

3-	1	USRINI -- Initialization for USR module
4-	1	.LOOKUP
5-	1	.ENTER
6-	1	.RENAME
7-	1	.DELETE
8-	1	.CLOSE
9-	1	.DSTATUS
10-	1	.FETCH
11-	1	.SFDAT, .SFTIM, & .FPROT
12-	1	.GFINF -- Get information about a file
14-	1	ALCEMT -- Allocate a device
15-	1	DLCEMT -- Deallocate a device
16-	1	TLCEMT -- Check to see if a device is allocated
17-	1	ALCTST -- See if device is allocated to another user
18-	1	ALCCOM -- Common setup for Allocate/Deallocate
19-	1	CHKUSE -- See if any channels open to a specified device
20-	1	MOUNT -- Mount a new file structure
21-	1	DISMNT -- Dismount a file structure
22-	1	DMTALL -- Dismount all mounted devices for job
24-	1	DMTDEV -- Remove entry from cache table
25-	1	MNTCOM -- MOUNT/DISMOUNT common setup
26-	1	CPYMNT -- Copy mount entries from another job
27-	1	CLRDIR -- Remove directory entries from dir cache
28-	1	USRCOM -- Common setup
29-	1	GETSPC -- Get file spec from user's area
30-	1	CPYSPC -- Copy file specification from user's area
31-	1	GETUSR -- Claim usr data base for our job
31-	53	FREUSR -- Free the USR
32-	1	CHKDEV -- See if requested device is legal
33-	1	CHKALC -- Check for device allocation
34-	1	CHKACC -- Check legality of file access
35-	1	FNDFIL -- Find file in directory
36-	1	FNDFRE -- Find a free slot for a new file
37-	1	ADDENT -- Add a tentative file entry to directory
38-	1	INSERT -- Add an empty file entry to directory
39-	1	SPLIT -- Split a directory segment
40-	1	CONSOL -- Directory segment consolidator
42-	1	GETDIR -- Locate next directory entry
43-	1	NXTDIR -- Locate next directory entry in current segment
44-	1	DIDDLE -- Allow user to access directory entry
45-	1	SBCALC -- Calculate starting block # for a file entry
46-	1	RDSEGI -- Read 1st directory segment
47-	1	RDSEG -- Read a directory segment
48-	1	WRTSEG -- Write directory segment
49-	1	SPLDIR -- Directory operations for special devices
50-	1	GETSPD -- Gain exclusive access to special device data base
50-	25	FRESPD -- Free special device data base
51-	1	CSHADD -- Add file entry to directory cache
52-	1	CSHDEL -- Remove a file entry from the directory cache
53-	1	CSHCHK -- Search for file entry in directory cache
54-	1	CSHTST -- Determine if device is to be cached
55-	1	GETDVU -- Get device # & unit # info
56-	1	CDJFLG -- Get user-flag for cached device entry
57-	1	USRXIT -- Exit from USR
58-	1	ERRNAM -- Set name of file spec for error message

.LOOKUP

```

58 ;
59 ; Found file. Save info about file in channel.
60 ; (R1 now points to directory entry for file).
61 ; (R0 Contains starting block number of the file).
62 ;
63 002300 010063 0000000 2$: MOV R0,C.SBLK(R3) ;SET FILE STARTING BLOCK #
64 002304 016100 0000000 MOV FD$LEN(R1),R0 ;GET ALLOCATED SIZE OF FILE
65 002310 010063 0000000 MOV R0,C.LENG(R3) ;STORE IN CHANNEL BLOCK
66 002314 010037 0000000 MOV R0,URO ;RETURN TO USER IN R0
67 002320 016137 0000000 0000000 MOV FD$TIM(R1),LSTFDT;SAVE TIME ENTRY FROM EACH LOOKUP
68 002326 016137 0000000 0000000 MOV FD$DAT(R1),LSTFDD;ALSO SAVE DATE ENTRY
69 ;
70 ; Allow user to diddle with directory entry if he wants to.
71 ;
72 002334 004737 013254' CALL DIDDLE ;ALLOW DIRECTORY ENTRY DIDDLE
73 ;
74 ; Finished
75 ;
76 002340 000137 015172' 3$: JMP USRXIT ;EXIT FROM USR PROCESSING

```

```

1          .TITLE  TSUSR  -- File operation EMT's
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  CBL
5          ;
6          ; TSUSR is the TSX module that contains the processing routines for
7          ; EMTs that perform file operations such as .lookup, .enter, .delete,
8          ; .rename, etc.
9          ;
10         ; Copyright (c) 1980, 1981, 1982, 1983, 1984, 1985.
11         ; S&H Computer Systems, Inc.  Nashville, TN
12         ; All rights reserved.
13         ;
14 000000          .CSECT  TSUSR
15 000000 103112  TSUSR: .RAD50  /USR/
16         ;
17         ; Macro calls
18         ;
19         .MCALL  .READW, .WRITW, .WAIT
20         ;
21         ; Global definitions
22         ;
23         .GLOBL  TSUSR, GETUSR, USRTOP, USRINI
24         .GLOBL  LOOKUP, ENTER, CLOSE, DELETE, RENAME
25         .GLOBL  DSTAT, FREUSR, USRUCA, ALCEMT, CLRDIR
26         .GLOBL  SFTIME, SFDATE, SFPROT, GFINFO
27         .GLOBL  MOUNT, DISMNT, FRESPD, FETCH, CPYMNT
28         ;
29         ; Global references
30         ;
31         .GLOBL  EXCJOB
32         .GLOBL  DVSTAT, CHNADR, FILSPC, EMTBLK, ASNTBL, LDDEMT, LDIEMT
33         .GLOBL  LDAEMT, ASNEND, CORUSR, S#QUSR, MAXLD, KPAR6, WLDNAM
34         .GLOBL  AT#LOG, AT#SIZ, AT#DEV, AT#FIL, AT#EXT, AT##SZ
35         .GLOBL  PO#ALC, PRIVCO, PO#BYP, PO#NFR, PO#NFW
36         .GLOBL  CHKABT, UREGO, NUMDEV, LSTFDT, LSTFDD
37         .GLOBL  TK3SVL, FD#TIM, SYTIMH, SYTIML, FC#CDX
38         .GLOBL  QNSPNX, JCDB, DVFLAG, DX#RAL, DX#NMT
39         .GLOBL  PNAME, CHNNUM, SYINDX, SYUNIT, RUNFLG, AF#BYA
40         .GLOBL  C. CSW, C. SBLK, C. LENG, C. USED, C. NUMQ, C. DEVO
41         .GLOBL  CS#NMX, CS#ENT, SERFLG, #KINIT, NLCHN
42         .GLOBL  CS#OPN, CHKSD, CS#SPL, CSHCLN, CSHALC
43         .GLOBL  LSW2, RESDEV, OKFILE, OF#DEV, LSW6
44         .GLOBL  OF#UNT, OT#RON, OF#FLG, OF#FIL, OF##SZ
45         .GLOBL  OKFEND, LSW5, CS#RON, ERRSPC, CS#EOF, CS#ERR
46         .GLOBL  DS#NRD, CL#LIX, PRIVCO, PO#SYS, DX#NRD
47         .GLOBL  DS#DIR, SDCLOS, SFCLS, VMXFIL, #UCLR, LSW7
48         .GLOBL  FD#STA, FD#NAM, FD#LEN, FD#JOB, FD#CHN
49         .GLOBL  FD#DAT, FD#OPT, CPLEMT, EMTCAD
50         .GLOBL  FS#TEN, FS#EMP, FS#PRM, FS#EOS, VNFCSH
51         .GLOBL  ALCTBL, ALCEND, AD#DVU, AD#JOB, AD#FLG, AD##SZ
52         .GLOBL  DH#NSG, DH#NXT, DH#HIS, DH#NEB, DH#BLK
53         .GLOBL  DH##SZ, DH##BS, LDFLAG, LD#RON
54         .GLOBL  EMTXIT, SETERR, URO, SYSCHN
55         .GLOBL  SYSDAT, GETCHA, LSW, #DILUP, LDSIZE
56         .GLOBL  FC##SZ, BADEMT, LNPRIM, CLTOTL
57         .GLOBL  IOWAIT, GETOXT, FREEXT

