640  005702                            TST     R2              ;WHAT KIND OF REQUEST WAS IT?
    111 011642  001412                            BEQ     8$              ;DEFAULT ALLOCATION
    112                                 ;  User wants largest free slot or he specified a specific size.
    113 011644  020227  177777                    CMP     R2,#-1          ;DOES HE WANT LARGEST OR SPECIFIC SIZE?
    114 011650  001002                            BNE     10$             ;BR IF HE SPECIFIED EXACT AMT HE WANTS
```

```
                                        MOV     L1SIZ,R2        ;RETURN SIZE OF LARGEST FREE SLOT
                                10$:    MOV     L1SEG,R0        ;GET SEGMENT # WITH ENTRY
                                        MOV     L1OFF,R1        ;GET ADDRESS OF ENTRY
                                        BR      23$             ;GO REREAD SEGMENT WITH EMPTY ENTRY
                                ;  User wants default allocation.
                                ;  (i.e., Larger of (1/2 largest slot  or  2nd largest slot) )
                                8$:     MOV     L1SIZ,R2        ;GET SIZE OF LARGEST SLOT
                                        CMP     R2,#1           ;IS LARGEST SLOT 1 BLOCK LONG?
                                        BEQ     11$             ;IF SO THEN USE ALL OF IT
                                        CLC                     ;DIVIDE BY 2
                                        ROR     R2
                                        CMP     R2,L2SIZ        ;COMPARE TO 2ND LARGEST SLOT
                                        BHI     11$             ;USE 1/2 OF LARGEST
                                        MOV     L2SEG,R0        ;GET # OF SEGMENT WITH 2ND LARGEST FREE BLOCK
                                        MOV     L2OFF,R1        ;GET OFFSET TO ENTRY WITHIN SEGMENT
                                        MOV     L2SIZ,R2        ;GET SIZE OF ENTRY
                                        BR      12$
                                11$:    MOV     L1SEG,R0        ;GET # OF SEGMENT WITH LARGEST FREE BLOCK
                                        MOV     L1OFF,R1        ;GET OFFSET TO ENTRY WITHIN SEGMENT
                                ;
                                ;  Constrain the largest file size returned in response to a 0-size request
                                ;  to a sysgen supplied parameter (MAXFIL).
                                ;
                                12$:    TST     VMXFIL          ;Did user specifiy a size to constrain alloc?
                                        BEQ     23$             ;Br if not
                                        CMP     R2,VMXFIL       ;IS FREE SLOT LARGER THAN MAXFIL PARAMETER?
                                        BLOS    23$             ;BR IF NOT
                                        MOV     VMXFIL,R2       ;ONLY USE THIS MUCH SPACE
                                ;
                                ;  Reread and consolidate the segment that has the empty entry we
                                ;  are going to use unless that segment is the current segment.
                                ;
                                23$:    CMP     R0,CURSEG       ;IS SEG WE WANT THE CURRENT SEGMENT?
                                        BEQ     21$             ;BR IF YES
                                        MOV     R1,-(SP)        ;SAVE ADDRESS OF DIRECTORY ENTRY
                                        CALL    RDSEG           ;REREAD THE SEGMENT WITH THE EMPTY ENTRY
                                        MOV     (SP)+,R1        ;GET ADDRESS OF DIRECTORY ENTRY
                                        CALL    CONSOL          ;CONSOLIDATE IT
                                        BR      21$             ;RETURN SUCCESSFULLY
                                ;
                                ;  We could not satisfy the request.
                                ;
                                9$:     MOV     #1,R0           ;RETURN ERROR CODE 1
                                24$:    SEC                     ;SIGNAL ERROR ON RETURN
                                        BR      22$
                                ;
                                ;  Successful return
                                ;
                                21$:    CLC                     ;SIGNAL GOOD RETURN
                                22$:    MOV     (SP)+,R4
                                        MOV     (SP)+,R3
                                        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.
                                ;
012026  010246                  ADDENT: MOV     R2,-(SP)
012030  010346                          MOV     R3,-(SP)
                                ;
                                ;  See if there is room in the current directory segment for three new entries
                                ;  (free-0, tentative, free-remainder)
                                ;
012032  010146                          MOV     R1,-(SP)        ;SAVE ADDRESS OF ENTRY WE ARE WORKING ON
012034  005000                          CLR     R0              ;FIND END OF SEGMENT ENTRY
012036  004737  013222'                  CALL    NXTDIR
012042  010103                          MOV     R1,R3           ;GET ADDRESS OF END OF SEGMENT ENTRY
012044  012601                          MOV     (SP)+,R1        ;GET BACK ADDRESS OF EMPTY FILE ENTRY
012046  013700  002006'                  MOV     DIRSIZ,R0       ;GET SIZE OF A DIRECTORY ENTRY
012052  060003                          ADD     R0,R3           ;ADD SPACE NEEDED FOR 3 ENTRIES
012054  060003                          ADD     R0,R3
012056  060003                          ADD     R0,R3
012060  020327  002002'                  CMP     R3,#USRBUF+1024.;IS THERE ROOM IN THIS SEGMENT?
012064  103403                          BLO     1$              ;BR IF THERE IS ROOM
                                ;
                                ;  Directory segment is full.
                                ;  Attempt to split it.
                                ;
012066  004737  012326'                  CALL    SPLIT           ;SPLIT THE SEGMENT
012072  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.
                                ;
012074  016100  000000G         1$:     MOV     FD$LEN(R1),R0   ;GET SIZE OF EMPTY FILE ENTRY
012100  010261  000000G                 MOV     R2,FD$LEN(R1)   ;PUT IN ALLOCATED SIZE OF TENTATIVE FILE
012104  160200                          SUB     R2,R0           ;CALC # BLOCKS LEFT OVER IN EMPTY AREA
012106  010002                          MOV     R0,R2
012110  012761  000000G 000000G         MOV     #FS$TEN,FD$STA(R1);MARK ENTRY AS TENTATIVE
012116  113761  000000G 000000G         MOVB    CORUSR,FD$JOB(R1);PUT IN JOB NUMBER
012124  113761  000000G 000000G         MOVB    CHNNUM,FD$CHN(R1);PUT IN CHANNEL NUMBER
012132  005061  000000G                 CLR     FD$DAT(R1)      ;SAY NO CREATION DATE YET -- SET IN CLOSE
012136  010103                          MOV     R1,R3           ;SAVE ADDRESS OF TENTATIVE ENTRY
012140  062703  000000G                 ADD     #FD$NAM,R3      ;POINT TO FIELD WHERE NAME GOES
012144  012700  000002G                 MOV     #FILSPC+2,R0    ;POINT TO FILE SPEC WE ARE CREATING FOR
012150  012023                          MOV     (R0)+,(R3)+     ;MOVE IN FILE NAME
012152  012023                          MOV     (R0)+,(R3)+
012154  012023                          MOV     (R0)+,(R3)+     ;AND EXTENSTION
                                ;
```

```
                                ;  Add an empty file entry following the tentative file entry to hold
                                ;  any left over blocks.
                                ;
012156  063701  002006'                ADD     DIRSIZ,R1           ;POINT TO FOLLOWING DIR ENTRY
012162  004737  012244'                CALL    INSERT              ;INSERT AN EMPTY FILE ENTRY
012166  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).
                                ;
012172  163701  002006'                SUB     DIRSIZ,R1           ;POINT TO OUR NEW TENTATIVE FILE ENTRY
012176  163701  002006'                SUB     DIRSIZ,R1           ;POINT TO PREVIOUS ENTRY
012202  020127  000000C                CMP     R1,#USRBUF+DH$$SZ;WAS THERE A PREVIOUS ENTRY?
012206  103410                        BLO     3$                  ;BR IF NOT
012210  032761  000000G 000000G        BIT     #FS$TEN,FD$STA(R1);WAS PREVIOUS ENTRY A TENTATIVE FILE ENTRY?
012216  001404                        BEQ     3$                  ;BR IF NOT
012220  063701  002006'                ADD     DIRSIZ,R1           ;POINT BACK TO OUR NEW TENTATIVE ENTRY
012224  004737  012244'                CALL    INSERT              ;INSERT A ZERO-LENGTH EMPTY FILE ENTRY
012230  063701  002006'        3$:     ADD     DIRSIZ,R1           ;POINT TO TENTATIVE FILE ENTRY WE CREATED
                                ;
                                ;  Finished
                                ;
012234  000241                        CLC                         ;SIGNAL GOOD RETURN
012236  012603                9$:     MOV     (SP)+,R3
012240  012602                        MOV     (SP)+,R2
012242  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.
                                ;
012244  010146                  INSERT: MOV     R1,-(SP)
                                ;
                                ;  Locate end-of-segment entry.
                                ;
012246  012700  0000000                 MOV     #FS$EOS,R0      ;SEARCH FOR END-OF-SEGMENT MARKER
012252  030061  0000000                 BIT     R0,FD$STA(R1)   ;ARE WE POINTING TO IT NOW?
012256  001002                          BNE     2$              ;BR IF YES
012260  004737  013222'                 CALL    NXTDIR          ;FIND END OF SEGMENT ENTRY
                                ;
                                ;  Shove down all entries beyond insert point.
                                ;
012264  005721                  2$:     TST     (R1)+           ;POINT BEYOND END-OF-SEGMENT WORD
012266  010100                          MOV     R1,R0
012270  063700  002006'                 ADD     DIRSIZ,R0       ;MAKE ROOM FOR NEW ENTRY
012274  014140                  1$:     MOV     -(R1),-(R0)     ;SHOVE DOWN ALL ENTRIES
012276  020116                          CMP     R1,(SP)         ;TILL WE REACH INSERT POINT
012300  101375                          BHI     1$
                                ;
                                ;  Initialize the new entry.
                                ;
012302  013700  002006'                 MOV     DIRSIZ,R0       ;SIZE OF ENTRY (BYTES)
012306  006200                          ASR     R0              ;GET # WORDS
012310  005021                  3$:     CLR     (R1)+           ;ZERO THE ENTRY
012312  077002                          SOB     R0,3$
012314  012601                          MOV     (SP)+,R1        ;GET ADDRESS OF NEW ENTRY
012316  012761  0000000 0000000         MOV     #FS$EMP,FD$STA(R1);SAY ENTRY IS EMPTY
                                ;
                                ;  Finished
                                ;
012324  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.
                                ;
012326  010246                  SPLIT:  MOV     R2,-(SP)
012330  010346                          MOV     R3,-(SP)
012332  010446                          MOV     R4,-(SP)
012334  010546                          MOV     R5,-(SP)
012336  010105                          MOV     R1,R5           ;SAVE ADDRESS OF ENTRY WE ARE INTERESTED IN
                                ;
                                ;  See if there is a free segment for us to split into.
                                ;
012340  023737  002010' 002004'         CMP     DIRNSG,DIRHIS   ;IS THERE A FREE SEGMENT?
012346  101524                          BLOS    10$             ;BR IF NOT
012350  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.
                                ;
012354  012701  000000C                 MOV     #USRBUF+DH$$SZ,R1;POINT TO FIRST ENTRY IN SEGMENT
012360  005000                          CLR     R0              ;COUNT # ENTRIES IN R0
012362  032761  000000G 000000G 1$:      BIT     #FS$EOS,FD$STA(R1);IS THIS THE END OF SEGMENT MARKER?
012370  001004                          BNE     2$              ;BR IF YES
012372  005200                          INC     R0              ;COUNT ANOTHER ENTRY
012374  063701  002006'                 ADD     DIRSIZ,R1       ;POINT TO NEXT ENTRY
012400  000770                          BR      1$
                                ;  Leave approximately half the entries in the current segment.
012402  006200                  2$:     ASR     R0              ;GET 1/2 # ENTRIES
012404  013702  000000C                 MOV     USRBUF+DH$BLK,R2;GET BASE BLOCK # OF 1ST FILE IN SEGMENT
012410  012701  000000C                 MOV     #USRBUF+DH$$SZ,R1;POINT TO 1ST ENTRY
012414  066102  000000G         3$:     ADD     FD$LEN(R1),R2   ;KEEP TRACK OF BLOCK #'S OF FILES
012420  063701  002006'                 ADD     DIRSIZ,R1       ;POINT TO NEXT DIRECTORY ENTRY
012424  077005                          SOB     R0,3$           ;SKIP OVER 1/2 OF THE ENTRIES
                                ;  Split on 1st permanent or tentative entry beyond mid-point.
012426  032761  000000C 000000G 4$:     BIT     #<FS$PRM+FS$TEN>,FD$STA(R1);IS THIS A PERM OR TENTATIVE ENTRY?
012434  001005                          BNE     7$              ;BR IF YES
012436  066102  000000G                 ADD     FD$LEN(R1),R2   ;KEEP TRACK OF STARTING BLOCK NUMBERS
012442  063701  002006'                 ADD     DIRSIZ,R1       ;POINT TO NEXT ENTRY
012446  000767                          BR      4$
                                ;
                                ;  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.
012450  020501                7$:     CMP     R5,R1           ;WILL ENTRY GO IN OLD OR NEW SEGMENT?
012452  103404                        BLO     5$              ;BR IF GOING IN OLD SEGMENT
012454  160105                        SUB     R1,R5           ;CALCULATE ITS ADDRESS IN NEW SEGMENT
012456  062705  000000C             ADD     #USRBUF+DH$$SZ,R5
012462  005004                        CLR     R4              ;REMEMBER ENTRY GOES IN NEW SEGMENT
                                ;
                                ;  Truncate current segment and set link to new segment.
                                ;
012464  011146                5$:     MOV     (R1),-(SP)      ;SAVE FD$STAT FOR ENTRY FOLLOWING SPLIT
012466  012711  000000G             MOV     #FS$EOS,(R1)    ;SET END-OF-SEGMENT MARKER
012472  013746  000000C             MOV     USRBUF+DH$NXT,-(SP);SAVE LINK TO NEXT SEGMENT
012476  013703  002004'             MOV     DIRHIS,R3       ;GET # OF HIGHEST SEGMENT CURRENTLY IN USE
012502  005203                        INC     R3              ;GET # OF 1ST FREE SEGMENT
012504  010337  000000C             MOV     R3,USRBUF+DH$NXT;SET AS NEW LINK
012510  004737  013706'             CALL    WRTSEG          ;WRITE OUT TRUNCATED OLD SEGMENT
                                ;
                                ;  Now set up new segment.
                                ;
                                ;  Set new header.
012514  012637  000000C                 MOV     (SP)+,USRBUF+DH$NXT;SET FLINK
012520  010337  002002'                 MOV     R3,CURSEG       ;SET OUR SEGMENT #
012524  010237  000000C                 MOV     R2,USRBUF+DH$BLK;STARTING BLOCK # FOR FILES IN SEGMENT
                                ;  Slide up directory entries from end of segment.
012530  012611                          MOV     (SP)+,(R1)      ;FIX FD$STA OF 1ST ENTRY
012532  012700  000000C                 MOV     #USRBUF+DH$$SZ,R0;POINT TO TOP OF AREA FOR ENTRIES
012536  012702  002002'                 MOV     #USRBUF+1024.,R2;POINT PAST END OF BUFFER
012542  160102                          SUB     R1,R2           ;GET # BYTES TO MOVE
012544  006202                          ASR     R2              ;GET # WORDS TO MOVE
012546  012120                6$:       MOV     (R1)+,(R0)+     ;MOVE UP ENTRIES
012550  077202                          SOB     R2,6$
                                ;  Write out the new segment.
012552  004737  013706'                 CALL    WRTSEG
                                ;
                                ;  Update information in header of 1st (master) directory segment.
                                ;
012556  004737  013454'                 CALL    RDSEG1          ;READ IN DIR SEG 1
012562  010337  000000C                 MOV     R3,USRBUF+DH$HIS;SET # OF HIGHEST SEG IN USE
012566  010337  002004'                 MOV     R3,DIRHIS
012572  004737  013706'                 CALL    WRTSEG          ;WRITE OUT SEG # 1
                                ;
                                ;  Now read back in the segment that has the directory entry we are
                                ;  interested in.
                                ;
012576  005704                          TST     R4              ;IS ENTRY IN OLD OR NEW DIRECTORY SEGMENT?
012600  001401                          BEQ     9$              ;BR IF IN NEW SEGMENT
012602  010403                          MOV     R4,R3           ;GET NUMBER OF OLD SEGMENT
012604  010300                9$:       MOV     R3,R0           ;SET AS ARGUMENT TO RDSEG
012606  004737  013570'                 CALL    RDSEG           ;READ IN SEGMENT WITH ENTRY WE WANT
                                ;
                                ;  Finished
                                ;
012612  010501                          MOV     R5,R1           ;GET ADDRESS OF ENTRY OF INTEREST
012614  000241                          CLC                     ;SIGNAL GOOD RETURN
012616  000401                          BR      11$
                                ;
```

```
                                        ;  Directory overflow
                                        ;
012620  000261                  10$:    SEC                             ;SIGNAL ERROR ON RETURN
012622  012605                  11$:    MOV     (SP)+,R5
012624  012604                          MOV     (SP)+,R4
012626  012603                          MOV     (SP)+,R3
012630  012602                          MOV     (SP)+,R2
012632  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.
                                ;
012634  010146                  CONSOL: MOV     R1,-(SP)
012636  010246                          MOV     R2,-(SP)
012640  010346                          MOV     R3,-(SP)
012642  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get exclusive access to job context block buffer
                                ;
012644                                  OCALL   GETCXT          ;Get exclusive access to context block buffer
                                ;
                                ;  Pass 1:  Convert unopen tentative entries into empty entries.
                                ;
012652  012701  000000C                 MOV     #USRBUF+DH$$SZ,R1;POINT TO 1ST ENTRY
012656  013704  002006'                 MOV     DIRSIZ,R4       ;CARRY DIRECTORY ENTRY SIZE IN R4
012662  160401                          SUB     R4,R1           ;POINT IN FRONT OF 1ST ENTRY FOR NXTDIR
                                ;  Find the next tentative file entry.
012664  012700  0000006         3$:     MOV     #FS$TEN,R0      ;LOOK FOR A TENTATIVE FILE ENTRY
012670  004737  013222'                 CALL    NXTDIR          ;DO SEARCH
012674  103436                          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.
012676  116102  0000006                 MOVB    FD$JOB(R1),R2   ;GET # OF JOB USING ENTRY
012702  001427                          BEQ     1$              ;BR IF NOT TSX JOB
012704  032702  000001                  BIT     #1,R2           ;TSX JOB NUMBER CAN NEVER BE ODD
012710  001024                          BNE     1$              ;BR IF NOT TSX JOB USING CHANNEL
012712  020227  0000006                 CMP     R2,#LSTSL       ;MAKE SURE JOB NUMBER IS NOT TOO BIG
012716  101021                          BHI     1$              ;BR IF NOT TSX JOB
012720  032762  0000006 0000006         BIT     #$DILUP,LSW(R2) ;IS THAT JOB LOGGED ON?
012726  001415                          BEQ     1$              ;BR IF NOT
012730  116103  0000006                 MOVB    FD$CHN(R1),R3   ;GET CHANNEL # ASSOCIATED WITH FILE
012734                                  OCALL   GETCHA          ;GET POINTER TO CHANNEL BLOCK
012742  032760  0000006 0000006         BIT     #CS$OPN,C.CSW(R0);IS THAT CHANNEL OPEN?
012750  001404                          BEQ     1$              ;BR IF NOT
012752  032760  0000006 0000006         BIT     #CS$ENT,C.CSW(R0);IS CHANNEL OPEN FOR CREATING A NEW FILE?
012760  001003                          BNE     2$              ;BR IF YES
                                ;  Convert this tentative file entry to an empty file entry.
012762  012761  0000006 0000006 1$:     MOV     #FS$EMP,FD$STA(R1);SAY THIS IS AN EMPTY FILE ENTRY
                                ;  Check next entry.
012770  000735                  2$:     BR      3$
```