```

```

58 .GLOBL UACHKW, VALADB, VALADW
59 .GLOBL HANSIZ, HANENT, DEVSIZ, EMTPS
60 .GLOBL SYSCHN, LSTGL, CHNSIZ
61 .GLOBL CURCP, R$CHN, R$XCHN, FS$PRO
62 .GLOBL $NOIN, LSW3, LDDEVX, CLDEVX, C1DEVX
63 .GLOBL ABORT, EMTADR, USREMT, CSHHD, FC$LNK
64 .GLOBL FD$$SZ, FC$SBL, CSHDEV, CSHDVN, Q. JNUM
65 .GLOBL Q. CSW, Q. FUNC, Q. UNIT, Q. JOB, Q. DEVX
66 .GLOBL VPAR6, Q. BUFF, Q. PAR, Q. WCNT, Q. COMP, Q. BLKN
67 .GLOBL Q. CHAN, SPUSR, S$QSPD, Q. UCSW, Q. ICSW
68 .GLOBL DF$LOK, DF$ENT, DF$DEL, DF$CLS, Q. PA6
69 .GLOBL OVRHC, INTPRI, PR7, PSW, LDPDEV, CLCLOS
70 .GLOBL USRJOB, SPDJOB, SPSIZE, SPFLDV, SPFLNM, GETQ, QID
71 .GLOBL $DUPRN, UMODE, EMTPS, ODTBAS
72 .GLOBL CD$DVU, CD$BAS, CD$$SZ, LDBASE, CD$TOP
73 .GLOBL CD$JOB, CD$$UB, CD$NAM, LDNAME, NSPLDV

```

```

1      ;
2      ; Macro definitions.
3      ;
4      .MACRO  DISABL          ;Disable interrupt processing
5      BIS    #PR7,@#PSW
6      .ENDM  DISABLE
7
8      .MACRO  ENABL          ;Enable interrupt processing
9      BIC    INTPRI,@#PSW
10     .ENDM  ENABL
11
12     ;
13     ; Macro definition for calling global routines residing in mapped system regions.
14     ;
15     .MACRO  OCALL  ENTADD
16     .IF    B,ENTADD
17     .ERROR ;OCALL SPECIFIED WITH NO ENTRY ADDRESS
18     .MEXIT
19     .ENDC
20     CALL  OVRHC          ;CALL THE OVERLAY HANDLER
21     .WORD ENTADD        ;SPECIFY THE ENTRY POINT
22     .ENDM
23
24     ; Assembly parameters.
25     ;
26     000300  USRUCA =      300          ;USR-User communication area
27     ;
28     ; Data areas
29     ;
30     000002  USRBUF: .BLKW  512.        ;Buffer to hold a directory segment
31     002002  177777  CURSEG: .WORD  -1    ;Number of segment currently in USRBUF
32     002004  000000  DIRHIS: .WORD  0     ;Highest segment # used in directory
33     002006  000000  DIRSIZ: .WORD  0     ;Number of bytes per directory entry
34     002010  000030  DIRNSG: .WORD  0     ;Number of directory segments available
35     002012  000000  FILBLK: .WORD  0     ;Block number of start of file
36     002014  000000  FILDVU: .WORD  0     ;Dev # in low byte, unit # in high byte
37     002016  075250  R50SY:  .RAD50  /SY/
38     002020  015270  R50DK:  .RAD50  /DK/
39     002022  075273  R50SYS: .RAD50  /SYS/
40     002024  100020  R50TSX: .RAD50  /TSX/
41     002026  177777  USRCNT: .WORD  -1    ;# of times we have claimed usr
42     002030  000000  L1SIZ:  .WORD  0     ;Size of largest free slot found
43     002032  000000  L1OFF:  .WORD  0     ;Address of entry in dir segment buffer
44     002034  000000  L1SEG:  .WORD  0     ;# of directory segment with largest slot
45     002036  000000  L2SIZ:  .WORD  0     ;Size of second largest free slot found
46     002040  000000  L2OFF:  .WORD  0     ;Address of entry in dir segment buffer
47     002042  000000  L2SEG:  .WORD  0     ;# of dir segment with 2nd largest slot
48     002044  000000  ASNSIZ: .WORD  0     ;Size specified for file with ASSIGN
49     ;
50     ; Byte data
51     ;
52     002046  000  CKADEV: .BYTE  0
53     002047  000  CKAUNT: .BYTE  0
54     ;
55     ; Internal USR error checks.
56     ; (Numbers correspond to KMON error table offsets)
57     ;

```

58	177762	UERR1	=	-16	; Can't find tentative file entry for file on close
59	177763	UERR2	=	-17	; File length in chan block not = to len in file entry
60	177760	UERR3	=	-20	; Highest block # written > file length
61	177757	UERR4	=	-21	; Empty file entry doesn't follow tentative file entry
62	177756	UERR5	=	-22	; Not tentative file entry during close
63	177755	UERR6	=	-23	; Illegal or uninitialized directory
64				. EVEN	

USRINI -- Initialization for USR module

```

1          .SBTTL  USRINI -- Initialization for USR module
2          ;-----
3          ; USRINI is called during system initialization to perform data table
4          ; allocation for the TSUSR module.
5          ;
6 002050 010246 USRINI: MOV     R2,-(SP)
7 002052 010346         MOV     R3,-(SP)
8          ;
9          ; Allocate tables within this overlay segment above top of code
10         ;
11 002054 012702 015254'         MOV     #USRTOP,R2      ;Point above top of code in this segment
12         ;
13         ; Allocate file directory cache table
14         ;
15 002060 010237 0000000        MOV     R2,CSHHD      ;Set pointer to 1st free directory cache entry
16 002064 013700 0000000        MOV     VNFCSH,R0     ;Get # cache entries to allocate
17 002070 010203 1#           1#: MOV     R2,R3          ;Get address of current entry
18 002072 062703 0000000        ADD     #FC##SZ,R3     ;Get address of next entry
19 002076 020327 140000        CMP     R3,#140000    ;Will current entry fit in overlay region?
20 002102 101403 2#           BLOS    2#              ;Br if yes
21 002104 162703 0000000        SUB     #FC##SZ,R3     ;No -- Backup pointer to current entry
22 002110 000406 3#           BR      3#
23 002112 010362 0000000        2#: MOV     R3,FC$LNK(R2) ;Make current entry point to next one
24 002116 005062 0000000        CLR     FC$CDX(R2)   ;Say entry is currently free
25 002122 010302 3#           MOV     R3,R2          ;Make next entry be current one
26 002124 077017 1#           SOB     R0,1#         ;Loop if more to allocate
27 002126 162703 0000000        3#: SUB     #FC##SZ,R3     ;Point to last entry
28 002132 005063 0000000        CLR     FC$LNK(R3)   ;Make its forward link zero
29         ;
30         ; Finished
31         ;
32 002136 012603 3#           MOV     (SP)+,R3
33 002140 012602 2#           MOV     (SP)+,R2
34 002142 000207 1#           RETURN

```

.ENTER

```

1          .SBTTL .ENTER
2          ;-----
3          ; Open a new file.
4          ;
5 002344 004737 007362' ENTER: CALL USRCOM ;DO COMMON SETUP
6 002350 005037 0000000 CLR URO ;FOR NOW, SAY FILE SIZE = 0
7          ;
8          ; See if we have write access to the file.
9          ;
10 002354 032763 0000000 0000000 BIT #CS$RD, C. CSW(R3); DO WE HAVE WRITE ACCESS TO THE FILE?
11 002362 001406 BEQ 4$ ;BR IF YES
12 002364 004737 015214' CALL ERRNAM ;SET NAME OF FILE FOR TSKMON ERROR MESSAGE
13 002370 012700 177763 MOV #-15, R0 ;SET ABORT CODE
14 002374 000137 015144' JMP USRCLS ;ABORT OPERATION
15          ;
16          ; See if it is a spooled device.
17          ;
18 002400 105737 0000000 4$: TSTB NSPLDV ;Are there any spooled devices?
19 002404 001404 BEQ 3$ ;Br if not
20 002406 DCALL CHKSD ;Are we opening to a spooled device?
21 002414 103074 BCC 9$ ;Br if this is a spooled device
22          ;
23          ; This is not a spooled device.
24          ; See if this is a magnetic tape.
25          ;
26 002416 032764 0000000 0000000 3$: BIT #DS$NRD, DVSTAT(R4); MAG TAPE TYPE DIRECTORY DEVICE?
27 002424 001004 BNE 7$ ;Br if yes
28 002426 032764 0000000 0000000 BIT #DX$NRD, DVFLAG(R4); Internal flag set?
29 002434 001407 BEQ 5$ ;BR IF NOT
30 002436 052763 0000000 0000000 7$: BIS #CS$ENT, C. CSW(R3); SET FLAG TO CAUSE TO CALL HANDLER ON CLOSE
31 002444 012702 0000000 MOV #DF$ENT, R2 ;GET CODE FOR ENTER
32 002450 000137 014010' JMP SPLDIR ;GO DO SPECIAL DEVICE ENTER
33          ;
34          ; See if this is a file structured device.
35          ;
36 002454 032764 0000000 0000000 5$: BIT #DS$DIR, DVSTAT(R4); FILE STRUCTURED DEVICE?
37 002462 001451 BEQ 9$ ;Br if not file structured
38          ;
39          ; This is a file structured device.
40          ; Look for a free slot large enough for the file.
41          ;
42 002464 013702 002044' MOV ASNSIZ, R2 ;WAS SIZE SPECIFIED WITH ASSIGN STATEMENT?
43 002470 001002 BNE 6$ ;BR IF YES
44 002472 013702 0000040 MOV EMTBLK+4, R2 ;GET REQUESTED SIZE FROM EMT ARG BLOCK
45 002476 004737 011324' 6$: CALL FNDFRE ;TRY TO FIND A FREE SLOT
46 002502 103002 BCC 1$ ;BR IF FOUND ONE
47          ; Error: Can't find a free slot or protected file exists with same name.
48          ; (Error code is returned in R0 by FNDFRE.)
49 002504 000137 015144' JMP USRCLS
50          ;
51          ; We have found a free slot.
52          ; Create a tentative file entry.
53          ;
54 002510 010237 0000000 1$: MOV R2, URO ;RETURN FILE SIZE IN USER'S R0
55 002514 004737 012026' CALL ADDENT ;CREATE A TENTATIVE FILE ENTRY
56 002520 103004 BCC 2$ ;BR IF WE WERE SUCCESSFUL
57          ; 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