```
                                        ;
                                        ;  Pass 2:  Consolidate consecutive empty entries and emtpy entries
                                        ;           of zero length preceded by a permanent file entry.
                                        ;
012772  012701  000000C         5$:     MOV     #USRBUF+DH$$SZ,R1;POINT TO 1ST ENTRY
012776  160401                          SUB     R4,R1           ;POINT IN FRONT OF 1ST ENTRY
013000  012700  000000G         11$:    MOV     #FS$EMP,R0      ;SEARCH FOR NEXT EMTPY FILE ENTRY
013004  004737  013222'                 CALL    NXTDIR
013010  103456                          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.
013012  010102                  7$:     MOV     R1,R2           ;GET ADDRESS OF CURRENT ENTRY
013014  060402                          ADD     R4,R2           ;POINT TO NEXT ENTRY
013016  032762  000000G 000000G         BIT     #FS$EMP,FD$STA(R2);IS NEXT ENTRY EMPTY TOO?
013024  001417                          BEQ     8$              ;BR IF NOT
                                        ;  Combine the two entries.
013026  066261  000000G 000000G         ADD     FD$LEN(R2),FD$LEN(R1);COMBINE ALLOCATED SIZES
                                        ;  Remove the second emtpy entry.
013034  010146                          MOV     R1,-(SP)        ;REMEMBER ADDRESS OF 1ST EMPTY ENTRY
013036  012700  000000G                 MOV     #FS$EOS,R0      ;SEARCH FOR END OF SEGMENT MARKER
013042  004737  013222'                 CALL    NXTDIR
                                        ;  R1 now points to end of segment marker.
                                        ;  Move up entries over the one being removed (pointed to by R2).
013046  010200                          MOV     R2,R0           ;GET ADDRESS OF ENTRY TO BE REMOVED
013050  060400                          ADD     R4,R0           ;POINT TO FOLLOWING ENTRY
013052  012022                  6$:     MOV     (R0)+,(R2)+     ;MOVE UP FOLLOWING ENTRIES OVER ONE BEING REMOVED
013054  020001                          CMP     R0,R1           ;MOVED END-OF-SEGMENT MARKER YET?
013056  101775                          BLOS    6$              ;BR IF NOT
013060  012601                          MOV     (SP)+,R1        ;GET BACK ADDRESS OF 1ST EMPTY ENTRY
013062  000753                          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.
                                        ;
013064  005761  000000G         8$:     TST     FD$LEN(R1)      ;IS THIS EMPTY OF ZERO LENGTH?
013070  001025                          BNE     9$              ;BR IF NOT
013072  010102                          MOV     R1,R2           ;GET ADDRESS OF ENTRY
013074  160402                          SUB     R4,R2           ;GET ADDRESS OF PREVIOUS ENTRY
013076  020227  000000C                 CMP     R2,#USRBUF+DH$$SZ;ARE WE 1ST ENTRY?
013102  103420                          BLO     9$              ;BR IF YES
013104  032762  000000G 000000G         BIT     #FS$PRM,FD$STA(R2);IS PREVIOUS ENTRY A PERMANENT FILE?
013112  001414                          BEQ     9$              ;BR IF NOT
                                        ;  Remove a zero length empty entry following a permanent file entry.
013114  010246                          MOV     R2,-(SP)        ;SAVE ADDRESS OF PERMANENT FILE ENTRY
013116  010102                          MOV     R1,R2           ;GET ADDRESS OF EMPTY ENTRY
013120  012700  000000G                 MOV     #FS$EOS,R0      ;SEARCH FOR END OF SEGMENT ENTRY
013124  004737  013222'                 CALL    NXTDIR
013130  010200                          MOV     R2,R0           ;GET ADDRESS OF EMPTY ENTRY
013132  060400                          ADD     R4,R0           ;POINT TO FOLLOWING ENTRY
013134  012022                  10$:    MOV     (R0)+,(R2)+     ;MOVE UP FOLLOWING ENTRIES OVER EMPTY ENTRY
013136  020001                          CMP     R0,R1           ;HAVE WE MOVED END-OF-SEGMENT MARKER YET?
013140  101775                          BLOS    10$             ;BR IF NOT
013142  012601                          MOV     (SP)+,R1        ;GET BACK ADDRESS OF PERMANENT FILE ENTRY
                                        ;  Look for next empty entry.
013144  000715                  9$:     BR      11$
                                        ;
                                        ;  Finished
```

```
                                        ;
013146                          12$:    OCALL   FRECXT          ;Release context block buffer
013154  012604                          MOV     (SP)+,R4
013156  012603                          MOV     (SP)+,R3
013160  012602                          MOV     (SP)+,R2
013162  012601                          MOV     (SP)+,R1
013164  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$xxx entry type flags.
                        ;   USRBUF = Current directory segment.
                        ;
                        ;  Outputs:
                        ;   R1 = Address of next directory entry of the specified type.
                        ;   USRBUF = Directory segment that contains the entry.
                        ;   C-flag set if no entry of the specified type can be found.
                        ;
013166                  GETDIR:
                        ;
                        ;  See if we can find entry of specified type in the current directory segment.
                        ;
013166  004737  013222' 1$:     CALL    NXTDIR          ;SEARCH FOR ENTRY OF SPECIFIED TYPE
013172  103012                  BCC     9$              ;BR IF WE FOUND ONE
                        ;
                        ;  Hit end of current directory segment.  See if there are more segments.
                        ;
013174  010046                  MOV     R0,-(SP)
013176  013700  000000C         MOV     USRBUF+DH$NXT,R0;GET # OF NEXT SEGMENT IN LIST
013202  001404                  BEQ     2$              ;BR IF REACHED END
013204  004737  013570'         CALL    RDSEG           ;READ IN NEXT SEGMENT
013210  012600                  MOV     (SP)+,R0        ;GET BACK TYPE FLAGS
013212  000765                  BR      1$              ;SEARCH THIS SEGMENT
                        ;
                        ;  No entry of the specified type was found.
                        ;
013214  012600          2$:     MOV     (SP)+,R0        ;CLEAN OFF STACK
013216  000261                  SEC                     ;SIGNAL FAILURE ON RETURN
                        ;
                        ;  finished
                        ;
013220  000207          9$:     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$xxx directory entry type flags.
                                ;   R1 = Address of last directory entry.
                                ;
                                ;  Outputs:
                                ;   R1 = Address of directory entry found.
                                ;   C-flag set if no entry of specified type could be found.
                                ;
013222  063701  002006'         NXTDIR: ADD     DIRSIZ,R1       ;POINT TO NEXT DIRECTORY ENTRY
013226  030061  000000G                 BIT     R0,FD$STA(R1)   ;IS THIS ENTRY OF THE RIGHT TYPE?
013232  001006                         BNE     1$              ;BR IF YES
013234  032761  000000G 000000G        BIT     #FS$EOS,FD$STA(R1);IS THIS THE END-OF-SEGMENT MARKER?
013242  001767                         BEQ     NXTDIR          ;KEEP SEARCHING IF NOT
                                ;  Hit end of segment.
013244  000261                         SEC                     ;SIGNAL FAILURE ON RETURN
013246  000401                         BR      9$
                                ;  Found entry.
013250  000241                 1$:     CLC                     ;SIGNAL SUCCESS ON RETURN
013252  000207                 9$:     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.
                                ;
013254  032737  000001  000020 DIDDLE: BIT     #1,EMTBLK+2     ;DOES USER WANT TO DIDDLE WITH THE ENTRY?
013262  001455                        BEQ     9$              ;BR IF NOT
013264  010246                        MOV     R2,-(SP)
013266  010346                        MOV     R3,-(SP)
013270  010446                        MOV     R4,-(SP)
013272  010546                        MOV     R5,-(SP)
013274  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.
                                ;
013276  012703  000300                MOV     #USRUCA,R3      ;GET ADDRESS OF AREA IN USER'S AREA
013302  012700  000000C               MOV     #FD$$SZ/2,R0    ;GET # WORDS TO MOVE
013306  012146                1$:     MOV     (R1)+,-(SP)     ;PICK UP WORD FROM DIRECTORY ENTRY
013310  106623                        MTPD    (R3)+           ;MOVE TO AREA IN USER'S SPACE
013312  077003                        SOB     R0,1$           ;MOVE ALL OF DIRECTORY ENTRY
                                ;
                                ;  Use special EMT to get control back from user's routine.
                                ;
013314  013703  000000G               MOV     EMTCAD,R3       ;Get pointer to top of return addr stack
013320  012743  013364'               MOV     #5$,-(R3)       ;Push our return address
013324  010337  000000G               MOV     R3,EMTCAD       ;Save updated stack pointer
013330  106506                        MFPD    SP              ;GET USER'S STACK POINTER
013332  012603                        MOV     (SP)+,R3
013334  012746  000000G               MOV     #CPLEMT,-(SP)   ;PUSH ADDR OF EMT INSTRUCTION ON USER'S STACK
013340  106643                        MTPD    -(R3)
013342  010346                        MOV     R3,-(SP)        ;NOW STORE UPDATED USER STACK POINTER
013344  106606                        MTPD    SP
                                ;
                                ;  Now enter user's routine in user mode.
                                ;  (Use RTI to do this)
                                ;  On entry to user's routine, R1 points to the directory entry.
                                ;
013346  012701  000300G               MOV     #USRUCA+FD$DAT,R1;MAKE R1 POINT TO DATE WORD OF DIRECTORY ENTRY
013352  013746  000000G               MOV     EMTPS,-(SP)     ;PS
013356  013746  000010G               MOV     EMTBLK+10,-(SP) ;PC = Address of user's routine
013362  000002                        RTI                     ;ENTER USER'S ROUTINE
                                ;
                                ;  Return here from user's routine (when EMT is done).
                                ;
```