```

.RENAME

```

1          .SBTTL .RENAME
2          ;-----
3          ; Rename a file.
4          ;
5 002612  004737  007362'  RENAME: CALL  USRCOM          ; DO COMMON SETUP
6          ;
7          ; See if this is a file structured device.
8          ;
9 002616  032764  0000000 0000000  BIT    #DS#DIR,DVSTAT(R4);FILE STRUCTURED DEVICE?
10 002624  001004          BNE    1$          ;BR IF YES
11 002626  012700  000002  MOV    #2,R0          ;INVALID OPERATION
12 002632  000137  015144'  JMP    USRCLS
13          ;
14          ; Make sure we have write access to old file.
15          ;
16 002636  032763  0000000 0000000 1$: BIT    #CS#RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO OLD FILE?
17 002644  001063          BNE    11$          ;BR IF NOT
18          ;
19          ; Locate directory entry with old name.
20          ; (Note: FILSPC currently contains old file name)
21          ;
22 002646  004737  011242'  CALL   FNDFIL          ;LOOK UP THE ENTRY
23 002652  103004          BCC    2$          ;BR IF FOUND
24 002654  012700  000001  MOV    #1,R0          ;FILE NOT FOUND
25 002660  000137  015144'  JMP    USRCLS
26          ; Save info about old file entry.
27 002664  010146  2$: MOV    R1,-(SP)      ;ADDRESS OF ITS DIRECTORY ENTRY
28 002666  010105          MOV    R1,R5          ;CARRY ADDR OF DIR ENTRY IN R5 FOR A WHILE
29 002670  013746  002002'  MOV    CURSEG,-(SP)   ;SEGMENT NUMBER CONTAINING ENTRY
30 002674  010446          MOV    R4,-(SP)      ;DEVICE TABLE INDEX
31          ;
32          ; Remove old file entry from directory cache.
33          ;
34 002676  004737  014616'  CALL   CSHDEL          ;DELETE FILE ENTRY FROM CACHE TABLE
35          ;
36          ; Set up new file spec.
37          ;
38 002702  013701  0000020  MOV    EMTBLK+2,R1    ;ADDRESS OF FILE SPEC AREA
39 002706  042701  000001  BIC    #1,R1          ;MAKE SURE ADDRESS IS EVEN
40 002712  062701  000010  ADD    #10,R1         ;POINT TO NEW FILE SPEC
41 002716  004737  007532'  CALL   GETSPC         ;MOVE DEVICE-FILE SPEC TO FILSPC
42 002722  004737  010212'  CALL   CHKDEV         ;MAKE SURE DEVICE IS LEGAL
43 002726  103006          BCC    3$          ;BR IF OK
44 002730  004737  015214'  CALL   ERRNAM         ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
45 002734  012700  177776  MOV    #-2,R0         ;ILLEGAL DEVICE
46 002740  000137  015144'  JMP    USRCLS
47          ;
48          ; Make sure user is authorized to access to new file.
49          ;
50 002744  004737  010566'  3$: CALL  CHKACC          ;SEE IF USER IS AUTHORIZED FOR ACCESS
51 002750  032763  0000000 0000000  BIT    #CS#RON,C.CSW(R3);IS USER AUTHORIZED TO WRITE TO NEW FILE?
52 002756  001016          BNE    11$          ;BR IF NOT
53          ;
54          ; Make sure device and unit match that of old file.
55          ; R4 = Device table index for new file.
56          ; R0 = Device unit number for new file.
57          ;

```

.RENAME

```

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

```

.RENAME

```

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 0000000 0000000 BIC      #FS$EMP,FD$STA(R1);CLEAR EMPTY-FILE FLAG
126 003210 052761 0000000 0000000 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 0000020          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

```

.DELETE

```

1          .SBTTL .DELETE
2          ;-----
3          ; Delete a permanent file entry.
4          ;
5 003266 004737 007362' DELETE: CALL  USRCOM          ;DO COMMON SETUP
6          ;
7          ; See if we have write access to file being deleted.
8          ;
9 003272 032763 000000G 000000G          BIT    #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO FILE?
10 003300 001406          BEQ    2$          ;BR IF YES
11 003302 004737 015214'          CALL   ERRNAM          ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
12 003306 012700 177763          MOV    #-15,R0         ;SET ABORT ERROR CODE
13 003312 000137 015144'          JMP    USRCLS          ;ABORT THE OPERATION
14          ;
15          ; See if device is a magnetic tape.
16          ;
17 003316 032764 000000G 000000G 2$:    BIT    #DS$NRD,DVSTAT(R4); IS THIS A MAG TAPE DEVICE?
18 003324 001004          BNE    6$          ;Br if yes
19 003326 032764 000000G 000000G          BIT    #DX$NRD,DVFLAG(R4); Internal flag set?
20 003334 001404          BEQ    5$          ;BR IF NOT
21 003336 012702 000000G          6$:    MOV    #DF$DEL,R2         ;SET DELETE FUNCTION CODE
22 003342 000137 014010'          JMP    SPLDIR          ;DO SPECIAL DEVICE DELETE
23          ;
24          ; See if this is a file structured device.
25          ;
26 003346 032764 000000G 000000G 5$:    BIT    #DS$DIR,DVSTAT(R4); IS THIS A DIRECTORY STRUCTURED DEVICE?
27 003354 001004          BNE    1$          ;BR IF YES
28          ;
29          ; Delete is invalid on non-file-structured device.
30          ;
31 003356 012700 000002          MOV    #2,R0          ;RETURN INVALID-OPERATION ERROR CODE
32 003362 000137 015144'          JMP    USRCLS
33          ;
34          ; Try to locate directory entry for file being deleted.
35          ;
36 003366 004737 011242'          1$:    CALL   FNDFIL          ;LOOK UP THE FILE
37 003372 103425          BCS    9$          ;BR IF FILE DOES NOT EXIST
38          ;
39          ; See if file is protected.
40          ;
41 003374 032761 000000G 000000G          BIT    #FS$PRO,FD$STA(R1); IS FILE PROTECTED?
42 003402 001404          BEQ    3$          ;BR IF NOT
43 003404 012700 000003          MOV    #3,R0          ;RETURN ERROR CODE 3 IF FILE IS PROTECTED
44 003410 000137 015144'          JMP    USRCLS
45          ;
46          ; Remove file entry from directory cache.
47          ;
48 003414 004737 014616'          3$:    CALL   CSHDEL          ;REMOVE FILE ENTRY FROM DIRECTORY CACHE
49          ;
50          ; File exists. Mark its entry as empty.
51          ;
52 003420 012761 000000G 000000G          MOV    #FS$EMP,FD$STA(R1); MARK DIR ENTRY AS EMPTY
53          ;
54          ; Consolidate and rewrite the directory segment.
55          ;
56 003426 004737 012634'          CALL   CONSOL          ;CONSOLIDATE THE SEGMENT
57 003432 004737 013706'          CALL   WRTSEG          ;WRITE SEGMENT TO DISK

```

.DELETE

```
58 ;  
59 ; Finished  
60 ;  
61 003436 005063 0000000 CLR C.CSW(R3) ;PURGE THE CHANNEL  
62 003442 000137 015172' JMP USRXIT  
63 ;  
64 ; File does not exist.  
65 ;  
66 003446 012700 000001 9#: MOV #1,R0 ;FILE DOES NOT EXIST  
67 003452 000137 015144' JMP USRCLS
```

.CLOSE

```

1          .SBTTL .CLOSE
2          ;-----
3          ; Close a channel.
4          ;
5 003456 013703 0000000 CLOSE:  MOV     CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6 003462 032763 0000000 0000000 BIT     #CS$DPN,C.CSW(R3); IS THE CHANNEL OPEN?
7 003470 001002          BNE     7$          ;BR IF CHANNEL IS OPEN
8 003472 000137 004202'  JMP     10$         ;NOP IF NOT OPEN
9          ;
10         ; Wait for all I/O to finish on this channel
11         ;
12 003476 004737 0000000 7$:    CALL    IOWAIT      ;WAIT FOR ALL I/O ON CHANNEL TO FINISH
13         ;
14         ; See if we are closing a spooled device.
15         ;
16 003502 032763 0000000 0000000 BIT     #CS$SPL,C.CSW(R3); IS THIS A SPOOLED DEVICE CHANNEL?
17 003510 001405          BEQ     1$          ;BR IF NOT
18 003512          DCALL   SDCLOS      ;CLOSE A SPOOLED DEVICE CHANNEL
19 003520 000137 004176'  JMP     9$          ;
20         ;
21         ; See if device being closed is a magnetic tape.
22         ;
23 003524 016300 0000000 1$:    MOV     C.CSW(R3),R0    ;GET CSW
24 003530 042700 0000000 BIC     #^C<CS$NMX>,R0 ;MASK OUT ALL BUT DEVICE TABLE INDEX
25 003534 032760 0000000 0000000 BIT     #DS$NRD,DVSTAT(R0); IS DEVICE A MAG TAPE?
26 003542 001004          BNE     18$         ;Br if yes
27 003544 032760 0000000 0000000 BIT     #DX$NRD,DVFLAG(R0); Internal flag set?
28 003552 001404          BEQ     6$          ;BR IF NOT
29 003554 012702 0000000 18$:   MOV     #DF$CLS,R2      ;DO SPECIAL DEVICE CLOSE
30 003560 000137 014010'  JMP     SPLDIK
31         ;
32         ; See if we are closing a shared file.
33         ;
34 003564 005737 0000000 6$:    TST     JCDB          ;Does job have any shared file chans open?
35 003570 001402          BEQ     17$         ;BR IF NOT
36 003572 004777 0000000          CALL    @SFCLS        ;CHECK FOR CLOSING A SHARED FILE
37         ;
38         ; See if this file was opened with a .LOOKUP or .ENTER
39         ;
40 003576 032763 0000000 0000000 17$:   BIT     #CS$ENT,C.CSW(R3); DID WE DO A .ENTER ON THE CHANNEL?
41 003604 001574          BEQ     9$          ;BR IF NOT
42         ;
43         ; We are closing a new file on a directory structured device.
44         ; Gain exclusive access to USR data base.
45         ;
46 003606 004737 007770'          CALL    GETUSR        ;CLAIM EXCLUSIVE ACCESS TO USR
47         ;
48         ; Set up device # and unit # in FILDVU.
49         ;
50 003612 116337 0000000 002015'  MOVB    C.DEVQ(R3),FILDVU+1; SET UNIT #
51 003620 016300 0000000 MOV     C.CSW(R3),R0    ;GET CHANNEL STATUS WORD
52 003624 042700 0000000 BIC     #^C<CS$NMX>,R0 ;MASK OUT ALL BUT DEVICE INDEX #
53 003630 110037 002014'  MOVB    R0,FILDVU      ;SET DEVICE INDEX #
54         ;
55         ; Set up info about this directory
56         ;
57 003634 005063 0000000 CLR     C.SBLK(R3)     ;SAY CHANNEL BEING USED FOR DIRECTORY OP

```

```

58 003640 042763 0000000 0000000      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      CHNUM,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 0000000 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 0000000 0000000 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 0000000 ADD R2,FD$LEN(R1) ;ADD RESIDUAL BLOCKS TO EMPTY ENTRY
123 ; Set date and time of file creation in file directory entry.
124 004112 163701 002006' SUB DIRSIZ,R1 ;POINT BACK TO NEW DIRECTORY ENTRY
125 004116 005061 0000000 CLR FD$TIM(R1) ;SAY CREATION TIME UNKNOWN
126 004122 005761 0000000 TST FD$DAT(R1) ;DID DIDDLE ROUTINE SET FILE DATE?
127 004126 001013 BNE 16$ ;BR IF YES
128 004130 013761 0000000 0000000 MOV SYSDAT,FD$DAT(R1);SET DATA THAT FILE WAS CREATED (CLOSED)
129 004136 013704 0000000 MOV SYTIMH,R4 ;GET CURRENT TIME OF DAY (HIGH ORDER)
130 004142 013705 0000000 MOV SYTIML,R5 ;GET LOW-ORDER TIME OF DAY
131 004146 071437 0000000 DIV TK3SVL,R4 ;CONVERT TO 3-SECOND UNITS
132 004152 010461 0000000 MOV R4,FD$TIM(R1) ;SET TIME OF FILE CREATION
133 ; Add entry for new file to directory cache.
134 004156 004737 014466' 16$: CALL CSHADD ;ADD TO CACHE TABLE
135 ; Consolidate and rewrite the directory segment.
136 004162 004737 012634' CALL CONSOL ;CONSOLIDATE THE DIRECTORY
137 004166 004737 013706' CALL WRTSEG ;WRITE OUT THE SEGMENT
138 ;
139 ; Free the USR
140 ;
141 004172 004737 010132' CALL FREUSR ;RELEASE USR DATA BASE
142 ;
143 ; Mark the channel as closed.
144 ;
145 004176 005063 0000000 9$: CLR C.CSW(R3) ;SAY THE CHANNEL IS CLOSED
146 ;
147 ; Finished
148 ;
149 004202 000137 0000000 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$: MOV #UERR3,R0
161 004226 000409 BR 20$
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

```

.DSTATUS

```

1
2
3
4
5 004246 004737 007770'
6
7 004252 013701 000000G
8 004256 004737 007532'
9
10 004262 004737 010212'
11 004266 103003
12
13 004270 005000
14 004272 000137 015156'
15
16
17
18
19
20 004276 010005
21 004300 013700 000004G
22 004304 010037 000000G
23 004310 004737 000000G
24 004314 016446 000000G
25 004320 106620
26 004322 016446 000000G
27 004326 106620
28 004330 016446 000000G
29 004334 106620
30 004336 016446 000000G
31 004342 106620
32
33
34
35 004344 020437 000000G
36 004350 001004
37 004352 006305
38 004354 016546 000000G
39 004360 106640
40
41
42
43 004362 000137 015172'

```

```

.SBTTL .DSTATUS
-----
.DSTATUS -- Determine device status.
DSTAT: CALL GETUSR ;GET EXCLUSIVE ACCESS TO USR DATA BASE
; Move device name to FILSPC buffer and perform any assigns.
MOV URO,R1 ;POINT TO DEVICE SPEC
CALL GETSPC ;MOVE TO FILSPC BUFFER
; Look up device in perm name table.
CALL CHKDEV ;LOOK UP THE DEVICE NAME
BCC 1$ ;BR IF FOUND
; Invalid device name
CLR RO ;SET ERROR CODE
JMP USRERR
; Valid device.
; Return info about it.
; (R4 now has device table index)
1$: MOV RO,R5 ;CARRY UNIT NUMBER IN R5
MOV EMTBLK+4,RO ;GET POINTER TO USER'S RESULT AREA
MOV RO,URO ;RETURN POINTER TO RESULT IN RO
CALL VALADW ;VALIDATE THE ADDRESS
MOV DVSTAT(R4),-(SP);DEVICE STATUS WORD
MTPD (RO)+
MOV HANSIZ(R4),-(SP);HANDLER SIZE
MTPD (RO)+
MOV HANENT(R4),-(SP);HANDLER ENTRY POINT
MTPD (RO)+
MOV DEVSIZ(R4),-(SP);DEVICE SIZE
MTPD (RO)+
; If this is a logical disk, return file size as device size
CMP R4,LDDEVX ;IS THIS A LOGICAL DISK?
BNE 2$ ;BR IF NOT
ASL R5 ;CONVERT UNIT # TO WORD TABLE INDEX
MOV LDSIZE(R5),-(SP);GET LOGICAL DISK SIZE
MTPD -(RO) ;PASS TO USER
; Finished
2$: JMP USRXIT ;FINISHED