```
                                        ;  Move directory entry back to directory buffer.
                                        ;
013364  011602                  5$:       MOV     (SP),R2           ;GET ADDRESS WHERE WE SHOULD PUT DIR ENTRY
013366  012703  000300                    MOV     #USRUCA,R3        ;POINT TO AREA IN USER'S AREA WITH DIR ENTRY
013372  012700  000000C                   MOV     #FD$$SZ/2,R0      ;GET # WORDS TO MOVE
013376  106523                  2$:       MFPD    (R3)+             ;GET A WORD FROM USER'S DIRECTORY ENTRY
013400  012622                            MOV     (SP)+,(R2)+       ;MOVE BACK TO OUR INTERNAL AREA
013402  077003                            SOB     R0,2$             ;MOVE ALL OF ENTRY
                                        ;
                                        ;  Finished
                                        ;
013404  012601                            MOV     (SP)+,R1
013406  012605                            MOV     (SP)+,R5
013410  012604                            MOV     (SP)+,R4
013412  012603                            MOV     (SP)+,R3
013414  012602                            MOV     (SP)+,R2
013416  000207                  9$:       RETURN
```

```
                                        .SBTTL  SBCALC -- Calculate starting block # for a file entry
                                ;-------------------------------------------------------------------
                                ;  SBCALC is called to calculate the starting block number of a file.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of directory entry for file.
                                ;   USRBUF = Directory segment containing entry.
                                ;
                                ;  Outputs:
                                ;   R0 = Starting block number of the file.
                                ;
013420  010246                  SBCALC: MOV     R2,-(SP)
013422  013700  000000C                 MOV     USRBUF+DH$BLK,R0;GET BLOCK # OF 1ST FILE IN THIS DIR SEGMENT
013426  012702  000000C                 MOV     #USRBUF+DH$$SZ,R2;POINT TO 1ST DIR ENTRY IN THIS SEGMENT
013432  020201                  1$:     CMP     R2,R1           ;HAVE WE REACHED OUR ENTRY?
013434  001405                          BEQ     2$              ;BR IF YES
013436  066200  0000000                 ADD     FD$LEN(R2),R0   ;ADD SPACE ALLOCATED TO THAT FILE ENTRY
013442  063702  002006'                 ADD     DIRSIZ,R2       ;POINT TO NEXT ENTRY
013446  000771                          BR      1$
013450  012602                  2$:     MOV     (SP)+,R2
013452  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.
                ;
013454          RDSEG1:
                ;
                ;  Read in directory segment # 1
                ;
013454  012700  000001          MOV     #1,R0           ;SAY WE WANT SEG # 1
013460  004737  013570'          CALL    RDSEG           ;READ THE SEGMENT IN
                ;
                ;  Set up parameters about the directory
                ;
013464  012701  000002'          MOV     #USRBUF,R1      ;POINT TO BUFFER WITH DIRECTORY SEGMENT
013470  016137  000000G 002004'  MOV     DH$HIS(R1),DIRHIS;HIGHEST SEGMENT # USED
013476  016137  000000G 002010'  MOV     DH$NSG(R1),DIRNSG;NUMBER OF DIRECTORY SEGMENTS AVAILABLE
013504  001425                  BEQ     5$              ;BR IF INVALID DIRECTORY
013506  023727  002010' 000037    CMP     DIRNSG,#31.     ;CHECK TO SEE IF VALID
013514  101021                  BHI     5$              ;BR IF INVALID
013516  016100  000000G          MOV     DH$NEB(R1),R0   ;# EXTRA BYTES PER DIRECTORY ENTRY
013522  020027  001000          CMP     R0,#512.        ;CHECK IF REASONABLE
013526  101014                  BHI     5$              ;BR IF INVALID
013530  062700  000000G          ADD     #FD$OPT,R0      ;# STANDARD BYTES PER DIRECTORY ENTRY
013534  010037  002006'          MOV     R0,DIRSIZ       ;TOTAL # BYTES PER DIRECTORY ENTRY
013540  026127  000000G 000000G  CMP     DH$BLK(R1),#DH$$BS;MAKE SURE BASE BLOCK # IS REASONABLE
013546  103404                  BLO     5$              ;BR IF INVALID
                ;
                ;  Point R1 in front of 1st directory entry so NXTDIR will get 1st real entry.
                ;
013550  062701  000000G          ADD     #DH$$SZ,R1      ;POINT TO 1ST REAL DIRECTORY ENTRY
013554  160001                  SUB     R0,R1           ;POINT BACK 1 ENTRY
                ;
                ;  Finished
                ;
013556  000207                  RETURN
                ;
                ;  Invalid directory
                ;
013560  012700  177755          5$:     MOV     #UERR6,R0       ;FATAL ERROR
013564  000137  015144'          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.
                                ;
013570  020037  002002'         RDSEG:  CMP     R0,CURSEG       ;IS DESIRED SEGMENT ALREADY IN BUFFER?
013574  001437                         BEQ     1$              ;BR IF YES
013576  010001                         MOV     R0,R1           ;GET SEG #
                                ;  Convert segment # to absolute block #.
013600  010137  002002'         2$:     MOV     R1,CURSEG       ;SAVE CURRENT SEGMENT #
013604  005301                         DEC     R1              ;1ST DIRECTORY SEGMENT IS # 1
013606  006301                         ASL     R1              ;CVT SEGMENT # TO BLOCK #
013610  062701  000000G                ADD     #DH$$BS,R1      ;BASE BLOCK # OF 1ST SEGMENT
                                ;  Read in the segment.
013614                                  .READW  #USREMT,CHNNUM,#USRBUF,#512.,R1
013656  103006                         BCC     1$              ;BR IF NO ERROR
013660  004737  015214'                CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
013664  012700  177775                 MOV     #-3,R0          ;DIRECTORY READ ERROR
013670  000137  015144'                JMP     USRCLS
                                ;  Set R1 to point to dummy directory entry in front of first real one.
013674  012701  000000C         1$:     MOV     #USRBUF+DH$$SZ,R1;POINT TO FIRST REAL ENTRY IN SEGMENT
013700  163701  002006'                 SUB     DIRSIZ,R1       ;POINT IN FRONT OF FIRST ONE
013704  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.
                                ;
013706  010146                  WRTSEG: MOV     R1,-(SP)
013710  013701  002002'                 MOV     CURSEG,R1       ;GET CURRENT SEGMENT #
013714  005301                          DEC     R1              ;1ST SEGMENT IS # 1
013716  006301                          ASL     R1              ;CVT SEGMENT # TO BLOCK #
013720  062701  000000G                 ADD     #DH$$BS,R1      ;BASE BLOCK # OF 1ST SEGMENT
013724                                  .WRITW  #USREMT,CHNNUM,#USRBUF,#512.,R1
013766  103006                          BCC     1$              ;BR IF WRITE OK
013770  004737  015214'                 CALL    ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
013774  012700  177775                  MOV     #-3,R0          ;DIRECTORY WRITE ERROR
014000  000137  015144'                 JMP     USRCLS          ;ABORT OPERATION
014004  012601                  1$:     MOV     (SP)+,R1
014006  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.
                                ;
014010                          SPLDIR:
                                ;
                                ;  Free the USR module.
                                ;
014010  004737  010132'                 CALL    FREUSR          ;FREE THE USR DATA BASE FOR OTHERS TO USE
                                ;
                                ;  Obtain exclusive access to special device data base.
                                ;
014014  004737  014376'                 CALL    GETSPD          ;GAIN EXCLUSIVE ACCESS TO SPD DATA BASE
                                ;
                                ;  Copy the untranslated device/file specification from the user's area
                                ;  to a system buffer.
                                ;
014020  013701  0000020                 MOV     EMTBLK+2,R1     ;Point to file spec in user's area
014024  042701  000001                  BIC     #1,R1           ;Make sure address is even
014030  010246                          MOV     R2,-(SP)        ;Save operation function code
014032  012702  0000000                 MOV     #SPFLDV,R2      ;Point to buffer where name is to be stored
014036  004737  007704'                 CALL    CPYSPC          ;Copy file spec to SPFLDV
014042  012602                          MOV     (SP)+,R2        ;Restore operation function code
014044  005037  0000000                 CLR     SPSIZE          ;HANDLER RETURNS FILE SIZE HERE
014050  005037  0000000                 CLR     SPUSR           ;HANDLER RETURNS ERROR CODE HERE
                                ;
                                ;  Build an I/O queue entry to do the operation.
                                ;
014054  004737  0000000                 CALL    GETQ            ;GET A FREE I/O QUEUE ENTRY
014060  010361  0000000                 MOV     R3,Q.UCSW(R1)   ;SET ADDRESS OF CHANNEL CSW
014064  013761  0000000 0000000         MOV     @#KPAR6,Q.PA6(R1);Save mapping for context block
014072  010105                          MOV     R1,R5           ;Get address of Q element
014074  062705  0000000                 ADD     #Q.ICSW,R5      ;Point to internal CSW cell
014100  010561  0000000                 MOV     R5,Q.CSW(R1)    ;Set up pointer to internal CSW cell
014104  010300                          MOV     R3,R0           ;Get pointer to original channel block
014106  012025                          MOV     (R0)+,(R5)+     ;Copy original channel block to internal one
014110  012025                          MOV     (R0)+,(R5)+
014112  012025                          MOV     (R0)+,(R5)+
014114  012025                          MOV     (R0)+,(R5)+
014116  011015                          MOV     (R0),(R5)
014120  016300  0000000                 MOV     C.CSW(R3),R0    ;Get the channel status word
014124  042700  177701                  BIC     #^C76,R0        ;Isolate the device index number
014130  110061  0000000                 MOVB    R0,Q.DEVX(R1)   ;store the device index number
014134  013700  0000060                 MOV     EMTBLK+6,R0     ;GET FILE SEQUENCE # FOR .ENTER
014140  020227  0000000                 CMP     R2,#DF$ENT      ;ARE WE DOING A .ENTER?
014144  001402                          BEQ     6$              ;BR IF YES
014146  013700  0000040                 MOV     EMTBLK+4,R0     ;GET SEQUENCE # FOR .LOOKUP, .DELETE, ETC.
014152  010061  0000000         6$:     MOV     R0,Q.BLKN(R1)   ;SET SEQUENCE # IN Q.BLKN FIELD
014156  013761  0000040 0000000         MOV     EMTBLK+4,Q.WCNT(R1);SET FILE ENTER SIZE IN Q.WCNT FIELD
```