```



```

1          .SBTTL .SFDAT, .SFTIM, & .FPROT
2          ;-----
3          ; .SFDAT -- Set date in file entry
4          ;
5 004440 004737 004704' SFDATE: CALL SFCOM ;LOCATE FILE ENTRY
6 004444 004737 004754' CALL SFWRIT ;MAKE SURE WE HAVE WRITE ACCESS
7 004450 013700 0000040 MOV EMTBLK+4,R0 ;GET SPECIFIED DATE VALUE
8 004454 001002 BNE 1$ ;BR IF DATE SPECIFIED
9 004456 013700 0000000 MOV SYSDAT,R0 ;USE CURRENT DATE
10 004462 016105 0000000 1$: MOV FD$DAT(R1),R5 ;SAVE OLD FILE DATE
11 004466 010061 0000000 MOV R0,FD$DAT(R1) ;SET NEW DATE IN FILE ENTRY
12          ;
13          ; If PIP is doing a .SFDAT, set the time entry in the file directory
14          ; to the time from the file last looked up.
15          ; This is done to preserve both date and time for a file across
16          ; a PIP copy operation.
17          ;
18          ; MOVB CORUSR,R4 ;GET CURRENT JOB INDEX NUMBER
19          ; BIT #$PIPRN,LSW6(R4); IS PIP RUNNING?
20          ; BEQ SFEXIT ;BR IF NOT
21          ; CMP R0,LSTFDD ;ARE WE USING THE OLD FILE'S DATE?
22          ; BNE SFEXIT ;BR IF NOT
23          ; MOV LSTFDT,FD$TIM(R1);USE TIME VALUE FROM LAST .LOOKUP
24          ;
25          ; Add new file entry to directory cache
26          ;
27 004472 004737 014466' SFEXIT: CALL CSHADD ;ADD ENTRY TO CACHE
28          ;
29          ; Rewrite directory segment
30          ;
31 004476 004737 013706' CALL WRTSEG ;REWRITE DIRECTORY SEGMENT
32          ;
33          ; Finished
34          ;
35 004502 005063 0000000 CLR C.CSW(R3) ;SAY CHANNEL IS CLOSED
36 004506 000137 015172' JMP USRXIT ;EXIT
37          ;
38          ;-----
39          ; .SFTIM -- Set file time in file entry
40          ;
41 004512 004737 004704' SFTIME: CALL SFCOM ;LOCATE FILE ENTRY
42 004516 004737 004754' CALL SFWRIT ;MAKE SURE WE HAVE WRITE ACCESS
43 004522 013761 0000040 0000000 MOV EMTBLK+4,FD$TIM(R1) ;SET TIME VALUE IN DIRECTORY ENTRY
44 004530 000760 BR SFEXIT ;REWRITE DIRECTORY SEGMENT
45          ;
46          ;-----
47          ; .FPROT -- Set file protection
48          ;
49 004532 004737 004704' SFPROT: CALL SFCOM ;DO COMMON SETUP
50 004536 004737 004754' CALL SFWRIT ;MAKE SURE WE HAVE WRITE ACCESS
51 004542 113700 0000040 MOVB EMTBLK+4,R0 ;GET PROTECT/UNPROTECT FLAG
52 004546 001407 BEQ 2$ ;BR IF UNPROTECT WANTED
53 004550 020027 0000001 CMP R0,#1 ;VALUE MUST BE 0 OR 1
54 004554 001410 BEQ 3$ ;BR IF PROTECT WANTED
55 004556 012700 0000003 MOV #3,R0 ;ERROR IF NOT 0 OR 1
56 004562 000137 015144' JMP USRCLS
57 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	

.GFINF -- Get information about a file

```

1          .SBTTL .GFINF -- Get information about a file
2
3          ;-----
4          ; .GFINF Get the following information about a file:
5          ; 1. Size of the file.
6          ; 2. Protected/Unprotected status.
7          ; 3. Date of creation.
8          ; 4. Time of creation.
9          ; 5. Starting block number of file.
10         ;
10 004606 004737 004704' GFINFO: CALL SFCOM ; LOCATE FILE ENTRY
11 004612 013700 0000040 MOV EMTBLK+4,R0 ; GET ADDRESS OF USER'S INFO BUFFER
12 004616 004737 0000000 CALL VALADW ; MAKE SURE ADDRESS IS VALID
13         ; File size
14 004622 016146 0000000 MOV FD$LEN(R1),-(SP); FILE SIZE
15 004626 106620 MTPD (R0)+
16         ; Protected/Unprotected status
17 004630 005046 CLR -(SP) ; ASSUME FILE IS NOT PROTECTED
18 004632 032761 0000000 0000000 BIT #FS$PRO,FD$STA(R1); IS FILE PROTECTED?
19 004640 001401 BEQ 1$ ; BR IF NOT
20 004642 005216 INC (SP) ; RETURN 1 IF FILE IS PROTECTED
21 004644 106620 1$: MTPD (R0)+
22         ; Creation date
23 004646 016146 0000000 MOV FD$DAT(R1),-(SP); CREATION DATE
24 004652 106620 MTPD (R0)+
25         ; Creation time
26 004654 016146 0000000 MOV FD$TIM(R1),-(SP); CREATION TIME
27 004660 106620 MTPD (R0)+
28         ; Starting block number
29 004662 010002 MOV R0,R2 ; SAVE BUFFER POINTER
30 004664 004737 013420' CALL SBCALC ; CALCULATE STARTING BLOCK NUMBER
31 004670 010046 MOV R0,-(SP)
32 004672 106622 MTPD (R2)+ ; RETURN STARTING BLOCK NUMBER
33         ;
34         ; Finished
35         ;
36 004674 005063 0000000 CLR C.CSW(R3) ; SAY CHANNEL IS CLOSED
37 004700 000137 015172' JMP USRXIT ; EXIT

```

.GFINF -- Get information about a file

```

1 ;-----
2 ; Common setup routine used by SFDATE and SFPROT routines.
3 ; The USR entry setup is done including moving the file spec and checking
4 ; to see that the channel is open.
5 ; The directory entry is found.
6 ; If any errors are detected, an error return from the EMT is taken.
7 ;
8 ; Outputs:
9 ; R1 = Address of directory entry for file.
10 ; R3 = Address of CSW for channel.
11 ; R4 = Index into device table for device being opened.
12 ;
13 004704 004737 007362' SFCDM: CALL USRCOM ; DO USR ENTRY SETUP
14 ;
15 ; Make sure this is a file structured device and we have write
16 ; access to it.
17 ;
18 004710 032764 0000000 0000000 BIT #DS$DIR, DVSTAT(R4) ; IS THIS A FILE STRUCTURED DEVICE?
19 004716 001004 BNE 1$ ; BR IF YES
20 004720 012700 000002 MOV #2, R0 ; RETURN ERROR CODE 2 IF NOT
21 004724 000137 015144' JMP USRCLS
22 ;
23 ; Locate directory entry for the file
24 ;
25 004730 004737 011242' 1$: CALL FNDFIL ; LOCATE THE DIRECTORY ENTRY
26 004734 103004 BCC 3$ ; BR IF ENTRY FOR FILE FOUND
27 004736 012700 000001 MOV #1, R0 ; ERROR 1 IF CAN'T FIND ENTRY
28 004742 000137 015144' JMP USRCLS
29 ;
30 ; Remove directory entry from cache
31 ;
32 004746 004737 014616' 3$: CALL CSHDEL ; REMOVE ENTRY FROM CACHE
33 ;
34 ; Finished
35 ;
36 004752 000207 RETURN
37 ;-----
38 ;
39 ; Make sure that we have write access to a file.
40 ; Produce an abort if not.
41 ;
42 ; Inputs:
43 ; R3 = Address of CSW for channel opened to the file.
44 ;
45 004754 032763 0000000 0000000 SFWRIT: BIT #CS$RON, C. CSW(R3) ; DO WE HAVE WRITE ACCESS TO THE FILE?
46 004762 001406 BEQ 1$ ; BR IF YES
47 004764 004737 015214' CALL ERRNAM ; SET FILE SPEC FOR TSKMON ERROR MESSAGE
48 004770 012700 177763 MOV #-15, R0 ; ERROR IF NOT
49 004774 000137 015144' JMP USRCLS
50 005000 000207 1$: RETURN ; SUCCESSFUL RETURN

```

```

1          .SBTTTL  ALCEMT -- Allocate a device
2          ;-----
3          ; The ALLOCATE EMT is used to allocate a device for exclusive access by
4          ; the current job.
5          ;
6 005002  ALCEMT:
7          ;
8          ; Determine if this is an Allocate, Deallocate, or test allocation EMT
9          ;
10 005002 113700 0000000      MOVB    EMTBLK,R0      ;Get sub-function code
11 005006 001410             BEQ     10$              ;0==>Allocate
12 005010 120027 0000001      CMPB    R0,#1         ;1==>Deallocate
13 005014 001475             BEQ     DLCEMT
14 005016 120027 0000002      CMPB    R0,#2         ;2==>Test allocation
15 005022 001567             BEQ     TLCEMT
16 005024 000137 0000000      JMP     BADEMT       ;Invalid sub-function code
17          ;
18          ; Allocate a device
19          ;
20 005030 032737 0000000 0000000 10$:  BIT     #P0$ALC,PRIVCO ;Are we authorized to allocate devices?
21 005036 001004             BNE     11$              ;Br if yes
22 005040 012700 0000005      MOV     #5,R0         ;Error code 5 if not authorized
23 005044 000137 0000000      JMP     SETERR
24          ;
25          ; Do common allocate setup
26          ;
27 005050 004737 005622'     11$:  CALL    ALCCOM          ;Do common setup
28          ;
29          ; See if the device being allocated is currently allocated to another job
30          ;
31 005054 004737 005422'     CALL    ALCTST          ;Test for allocation to another job
32          ;
33          ; Device is not currently allocated.
34          ; Check to see if the device is already allocated by our job or a
35          ; related job.
36          ;
37 005060 012702 0000000      MOV     #ALCTBL,R2    ;Point to start of allocation table
38 005064 013700 002014'     MOV     FILDVU,R0     ;Get device # and index #
39 005070 020062 0000000 4$:  CMP     R0,AD$DVU(R2) ;Search for device in allocation table
40 005074 001406             BEQ     5$              ;Br if found it
41 005076 062702 0000000      ADD     #AD$$SZ,R2    ;Point to next entry
42 005102 020227 0000000      CMP     R2,#ALCEND    ;Checked all entries?
43 005106 103770             BLD     4$              ;Br if not
44 005110 000410             BR      7$              ;Br if not currently in allocation table
45          ;
46          ; Device is already in allocation table
47          ;
48 005112 116200 0000000 5$:  MOVB    AD$JOB(R2),R0 ;Get # of job that owns device
49 005116 120160 0000000      CMPB    R1,LNPRIM(R0) ;Is primary job reallocating device?
50 005122 001030             BNE     9$              ;Br if not
51 005124 110162 0000000      MOVB    R1,AD$JOB(R2) ;Let primary job take over device
52 005130 000425             BR      9$
53          ;
54          ; Device is not currently allocated.
55          ; Search for a free entry in the allocation table.
56          ;
57 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 0000003 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
```

DLCEMT -- Deallocate a device

```

1          .SBTTL DLCEMT -- Deallocate a device
2          ;-----
3          ; Deallocate a device.
4          ;
5 005210 032737 0000000 0000000 DLCEMT: BIT    #PO$ALC,PRIVCO ;Are we authorized to allocate devices?
6 005216 001004                BNE      11$      ;Br if yes
7 005220 012700 0000005                MOV     #5,R0      ;Error code 5 if not authorized
8 005224 000137 0000000                JMP     SETERR
9 005230 004737 005622' 11$: CALL   ALCCOM      ;Do common setup
10 005234 012702 0000000                MOV     #ALCTBL,R2 ;Point to start of allocation table
11          ;
12          ; If the device name is null, deallocate all devices allocated by
13          ; this user.
14          ;
15 005240 013700 0000000                MOV     FILSPC,R0 ;Is device name null?
16 005244 001015                BNE     2$        ;Br if not
17 005246 120162 0000000 1$: CMPB  R1,AD$JOB(R2) ;Is this entry an allocation for our job?
18 005252 001004                BNE     3$        ;Br if not
19 005254 105062 0000000                CLRB   AD$JOB(R2) ;Say no job using this entry
20 005260 005062 0000000                CLR    AD$DVU(R2) ;Say entry is free
21 005264 062702 0000000 3$: ADD   #AD$$SZ,R2 ;Point to next allocation table entry
22 005270 020227 0000000                CMP    R2,#ALCEND ;Checked all entries?
23 005274 103764                BLD    1$        ;Loop if more to check
24 005276 000437                BR     9$
25          ;
26          ; The device name is not null.
27          ; Deallocate the specified device.
28          ;
29 005300 105737 002014' 2$: TSTB  FILDVU      ;Trying to deallocate TT?
30 005304 001434                BEQ    9$        ;Ignore if yes
31 005306 013700 002014'                MOV    FILDVU,R0 ;Get device/unit ID for device being dealloc
32 005312 020062 0000000 7$: CMP   R0,AD$DVU(R2) ;Search for specified device in alloc table
33 005316 001406                BEQ    5$        ;Br if found it
34 005320 062702 0000000                ADD   #AD$$SZ,R2 ;Point to next entry in table
35 005324 020227 0000000                CMP   R2,#ALCEND ;Checked all entries?
36 005330 103770                BLD    7$        ;Keep looking if not
37 005332 000421                BR    9$        ;Device was not allocated by anyone
38 005334 116200 0000000 5$: MOVB  AD$JOB(R2),R0 ;Get # of job that owns this device
39 005340 126061 0000000 0000000 CMPB   LNPRIM(R0),LNPRIM(R1) ;Is device owned by us?
40 005346 001407                BEQ    6$        ;Br if yes
41 005350 006200                ASR   R0          ;Convert job index # to job #
42 005352 010037 0000000                MOV   R0,URO     ;Return owner job # in R0
43 005356 012700 0000001                MOV   #1,R0      ;Error 1 -- Device is allocated to someone
44 005362 000137 015156'                JMP   USRERR
45 005366 005062 0000000 6$: CLR   AD$DVU(R2) ;Say this table entry is free
46 005372 105062 0000000                CLRB  AD$JOB(R2)
47          ;
48          ; Finished
49          ;
50 005376 000137 015172' 9$: JMP   USRXIT

```

TLCEMT -- Check to see if a device is allocated

```
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
```

ALCTST -- See if device is allocated to another user

```

1          .SBTTL  ALCTST -- See if device is allocated to another user
2          ;-----
3          ; ALCTST is a subroutine called to determine if a device is allocated to
4          ; another user or in use by another user.
5          ;
6          ; Inputs:
7          ;   FILSPC = File spec for device being allocated.
8          ;   FILDVU = Device index # and unit # for device being tested.
9          ;   RI = Current job index number
10         ;
11         ; Outputs:
12         ;   If the device is in use by another user, error returns are made
13         ;   by jumping to USRERR.
14         ;   If the device is not allocated by other users, this routine returns.
15         ;
16 005422 010246 ALCTST: MOV     R2,-(SP)
17 005424 010346         MOV     R3,-(SP)
18         ;
19         ; Clear returned value for R0 in case device is not in use or allocated
20         ;
21 005426 005037 0000006         CLR     URO           ; Say no use of device found yet
22         ;
23         ; See if the device is legal
24         ;
25 005432 005737 0000006         TST     FILSPC        ; Don't allow null device
26 005436 001413         BEQ     11$           ; Error if name null
27 005440 105737 002014'        TSTB    FILDVU         ; Trying to allocate TT?
28 005444 001410         BEQ     11$           ; Br if yes
29 005446 123737 002014' 0000006 6$:  CMPB   FILDVU,SYINDEX ; Is this the system device?
30 005454 001010         BNE     12$           ; Br if not
31 005456 123737 002015' 0000016     CMPB   FILDVU+1,SYUNIT+1; Is this the system unit?
32 005464 001004         BNE     12$           ; Br if not
33 005466 012700 000002        11$:  MOV     #2,R0           ; Invalid device if null
34 005472 000137 015156'        JMP     USRERR
35         ;
36         ; See if the device is currently allocated to any user
37         ;
38 005476 012702 0000006        12$:  MOV     #ALCTBL,R2        ; Point to start of allocation table
39 005502 023762 002014' 0000006 1$:  CMP     FILDVU,AD$D$VU(R2); Search for entry for this device and unit
40 005510 001423         BEQ     5$             ; Br if found it
41 005512 062702 0000006        ADD     #AD$$SZ,R2        ; Point to next entry
42 005516 020227 0000006        CMP     R2,#ALCEND        ; Searched all entries?
43 005522 103767         BLO     1$             ; Loop if not
44         ;
45         ; Device is not currently allocated.
46         ; See if any other job has a channel open to this device.
47         ;
48 005524 004737 005742'        CALL   CHKUSE           ; See if device is being used by another job
49 005530 103007         BCC     13$           ; Br if no other jobs have device open
50 005532 006200         ASR     R0             ; Convert job index # to job #
51 005534 010037 0000006        MOV     R0,URO         ; Return in R0
52 005540 012700 000004        MOV     #4,R0         ; Error 4 -- Device in use by another job
53 005544 000137 015156'        JMP     USRERR
54 005550 006200        13$:  ASR     R0             ; Get # of job that is using the device
55 005552 010037 0000006        MOV     R0,URO         ; Return in R0
56 005556 000416         BR     9$
57         ;

```