```
     58 014164  110261  0000000                      MOVB    R2,Q.FUNC(R1)    ;SET OPERATION FUN CODE (LOOKUP, ENTER, ETC.)
     59 014170  116361  0000000 0000000              MOVB    C.DEVQ(R3),Q.UNIT(R1);SET UNIT NUMBER
     60 014176  113700  0000000                      MOVB    CORUSR,R0        ;Get current job index number
     61 014202  006200                               ASR     R0               ;Convert to job number
     62 014204  110061  0000000                      MOVB    R0,Q.JOB(R1)     ;Set job number in queue element
     63 014210  072027  000003                       ASH     #3,R0            ;Position for Q.JNUM field
     64 014214  042700  177407                       BIC     #^C<370>,R0      ;Clear all but job number field
     65 014220  150061  0000000                      BISB    R0,Q.JNUM(R1)    ;Set job # in queue element
     66 014224  012704  0000000                      MOV     #SPFLNM,R4       ;GET ADDRESS OF FILE SPEC BUFFER
     67 014230  005005                               CLR     R5               ;SET FOR SHIFT
     68 014232  073427  177772                       ASHC    #-6,R4           ;SHIFT LOW-ORDER 6 BITS INTO R5
     69 014236  000241                               CLC
     70 014240  006005                               ROR     R5               ;RIGHT JUSTIFY LOW-ORDER 6 BITS IN R5
     71 014242  072527  177767                       ASH     #-9.,R5
     72 014246  062705  0000000                      ADD     #VPAR6,R5        ;BIAS ADDRESS TO BE IN PAR6 RANGE
     73 014252  010561  0000000                      MOV     R5,Q.BUFF(R1)    ;SET THIS AS VIRTUAL BUFFER ADDRESS
     74 014256  010461  0000000                      MOV     R4,Q.PAR(R1)     ;SET PAR6 BASE OFFSET
     75 014262  005061  0000000                      CLR     Q.COMP(R1)       ;NO COMPLETION ROUTINE
     76 014266  013761  0000000 0000000              MOV     CHNNUM,Q.CHAN(R1);SET USER'S CHANNEL NUMBER
     77                                      ;
     78                                      ;  Initiate the I/O operation.
     79                                      ;
     80 014274  004737  0000000                          CALL    QIO              ;QUEUE THE I/O OPERATION
     81                                      ;
     82                                      ;  Wait for I/O to finish.
     83                                      ;
     84 014300                                           .WAIT   CHNNUM           ;WAIT FOR OPERATION TO FINISH
     85 014310  103003                                   BCC     3$               ;BR IF NO I/O ERROR
     86                                      ;  Directory I/O error.
     87 014312  012737  177775  0000000                    MOV     #-3,SPUSR        ;SET I/O ERROR CODE
     88                                      ;
     89                                      ;  Return handler reported file size to user in R0.
     90                                      ;
     91 014320  013737  0000000 0000000 3$:      MOV     SPSIZE,URO       ;RETURN HANDLER REPORTED FILE SIZE IN USER'S R0
     92 014326  013705  0000000                          MOV     SPUSR,R5         ;GET HANDLER REPORTED ERROR CODE
     93                                      ;
     94                                      ;  Release special device data base area.
     95                                      ;
     96 014332  004737  014440'                          CALL    FRESPD           ;FREE SPD AREA
     97                                      ;
     98                                      ;  See if we should purge the channel.
     99                                      ;
    100 014336  020227  0000000                          CMP     R2,#DF$LOK       ;DOING A LOOKUP?
    101 014342  001405                                   BEQ     4$               ;BR IF YES
    102 014344  020227  0000000                          CMP     R2,#DF$ENT       ;DOINT AN ENTER?
    103 014350  001402                                   BEQ     4$               ;BR IF YES
    104 014352  005063  0000000                          CLR     C.CSW(R3)        ;PURGE THE CHANNEL
    105                                      ;
    106                                      ;  See if handler reported an error
    107                                      ;
    108 014356  010500                          4$:      MOV     R5,R0            ;DID HANDLER RETURN AN ERROR CODE?
    109 014360  001002                                   BNE     5$               ;BR IF YES
    110 014362  000137  0000000                          JMP     EMTXIT           ;NORMAL EMT EXIT
    111 014366  005063  0000000                 5$:      CLR     C.CSW(R3)        ;PURGE CHANNEL IF ERROR OCCURED
    112 014372  000137  0000000                          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.
                                ;
014376  113700  0000000G      GETSPD: MOVB    SPDJOB,R0       ;IS SPD DATA BASE FREE NOW?
014402  001412                        BEQ     1$              ;BR IF YES
014404  120037  0000000G              CMPB    R0,CORUSR       ;HAVE WE ALREADY GOT IT?
014410  001412                        BEQ     2$              ;BR IF YES
                                ;
                                ;  Someone else owns SPD data base.
                                ;  Suspend our job until it becomes free.
                                ;
014412  012700  0000000G              MOV     #S$QSPD,R0      ;QUEUE FOR SPD DATA BASE
014416  004737  0000000G              CALL    QNSPNX          ;PUT US AT TAIL OF QUEUE FOR IT
014422  004737  0000000G              CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
014426  000763                        BR      GETSPD          ;NOW TRY TO GET SPD
                                ;
                                ;  We can get SPD data base now.
                                ;
014430  113737  0000000G 0000000G 1$:     MOVB    CORUSR,SPDJOB   ;CLAIM SPD DATA BASE FOR US
014436  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.
                                ;
014440  123737  0000000G 0000000G FRESPD: CMPB    CORUSR,SPDJOB   ;DO WE CURRENTLY OWN SPD?
014446  001006                        BNE     1$              ;BR IF NOT
014450  105037  0000000G              CLRB    SPDJOB          ;FREE IT
                                ;
                                ;  Restart any job that is waiting for SPD data base.
                                ;
014454  012700  0000000G              MOV     #S$QSPD,R0      ;GET SPD WAIT QUEUE STATE CODE
014460  004737  0000000G              CALL    UREGO           ;RESTART ANY WAITING JOB
                                ;
                                ;  Finished
                                ;
014464  000207                  1$:     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 #
                        ;
014466  010246         CSHADD: MOV     R2,-(SP)
014470  010346                 MOV     R3,-(SP)
014472  010446                 MOV     R4,-(SP)
014474  010546                 MOV     R5,-(SP)
                        ;
                        ;  See if we are caching file entries for this device
                        ;
014476  004737  014776'        CALL    CSHTST          ;ARE WE TO CACHE ENTRIES FOR THIS DEVICE?
014502  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).
                        ;
014504  012702  000000C        MOV     #CSHHD-FC$LNK,R2;DUMMY POINTER TO PHANTOM FIRST ENTRY
014510  010203         1$:     MOV     R2,R3           ;REMEMBER ADDRESS OF PREVIOUS ENTRY
014512  016202  000000G        MOV     FC$LNK(R2),R2   ;CHAIN FORWARD TO NEXT ENTRY IN LIST
014516  005762  000000G        TST     FC$CDX(R2)      ;IS THIS ENTRY UNUSED?
014522  001403                 BEQ     2$              ;BR IF YES
014524  005762  000000G        TST     FC$LNK(R2)      ;IS THIS THE LAST ENTRY IN LIST?
014530  001367                 BNE     1$              ;BR IF NOT
                        ;
                        ;  Relink cache entry at head of chain (it is most recently used).
                        ;
014532  016263  000000G 000000G 2$:     MOV     FC$LNK(R2),FC$LNK(R3);RELINK CHAIN AROUND US
014540  013762  000000G 000000G        MOV     CSHHD,FC$LNK(R2);NOW LINK US AT FRONT OF LIST
014546  010237  000000G              MOV     R2,CSHHD
                        ;
                        ;  Copy info from file directory entry to cache entry.
                        ;
014552  010203                 MOV     R2,R3           ;GET ADDRESS OF CACHE ENTRY
014554  010104                 MOV     R1,R4           ;GET ADDRESS OF DIRECTORY ENTRY
014556  012700  000000G        MOV     #FD$$SZ,R0      ;GET # BYTES TO MOVE
014562  006200                 ASR     R0              ;GET # WORDS TO MOVE
014564  012423         3$:     MOV     (R4)+,(R3)+     ;COPY DIRECTORY INFO TO CACHE ENTRY
014566  077002                 SOB     R0,3$
                        ;
                        ;  Store pointer to cached-device table entry into file cache entry
                        ;
014570  010562  000000G        MOV     R5,FC$CDX(R2)   ;REMEMBER WHICH DEVICE FILE IS ASSOC WITH
                        ;
                        ;  Store starting block number of file
                        ;
014574  004737  013420'        CALL    SBCALC          ;CALCULATE STARTING BLOCK # OF FILE
014600  010062  000000G        MOV     R0,FC$SBL(R2)   ;SAVE STARTING BLOCK #
                        ;
                        ;  Finished
                        ;
014604  012605         9$:     MOV     (SP)+,R5
014606  012604                 MOV     (SP)+,R4
```