```
58          ; The device is allocated.
59          ; See if it is allocated to our job or to another job.
60          ;
61 005560 116203 0000000 5#:   MOV      AD#JOB(R2),R3   ;Get # of job to which dev is allocated
62 005564 010337 0000000      MOV      R3,URO           ;Return job # to user in R0
63 005570 006237 0000000      ASR      URO
64 005574 126361 0000000 0000000  CMPB     LNPRIM(R3),LNPRIM(R1) ;Is dev allocated to our family?
65 005602 001404          BEQ      9#             ;Br if yes
66 005604 012700 000001    MOV      #1,R0           ;Error 1 -- Device is allocated to another job
67 005610 000137 015156'   JMP      USRERR
68          ;
69          ; Device is not allocated to another user
70          ;
71 005614 012603 9#:   MOV      (SP)+,R3
72 005616 012602      MOV      (SP)+,R2
73 005620 000207      RETURN
```

```

1          .SBTTL  ALCCOM -- Common setup for Allocate/Deallocate
2          ;-----
3          ; Do common setup for Allocate/Deallocate EMT's.
4          ;
5          ; Outputs:
6          ;   USR data base is locked for exclusive access.
7          ;   FILSPC = File spec passed with EMT.
8          ;   FILDVU = Device index # and unit #
9          ;   R4 = Device index number.
10         ;
11 005622 010146 ALCCOM: MOV     R1,-(SP)
12         ;
13         ; Gain exclusive access to USR data base
14         ;
15 005624 004737 007770'      CALL     GETUSR      ;Get exclusive use of USR
16         ;
17         ; Move file spec to FILSPC and apply any assigns
18         ;
19 005630 013701 000002G      MOV     EMTBLK+2,R1    ;Get pointer to device spec argument
20 005634 004737 007532'      CALL     GETSPC      ;Setup FILSPC
21         ;
22         ; Check to see if the device is valid
23         ;
24 005640 005737 000000G      TST     FILSPC      ;Is the device null?
25 005644 001430              BEQ     2$              ;Treat this as a special valid device
26 005646 004737 010212'      CALL     CHKDEV      ;See if device is valid
27 005652 103016              BCC     1$              ;Br if CHKDEV says it is valid
28 005654 020437 000000G      CMP     R4,CLDEVX     ;Is this a CL unit?
29 005660 001004              BNE     3$              ;Br if not
30 005662 020027 000000G      CMP     R0,#CLTOTL    ;Valid CL unit number?
31 005666 103417              BLO     2$              ;Br if yes
32 005670 000412              BR     8$              ;Invalid device
33 005672 020437 000000G      3$:    CMP     R4,C1DEVX     ;Is this a C1 unit?
34 005676 001007              BNE     8$              ;Br if not
35 005700 020027 177770G      CMP     R0,#CLTOTL-8. ;Is this a valid C1 unit number?
36 005704 002410              BLT     2$              ;Br if yes
37 005706 000403              BR     8$              ;Br if not
38 005710 020437 000000G      1$:    CMP     R4,LDDEVX     ;Is this a logical disk?
39 005714 001004              BNE     2$              ;Br if not -- Don't allow alloc of LD
40         ;
41         ; This is an invalid device
42         ;
43 005716 012700 000002G      8$:    MOV     #2,R0              ;Error 2 -- Invalid device
44 005722 000137 015156'      JMP     USRERR
45         ;
46         ; This is a valid device. Build the FILDVU word
47         ;
48 005726 110437 002014'      2$:    MOVB    R4,FILDVU     ;Set device index number
49 005732 110037 002015'      MOVB    R0,FILDVU+1     ;Set unit number
50         ;
51         ; Finished
52         ;
53 005736 012601              MOV     (SP)+,R1
54 005740 000207              RETURN

```

CHKUSE -- See if any channels open to a specified device

```

1          .SBTTL  CHKUSE -- See if any channels open to a specified device
2          ;-----
3          ;  CHKUSE is called to see if any jobs other than a specified job and its
4          ;  associated virtual jobs have any channels open to a specified device.
5          ;
6          ;  Inputs:
7          ;  R1 = Job whose access is to be allowed.
8          ;  FILDVU = Device index # and unit # of device to be checked.
9          ;
10         ;  Outputs:
11         ;  C-flag cleared ==> No job other than current job has
12         ;                    channels opened to the device.
13         ;  C-flag set      ==> Some other job has a channel open to the device.
14         ;  R0 = Job index number of job that is accessing the device.
15         ;        (Zero returned if no job is using the device).
16         ;
17 005742 010246  CHKUSE: MOV     R2,-(SP)
18 005744 010346          MOV     R3,-(SP)
19 005746 010446          MOV     R4,-(SP)
20 005750 010546          MOV     R5,-(SP)
21         ;
22         ;  Get exclusive access to job context block buffer
23         ;
24 005752          DCALL  GETCXT          ;Get exclusive access to context block buffer
25         ;
26         ;  First check to see if the specified device is mounted by another user.
27         ;  We also check to see if any logical disk on the specified device is
28         ;  mounted.
29         ;
30 005760 005004          CLR     R4          ;Have not found any job accessing device
31 005762 013705 0000000  MOV     CSHDEV,R5          ;Point to start of mount table
32 005766 005765 0000000 6$:  TST     CD$DVU(R5)          ;Is this mount entry in use?
33 005772 001430          BEQ     7$          ;Br if not
34 005774 026537 0000000 002014'  CMP     CD$DVU(R5),FILDVU ;Does this match device of interest?
35 006002 001024          BNE     7$          ;Br if not
36         ;
37         ;  We found a mount entry for the device we are checking.
38         ;  Now we must determine which job(s) have mounted the device.
39         ;
40 006004 012702 0000000          MOV     #LSTSL,R2          ;Get index # of last job
41 006010 020201          8$:  CMP     R2,R1          ;Are we checking current job?
42 006012 001415          BEQ     10$          ;Br if yes -- It is ok
43 006014 010203          MOV     R2,R3          ;Get index number
44 006016 010246          MOV     R2,-(SP)          ;Preserve R2
45 006020 004737 015112'          CALL  CDJFLG          ;Get pointer to flag for this job
46 006024 010200          MOV     R2,R0          ;Get pointer to byte with flag bit
47 006026 012602          MOV     (SP)+,R2          ;Recover job number
48 006030 130310          BITB   R3,(R0)          ;Does this job have device mounted?
49 006032 001405          BEQ     10$          ;Br if not
50         ;
51         ;  We have found a job that has the device mounted.
52         ;  See if it is ok for that job to access the device.
53         ;
54 006034 126261 0000000 0000000  CMPB   LNPRIM(R2),LNPRIM(R1) ;Is this job allowed to access the dev?
55 006042 001073          BNE     5$          ;Br if not
56 006044 010204          MOV     R2,R4          ;Remember # of job accessing device
57 006046 162702 0000002 10$:  SUB     #2,R2          ;More jobs to check?

```

```

58 006052 003356          BGT      8$          ;Br if yes
59                          ;
60                          ; Check next mount table entry
61                          ;
62 006054 062705 0000000 7$:      ADD      #CD#$SZ,R5      ;Point to next mount table entry
63 006060 020537 0000000      CMP      R5,CSHDVN      ;Have we checked all mount table entries?
64 006064 103740          BLO      6$          ;Br if not
65                          ;
66                          ; Finished checking all mount table entries
67                          ;
68 006066 010405          MOV      R4,R5          ;Get # of any job accessing device
69                          ;
70                          ; Begin loop to cycle through all jobs to see if any job has
71                          ; a channel opened to this device.
72                          ;
73 006070 012702 0000002      MOV      #2,R2          ;Get index # of 1st job
74                          ;
75                          ; Ignore this job if it is not logged on.
76                          ;
77 006074 032762 0000000 0000000 1$:    BIT      ##KINIT,LSW(R2) ;Is this job logged on?
78 006102 001440          BEQ      2$          ;Br if not
79                          ;
80                          ; Begin loop to cycle through all channels for this job
81                          ;
82 006104 005003          CLR      R3          ;Start with channel # 0
83 006106          3$:      OCALL     GETCHA      ;Get address of this channel
84                          ;
85                          ; Ignore this channel if it is not open
86                          ;
87 006114 032760 0000000 0000000      BIT      #CS$DPN,C.CSW(R0) ;Is this channel open?
88 006122 001424          BEQ      4$          ;Br if not
89                          ;
90                          ; Get the device index number and the channel number out of the
91                          ; channel block.
92                          ;
93 006124 010004          MOV      R0,R4          ;Get addr of channel block to R4
94 006126 016400 0000000      MOV      C.CSW(R4),R0      ;Get CSW
95 006132 042700 177701      BIC      #^C76,R0        ;Extract device index number
96 006136 116404 0000000      MOVVB   C.DEVQ(R4),R4      ;Get unit number
97 006142 042704 177770      BIC      #^C7,R4         ;Clear all but unit number
98 006146 120037 002014'      CMPB    R0,FILDVU        ;Does it match the device of interest?
99 006152 001010          BNE     4$          ;Br if not
100 006154 120437 002015'      CMPB    R4,FILDVU+1      ;Does unit number match?
101 006160 001005          BNE     4$          ;Br if not
102                          ;
103                          ; We found a job that has a channel open to the device
104                          ;
105 006162 126261 0000000 0000000      CMPB    LNPRIM(R2),LNPRIM(R1) ;Is this job allowed to access the dev?
106 006170 001020          BNE     5$          ;Br if not
107 006172 010205          MOV      R2,R5          ;Remember # of friendly job
108                          ;
109                          ; Check next channel (including Command file, log file, spool control
110                          ; and SAV file load channels)
111                          ;
112 006174 005203          4$:      INC      R3          ;Advance the channel number
113 006176 020327 0000000      CMP      R3,#NLCHN      ;Checked all of user's channels?
114 006202 103741          BLO     3$          ;Loop if more to check

```

CHKUSE -- See if any channels open to a specified device