```
     58 014610  012603                          MOV     (SP)+,R3
     59 014612  012602                          MOV     (SP)+,R2
     60 014614  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.
                                ;
014616  010146                  CSHDEL: MOV     R1,-(SP)
                                ;
                                ;  Search for file entry in cache table.
                                ;
014620  062701  000000G                 ADD     #FD$NAM,R1      ;POINT TO FILE NAME IN DIRECTORY ENTRY
014624  004737  014642'                 CALL    CSHCHK          ;SEE IF ENTRY IS IN DIRECTORY CACHE
014630  103402                          BCS     4$              ;BR IF FILE ENTRY IS NOT IN CACHE
                                ;
                                ;  Found entry in cache table.
                                ;  Mark it as deleted.
                                ;
014632  005061  000000G                 CLR     FC$CDX(R1)      ;MARK ENTRY AS DELETED
                                ;
                                ;  Finished
                                ;
014636  012601                  4$:     MOV     (SP)+,R1
014640  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.
                                ;
014642  010246                  CSHCHK: MOV     R2,-(SP)
014644  010346                          MOV     R3,-(SP)
014646  010546                          MOV     R5,-(SP)
014650  010103                          MOV     R1,R3           ;CARRY FILE NAME POINTER IN R3
                                ;
                                ;  See if this device is being cached and get pointer to device table entry
                                ;
014652  004737  014776'                 CALL    CSHTST          ;SEE IF THIS DEVICE IS BEING CACHED
014656  103442                          BCS     4$              ;BR IF DEVICE IS NOT BEING CACHED
                                ;
                                ;  Search for file entry in cache table.
                                ;
014660  012701  000000C                 MOV     #CSHHD-FC$LNK,R1;GET POINTER TO DUMMY STARTING POINT
014664  010102                  1$:     MOV     R1,R2           ;REMEMBER ADDRESS OF PREVIOUS ENTRY
014666  016101  0000000                 MOV     FC$LNK(R1),R1   ;GET ADDRESS OF NEXT ENTRY IN LIST
014672  020561  0000000                 CMP     R5,FC$CDX(R1)   ;IS FILE ON CACHED DEVICE?
014676  001012                          BNE     2$              ;BR IF NOT
014700  010300                          MOV     R3,R0           ;POINT TO FILE NAME
014702  022061  0000000                 CMP     (R0)+,FD$NAM(R1);COMPARE FILE NAMES
014706  001006                          BNE     2$
014710  022061  0000020                 CMP     (R0)+,FD$NAM+2(R1)
014714  001003                          BNE     2$
014716  021061  0000040                 CMP     (R0),FD$NAM+4(R1)
014722  001404                          BEQ     3$              ;BR IF FOUND ENTRY FOR FILE
014724  005761  0000000          2$:     TST     FC$LNK(R1)      ;IS THIS LAST ENTRY IN LIST?
014730  001355                          BNE     1$              ;BR IF NOT
014732  000414                          BR      4$              ;FILE ENTRY IS NOT IN CACHE TABLE
                                ;
                                ;  Found entry -- Relink at head of list (it is most recently used).
                                ;
014734  016162  0000000  0000000 3$:     MOV     FC$LNK(R1),FC$LNK(R2);RELINK LIST AROUND US
014742  013761  0000000  0000000         MOV     CSHHD,FC$LNK(R1);LINK US IN AT HEAD OF LIST
014750  010137  0000000                 MOV     R1,CSHHD
                                ;
                                ;  Return with R1 = address of cache entry, R0 = Starting block #.
                                ;
014754  016100  0000000                 MOV     FC$SBL(R1),R0   ;GET STARTING BLOCK # FOR FILE
014760  000241                          CLC                     ;SIGNAL SUCCESSFUL RETURN
014762  000401                          BR      5$
                                ;
                                ;  Can't find file entry in cache.
```

```
      58                                ;
      59 014764  000261                4$:     SEC                     ;SIGNAL UNSUCCESSFUL RETURN
      60                                ;
      61                                ;  Return
      62                                ;
      63 014766  012605                5$:     MOV     (SP)+,R5
      64 014770  012603                        MOV     (SP)+,R3
      65 014772  012602                        MOV     (SP)+,R2
      66 014774  000207                        RETURN
```

```
 1                                          .SBTTL  CSHTST -- Determine if device is to be cached
 2                                  ;-----------------------------------------------------------------------
 3                                  ;  CSHTST is called to determine if file entries for a particular
 4                                  ;  device & unit are to be stored in the directory cache.
 5                                  ;
 6                                  ;  Inputs:
 7                                  ;   FILDVU = Device & unit #
 8                                  ;
 9                                  ;  Outputs:
10                                  ;   C-flag cleared if device is to be cached.
11                                  ;   R5 = Address of device entry in cached device table.
12                                  ;
13 014776  010346                  CSHTST: MOV     R3,-(SP)
14 015000  010446                          MOV     R4,-(SP)
15                                  ;
16                                  ;  If this file is on a logical disk, set up information about the LD.
17                                  ;
18 015002  004737  015054'                  CALL    GETDVU          ;GET INFO ABOUT PHYSICAL DEVICE AND UNIT
19                                  ;
20                                  ;  Search for device information in table of cached devices
21                                  ;
22 015006  013705  0000000                  MOV     CSHDEV,R5       ;POINT TO TABLE OF MOUNTED DEVICES
23 015012  020365  0000000          2$:     CMP     R3,CD$DVU(R5)   ;DO DEVICE # AND UNIT #'S MATCH?
24 015016  001003                          BNE     5$              ;BR IF NOT
25 015020  020465  0000000                  CMP     R4,CD$BAS(R5)   ;DOES BASE BLOCK # OF DISK MATCH?
26 015024  001407                          BEQ     3$              ;BR IF YES -- THIS DEVICE IS CACHED
27 015026  062705  0000000          5$:     ADD     #CD$$SZ,R5      ;POINT TO NEXT TABLE ENTRY
28 015032  020537  0000000                  CMP     R5,CSHDVN       ;REACHED END OF TABLE?
29 015036  103765                          BLO     2$              ;BR IF NOT
30 015040  000261                          SEC                     ;SIGNAL THAT DEVICE IS NOT TO BE CACHED
31 015042  000401                          BR      4$
32 015044  000241                  3$:     CLC                     ;SIGNAL THAT DEVICE IS TO BE CACHED
33                                  ;
34                                  ;  Finished
35                                  ;
36 015046  012604                  4$:     MOV     (SP)+,R4
37 015050  012603                          MOV     (SP)+,R3
38 015052  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)
                                ;
015054  013703  002014'         GETDVU: MOV     FILDVU,R3       ;GET DEVICE # AND UNIT #
015060  005004                         CLR     R4              ;ASSUME DEVICE IS NOT A LOGICAL DISK
015062  120337  0000000                 CMPB    R3,LDDEVX       ;IS THIS DEVICE A LOGICAL DISK?
015066  001010                         BNE     4$              ;BR IF NOT
015070  000303                         SWAB    R3              ;GET LOGICAL UNIT # TO LOW ORDER BYTE
015072  042703  177770                 BIC     #^C7,R3         ;MASK OUT ALL BUT UNIT #
015076  006303                         ASL     R3              ;CONVERT TO WORD TABLE INDEX
015100  016304  0000000                 MOV     LDBASE(R3),R4   ;GET STARTING POSITION OF LOGICAL DISK
015104  016303  0000000                 MOV     LDPDEV(R3),R3   ;GET PHYSICAL DEVICE # AND UNIT #
                                ;
                                ;  Finished
                                ;
015110  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.
                                ;
015112  006203                  CDJFLG: ASR     R3              ;Convert job index to index by 1
015114  005303                          DEC     R3              ;Make base job number 0
015116  005002                          CLR     R2              ;Clear for divide
015120  071227  000010                  DIV     #8.,R2          ;Divide by 8 jobs per byte
015124  060502                          ADD     R5,R2           ;Get address of byte within
015126  062702  0000000                 ADD     #CD$JOB,R2      ; cached-device table entry
015132  012700  000001                  MOV     #1,R0           ;Get a mount flag
015136  072003                          ASH     R3,R0           ;Position flag according to job number
015140  010003                          MOV     R0,R3           ;Return flag in R3
015142  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]
                                ;
015144  013703  0000000G        USRCLS: MOV     CHNADR,R3       ;GET ADDRESS OF CURRENT CHANNEL
015150  001402                          BEQ     USRERR          ;BR IF NONE
015152  005063  0000000G                CLR     C.CSW(R3)       ;PURGE THE CHANNEL
015156  010046                  USRERR: MOV     R0,-(SP)        ;SAVE ERROR CODE
015160  004737  010132'                 CALL    FREUSR          ;FREE USR DATA BASE IF WE HAVE IT
015164  012600                          MOV     (SP)+,R0        ;GET BACK ERROR CODE
015166  000137  0000000G                JMP     SETERR          ;RETURN ERROR CODE FOR EMT
                                ;
                                ;  Normal exit from USR
                                ;
015172  004737  010132'         USRXIT: CALL    FREUSR          ;FREE USR MODULE
015176  000137  0000000G                JMP     EMTXIT          ;EXIT FROM EMT
                                ;
                                ;  Fatal abort due to internal consistency check.
                                ;
                                ;  Inputs:
                                ;   R0 = Abort code.
                                ;
015202  010005                  UABORT: MOV     R0,R5           ;GET ERROR CODE FOR ABORT
015204  013704  0000000G                MOV     EMTADR,R4       ;GET ADDRESS OF ABORTED EMT
015210  000137  0000000G                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.
                                ;
015214                          ERRNAM:
                                ;
                                ;  Don't set file spec for error message if a .SERR has been done
                                ;
015214  105737  0000000                 TSTB    SERFLG                  ;Has a .SERR been done?
015220  001014                          BNE     9$                      ;Br if yes
                                ;
                                ;  Move file spec to holding area
                                ;
015222  013737  0000000 0000000         MOV     FILSPC,ERRSPC           ;Device name
015230  013737  0000020 0000020         MOV     FILSPC+2,ERRSPC+2       ;File name part 1
015236  013737  0000040 0000040         MOV     FILSPC+4,ERRSPC+4       ;File name part 2
015244  013737  0000060 0000060         MOV     FILSPC+6,ERRSPC+6       ;Extension
                                ;
                                ;  Finished
                                ;
015252  000207                  9$:     RETURN
                                ;
                                ;  Top of TSUSR module
                                ;
015254                          USRTOP:
        000001                          .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  10528 Words  ( 42 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:01:28.47
DK:TSUSR,LP:TSUSR=DK:TSUSR.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

```
$DILUP    1-55      40-39
$DUPRN    1-71      34-130
$KINIT    1-41      19-77
$NOIN     1-62      34-49
$UCLRN    1-47      34-41
...V1    47-23      47-23     48-14     48-14     49-84     49-84
...V2    47-23      47-23     47-23#    47-23#    48-14     48-14     48-14#    48-14#    49-84     49-84#
ABORT     1-63      57-30
AD$$SZ    1-51      14-41     14-60     15-21     15-34     17-41     33-24
AD$DVU    1-51      14-39     14-68*    15-20*    15-32     15-45*    17-39     33-22
AD$FLG    1-51      14-70*
AD$JOB    1-51      14-48     14-51*    14-58     14-69*    15-17     15-19*    15-38     15-46*    17-61     33-41
ADDENT    5-55      37-16#
AF$BYA    1-39      29-26
ALCCOM   14-27      15-9      16-6      18-11#
ALCEMT    1-25      14-6#
ALCEND    1-51      14-42     14-61     15-22     15-35     17-42     33-25
ALCTBL    1-51      14-37     14-57     15-10     17-38     33-20
ALCTST   14-31      16-10     17-16#
ASNEND    1-33      29-32
ASNSIZ    2-48#      5-42     29-15*    29-50*
ASNTBL    1-32      29-28
AT$$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*    34-71*    34-146*   40-43     40-45     49-49     49-104*   49-111*   57-12*
C.DEVQ    1-40       6-63      8-50      19-96     27-18     28-60*    49-59
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$$SZ    1-72      19-62     20-35     22-27     26-25     54-27
CD$$UB    1-73      20-66     24-21
CD$BAS    1-72      20-49*    24-29     54-25
CD$DVU    1-72      19-32     19-34     20-33     20-48*    22-20     24-31     24-43     24-48*    54-23
CD$JOB    1-73      20-65     24-20     56-19
CD$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     56-14#
CHKABT    1-36      31-25     50-17
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     57-10
CHNNUM    1-39       8-67     37-49     47-23     47-23     48-14     48-14     49-76     49-84     49-84
CHNSIZ    1-60
CKADEV    2-52#     34-82*    34-91     34-119    34-132
```

Cross reference table (CREF V05.04)

```
CKAUNT    2-53#    34-81*   34-93    34-134
CL$LIX    1-46     32-83
CLCLOS    1-69
CLDEVX    1-62     18-28    32-77
CLOSE     1-24      8-5#
CLRDIR    1-25     27-12#
CLTOTL    1-56     18-30    18-35
CONSOL    6-117     6-144    7-56     8-100    8-136    36-31    36-152   40-12#
CORUSR    1-33      8-73    20-72    22-13    24-13    26-22    31-14    31-26    31-33    31-58    33-42    34-22
         34-129    37-48    49-60    50-9     50-22    50-30
CPLEMT    1-49     44-42
CPYMNT    1-27     26-9#
CPYSPC   29-20     30-10#   49-30
CS$ENT    1-41      5-30     5-66     8-40    40-45
CS$EOF    1-45      8-58
CS$ERR    1-45      8-58
CS$NMX    1-41      8-24     8-52
CS$OPN    1-42      8-6     19-87    28-35    28-62    40-43
CS$RON    1-45      5-10     6-16     6-51     6-68     7-9     13-45    34-32    34-71    34-146
CS$SPL    1-42      8-16
CSHADD    4-56      6-140    8-134    11-27    51-9#
CSHALC    1-42     24-41
CSHCHK    4-42     52-14    53-17#
CSHCLN    1-42     24-44
CSHDEL    6-34      6-107    7-48     8-92     13-32    52-9#
CSHDEV    1-64     19-31    20-32    22-19    26-17    54-22
CSHDMT   21-12     21-49#
CSHDVN    1-64     19-63    20-36    22-28    26-26    54-28
CSHHD     1-63      3-15*   21-50    24-52    27-29    51-22    51-33    51-34*   53-29    53-48    53-49*
CSHTST   20-13     23-12    27-24    51-16    53-24    54-13#
CURCP     1-61     31-39
CURSEG    2-31#     5-64     6-29     6-82     6-112    8-85     8-96     31-46*   36-77    36-88    36-94    36-147
         39-27     39-80*   47-14    47-18*   48-10
DELETE    1-24      7-5#
DEVSIZ    1-59      9-30
DF$CLS    1-68      8-29
DF$DEL    1-68      7-21
DF$ENT    1-68      5-31    49-53    49-102
DF$LOK    1-68      4-24    49-100
DH$$BS    1-53     46-36    47-21    48-13
DH$$SZ    1-53     37-72    39-34    39-44    39-63    39-84    40-23    41-5     41-39    45-14    46-41    47-29
DH$BLK    1-52     39-43    39-81*   45-13    46-36
DH$HIS    1-52     39-96*   46-26
DH$NEB    1-52     46-31
DH$NSG    1-52     46-27
DH$NXT    1-52     36-100   39-70    39-73*   39-79*   42-27
DIDDLE    4-72      5-74     6-136   44-18#
DIRHIS    2-32#    39-25    39-71    39-97*   46-26*
DIRNSG    2-34#    39-25    46-27*   46-29
DIRSIZ    2-33#     8-119    8-124   36-35    37-27    37-61    37-70    37-71    37-76    37-78    38-27    38-34
         39-39     39-46    39-52    40-24    43-14    45-18    46-35*   47-30
DISMNT    1-27     21-5#
DLCEMT   14-13     15-5#
DMTALL   21-14     22-5#
DMTDEV   22-26     23-17    24-8#
DMTSUB   20-9      21-28    23-8#
```

Cross reference table (CREF V05.04)

```
DS$DIR    1-47    4-29    5-36    6-9     7-26    13-18   20-21   34-65
DS$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
          34-65
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    57-29
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   44-18   44-53   49-26   49-52   49-55   49-57
EMTCAD    1-49    44-37   44-39*
EMTPS     1-59    1-71    30-22   44-52
EMTXIT    1-54    8-149   49-110  57-21
ENTER     1-24    5-5#
ERRNAM    5-12    6-44    6-70    7-11    13-47   25-26   28-50   33-34   33-45   34-139  47-25   48-16
          58-13#
ERRSPC    1-45    58-22*  58-23*  58-24*  58-25*
EXCJOB    1-31    31-21   31-22*  31-28*
FC$$SZ    1-56    3-18    3-21    3-27
FC$CDX    1-37    3-24*   21-51*  24-53   24-55*  27-30   27-32*  51-25   51-47*  52-20*  53-32
FC$LNK    1-63    3-23*   3-28*   21-53   24-57   27-34   51-22   51-24   51-27   51-32   51-32*  51-33*
          53-29   53-31   53-41   53-47   53-47*  53-48*
FC$SBL    1-64    51-52*  53-53
FD$$SZ    1-64    44-30   44-62   51-40
FD$CHN    1-48    8-71    37-49*  40-41
FD$DAT    1-49    4-68    8-126   8-128*  11-10   11-11*  12-23   37-50*  44-51
FD$JOB    1-48    8-73    37-48*  40-33
FD$LEN    1-48    4-64    8-111   8-115*  8-122*  12-14   36-63   37-43   37-44*  37-63*  39-45   39-51
          41-17   41-17*  41-35   45-17
FD$NAM    1-48    6-128   8-78    21-52*  24-56*  27-33*  35-29   36-50   37-52   52-13   53-35   53-37
          53-39
FD$OPT    1-49    46-34
FD$STA    1-48    6-80*   6-81*   6-93    6-100*  6-101*  6-109*  6-125*  6-126*  7-41    7-52*   8-94*
          8-108   8-110*  8-120   11-57*  11-59*  12-18   36-36   37-47*  37-74   38-19   38-39*  39-36
          39-49   40-48*  41-14   41-41   43-15   43-17
FD$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   55-14
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   34-63   34-97   35-30   36-51   37-53   58-22   58-23   58-24   58-25
FNDFIL    4-48    6-22    6-87    7-36    8-89    13-25   35-15#
FNDFRE    5-45    36-17#
FRECXT    1-57    19-125  19-132  41-59
FRESPD    1-27    49-96   50-30#
FREUSR    1-25    8-141   31-57#  49-17   57-14   57-20
FS$EMP    1-50    6-81    6-100   6-109   6-125   7-52    8-94    8-120   36-37   36-39   38-39   40-48
          41-7    41-14
FS$EOS    1-50    36-37   36-41   38-18   39-36   39-69   41-20   41-46   43-17
FS$PRM    1-50    6-80    6-101   6-126   8-110   35-24   36-37   39-49   41-41
FS$PRO    1-61    6-93    7-41    11-57   11-59   12-18   36-47
FS$TEN    1-50    8-68    8-108   37-47   37-74   39-49   40-27
```

Cross reference table (CREF V05.04)

```
GETCHA    1-55      19-83     40-42
GETCXT    1-57      19-24     40-19
GETDIR    8-69      35-25     42-17#
GETDVU   20-44      54-18     55-14#
GETQ      1-70      49-37
GETSPC    6-41       9-8      10-12     18-20     25-20     28-44     29-13#
GETSPD   49-21      50-7#     50-18
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   37-62      37-77     38-14#
INTPRI    1-69      31-20     31-34     31-66     31-73
IOWAIT    1-57       8-12
JCDB      1-38       8-34
KPAR6     1-33      49-39
L1OFF     2-43#     36-76*    36-85     36-87*    36-117    36-133
L1SEG     2-44#     36-77*    36-84     36-88*    36-116    36-132
L1SIZ     2-42#     36-22*    36-71     36-75*    36-81     36-83     36-86*    36-108    36-115    36-121
L2OFF     2-46#     36-85*    36-93*    36-129
L2SEG     2-47#     36-84*    36-94*    36-128
L2SIZ     2-45#     36-23*    36-83*    36-90     36-92*    36-126    36-130
LD$RON    1-53      34-30
LDAEMT    1-33      21-44
LDBASE    1-72      55-21
LDDEMT    1-32      21-36
LDDEVX    1-62       9-35     18-38     32-65     34-26     55-16
LDFLAG    1-53      34-30
LDIEMT    1-32      21-40
LDNAME    1-73      20-56
LDPDEV    1-69      32-69     55-22
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    40-37
LSW       1-55      19-77     40-39
LSW2      1-43
LSW3      1-62      34-49
LSW5      1-45
LSW6      1-43      34-130
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   37-24      38-21     40-28     41-8      41-21     41-47     42-21     43-14#    43-18
ODTBAS    1-71      30-24
OF$$SZ    1-44      34-110
OF$DEV    1-43      34-88
OF$FIL    1-44      34-96
OF$FLG    1-44      34-105
OF$UNT    1-44      34-93
```

Cross reference table (CREF V05.04)

```
OKFEND    1-45      34-111
OKFILE    1-43      34-84
OT$RON    1-44      34-105
OVRHC     1-69       4-14      5-20      8-18      19-24     19-83     19-125    19-132    40-19     40-42     41-59
PO$ALC    1-35      14-20     15-5
PO$BYP    1-35      34-36
PO$NFR    1-35      34-69
PO$NFW    1-35      34-67
PO$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     34-67     34-69
PSW       1-69      31-11*    31-20*    31-34*    31-57*    31-66*    31-73*
Q.BLKN    1-66      49-56*
Q.BUFF    1-66      49-73*
Q.CHAN    1-67      49-76*
Q.COMP    1-66      49-75*
Q.CSW     1-65      49-42*
Q.DEVX    1-65      49-51*
Q.FUNC    1-65      49-58*
Q.ICSW    1-67      49-41
Q.JNUM    1-64      49-65*
Q.JOB     1-65      49-62*
Q.PA6     1-68      49-39*
Q.PAR     1-66      49-74*
Q.UCSW    1-67      49-38*
Q.UNIT    1-65      49-59*
Q.WCNT    1-66      49-57*
QIO       1-70      49-80
QNSPNX    1-38      31-24     50-16
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     36-102    36-150    39-107    42-29     46-21     47-14#
RDSEG1    8-59      35-20     36-27     39-95     46-16#
RENAME    1-24       6-5#
RESDEV    1-43      34-75
RUNFLG    1-39      29-26
S$QSPD    1-67      50-15     50-36
S$QUSR    1-33      31-23     31-67
SBCALC    5-69      12-30     35-38     45-12#    51-51
SDCLOS    1-47       8-18
SERFLG    1-41      58-17
SETERR    1-54      14-23     15-8      49-112    57-16
SFCLS     1-47       8-36
SFCOM    11-5       11-41     11-49     12-10     13-13#
SFDATE    1-26      11-5#
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      50-7      50-22*    50-30     50-32*
SPFLDV    1-70      49-29
```

Cross reference table (CREF V05.04)

```
SPFLNM    1-70      49-66
SPLDIR    4-25       5-32      7-22      8-30      49-13#
SPLIT    37-37      39-17#
SPSIZE    1-70      49-32*    49-91
SPUSR     1-67      49-33*    49-87*    49-92
SYINDX    1-39      17-29     24-32     32-41     34-51     34-132
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     34-134
TK3SVL    1-37       8-131
TLCEMT   14-15      16-5#
TSUSR     1-15#      1-23
UABORT    8-167     57-28#
UACHKW    1-58      30-15
UERR1     2-58#      8-154
UERR2     2-59#      8-157
UERR3     2-60#      8-160
UERR4     2-61#      8-163
UERR5     2-62#      8-166
UERR6     2-63#     46-50
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*    49-91*
UREGO     1-36      31-68     50-37
USRBUF    2-30#     36-100    37-31     37-72     39-34     39-43     39-44     39-63     39-70     39-73*    39-79*    39-81*
         39-84      39-85     39-96*    40-23     41-5      41-39     42-27     45-13     45-14     46-25     47-23     47-29
         48-14
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     34-141    46-51     47-27
         48-18      57-10#
USRCNT    2-41#     31-35*    31-60*
USRCOM    4-5        5-5       6-5       7-5      13-13     28-26#
USREMT    1-63      47-23     48-14
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      57-11     57-13#
USRINI    1-23       3-6#
USRJOB    1-70      31-12     31-33*    31-58     31-62*
USRTOP    1-23       3-11     58-33#
USRUCA    1-25       2-26#    44-29     44-51     44-61
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     57-20#
VALADB    1-58
VALADW    1-58       9-23     12-12
VMXFIL    1-47      36-138    36-140    36-142
VNFCSH    1-50       3-16
VPAR6     1-66      49-72
WLDNAM    1-33      34-101
WRTSEG    5-78       6-84      6-102     6-118     6-145     7-57      8-101     8-137     11-31     39-74     39-91     39-98
         48-9#
```

Cross reference table (CREF V05.04)

```
...CM1   47-23     48-14
...CM2   47-23     47-23     47-23     47-23     48-14     48-14     48-14     48-14
...CM5   47-23     48-14
...CM7   47-23     48-14
.READW    1-19#    47-23
.WAIT     1-19#    49-84
.WRITW    1-19#    48-14
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    40-19     40-42     41-59
```