```

115 ;
116 ; Check next job
117 ;
118 006204 062702 000002 2$: ADD #2,R2 ;Get # of next job
119 006210 020227 0000000 CMP R2,#LSTSL ;Checked all jobs?
120 006214 101727 BLOS 1$ ;Loop if more to check
121 ;
122 ; Finished checking all jobs
123 ; See if a friendly job is using the device
124 ;
125 006216 DCALL FREEXT ;Free context block buffer
126 006224 010500 MOV R5,R0 ;Get # of any job accessing the device
127 006226 000241 CLC ;Set flag saying device is free
128 006230 000405 BR 9$
129 ;
130 ; We found a job accessing the channel who is in conflict with testing job
131 ;
132 006232 5$: DCALL FREEXT ;Free context block buffer
133 006240 010200 MOV R2,R0 ;Return job index number in R0
134 006242 000261 SEC ;Set flag saying that device is in use
135 ;
136 ; Finished
137 ;
138 006244 012605 9$: MOV (SP)+,R5
139 006246 012604 MOV (SP)+,R4
140 006250 012603 MOV (SP)+,R3
141 006252 012602 MOV (SP)+,R2
142 006254 000207 RETURN

```

MOUNT -- Mount a new file structure

```

1          .SBTTL  MOUNT  -- Mount a new file structure
2          ;-----
3          ; The MOUNT EMT is used to introduce a new file structure to the system.
4          ;
5 006256 004737 007120' MOUNT: CALL  MNTCOM          ; DO COMMON SETUP
6          ;
7          ; First dismount the device to clear the directory file entries
8          ;
9 006262 004737 006732'          CALL  DMTSUB          ; DISMOUNT THE DEVICE
10         ;
11        ; See if this device is mounted by other users.
12        ;
13 006266 004737 014776'          CALL  CSHTST          ; SEARCH FOR DEVICE IN MOUNT TABLE
14 006272 103105          BCC    4$              ; BR IF DEVICE IS ALREADY MOUNTED
15        ;
16        ; Device is not currently mounted by any job.
17        ; Check to see if this device is eligible for directory caching.
18        ;
19 006274 113705 002014'          MOVB   FILDVU,R5          ; GET DEVICE INDEX NUMBER
20 006300 016500 0000000          MOV    DVSTAT(R5),R0      ; GET STATUS FLAGS FOR THIS DEVICE
21 006304 032700 0000000          BIT    #DS$DIR,R0          ; IS THIS A DIRECTORY STRUCTURED DEVICE?
22 006310 001503          BEQ    3$              ; BR IF NOT -- DON'T CACHE
23 006312 032700 0000000          BIT    #DS$NRD,R0          ; NON RT-11 DIRECTORY STRUCTURE (MAG TAPE)?
24 006316 001100          BNE    3$              ; BR IF YES -- DON'T CACHE
25 006320 032765 0000000 0000000 BIT    #DX$NRD,DVFLAG(R5); Internal non-RT-dir flag set?
26 006326 001074          BNE    3$              ; Br if yes
27 006330 032765 0000000 0000000 BIT    #DX$NMT,DVFLAG(R5); Is it flagged for no mount?
28 006336 001070          BNE    3$              ; Br if yes
29        ;
30        ; Look for a free entry in the mount table.
31        ;
32 006340 013705 0000000          MOV    CSHDEV,R5          ; POINT TO TABLE OF STRUCTURES
33 006344 005765 0000000 1$: TST   CD$DVU(R5)          ; SEARCH FOR A FREE SLOT IN THE TABLE
34 006350 001411          BEQ    2$              ; BR IF FOUND ONE
35 006352 062705 0000000          ADD    #CD$SZ,R5          ; POINT TO NEXT TABLE ENTRY
36 006356 020537 0000000          CMP    R5,CSHDVN          ; HIT END OF TABLE?
37 006362 103770          BLO    1$              ; BR IF NOT
38 006364 012700 0000001          MOV    #1,R0              ; NO FREE SLOTS IN TABLE
39 006370 000137 015156'          JMP    USRERR
40        ;
41        ; We found a free entry in the mount table.
42        ; Set up the entry for this device.
43        ;
44 006374 004737 015054' 2$: CALL  GETDVU          ; GET PHYSICAL DEVICE # AND UNIT #
45 006400 005065 0000000          CLR    CD$NAM(R5)          ; CLEAR FILE NAME (ASSUME NOT LOGICAL DISK)
46 006404 005065 0000026          CLR    CD$NAM+2(R5)
47 006410 012765 177777 0000000 MOV    #177777,CD$TOP(R5); ASSUME NOT LOGICAL DISK FOR TOP BLOCK
48 006416 010365 0000000          MOV    R3,CD$DVU(R5)      ; STORE PHYSICAL DEVICE # AND UNIT #
49 006422 010465 0000000          MOV    R4,CD$BAS(R5)     ; STORE BASE BLOCK # IF LOGICAL DISK
50 006426 001420          BEQ    6$              ; BR IF NOT A LOGICAL DISK
51 006430 113703 002015'          MOVB  FILDVU+1,R3        ; THIS IS A LOGICAL DISK, GET LD UNIT #
52 006434 006303          ASL    R3              ; CONVERT UNIT # TO WORD TABLE INDEX
53 006436 066304 0000000          ADD    LD$SIZE(R3),R4    ; GET # OF BLOCK ABOVE TOP OF LOGICAL DISK
54 006442 010465 0000000          MOV    R4,CD$TOP(R5)     ; SET TOP BLOCK NUMBER OF LOGICAL DISK
55 006446 072327 0000002          ASH   #2,R3              ; TIMES 8 BYTES PER ENTRY
56 006452 062703 0000000          ADD    #LD$NAME,R3       ; POINT TO ENTRY FOR NAME OF THIS LD FILE
57 006456 005723          TST   (R3)+              ; SKIP OVER DEVICE NAME

```

MOUNT -- Mount a new file structure

```

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$:      MOVB   CORUSR,R3      ;Get our job index number
73 006512 004737 015112'      CALL   CDJFL0      ;GET MOUNT FLAG FOR OUR JOB
74 006516 150312      BISB   R3,(R2)      ;SET MOUNT FLAG FOR OUR JOB
75 006520 000137 015172'      3$:      JMP     USRXIT

```

```

1          .SBTTL  DISMNT -- Dismount a file structure
2          ;-----
3          ; The DISMNT EMT is used to remove a file structure from the system tables.
4          ;
5 006524   DISMNT:
6          ;
7          ; Determine if this is a request to dismount a specific file structure
8          ; or a request to clean out the directory cache.
9          ;
10 006524 113700 000000G      MOVB    EMTBLK,RO      ;Get sub-function code
11 006530 120027 000001      CMPB    RO,#1         ;Clean out directory cache?
12 006534 001430              BEQ     CSHDMT        ;Br if yes
13 006536 120027 000002      CMPB    RO,#2         ;Dismount all devices for job?
14 006542 001442              BEQ     DMTALL        ;Br if yes
15 006544 120027 000003      CMPB    RO,#3         ;Dismount a LD structure?
16 006550 001414              BEQ     1$           ;Br if yes
17 006552 120027 000004      CMPB    RO,#4         ;Get information about a LD structure?
18 006556 001413              BEQ     2$           ;Br if yes
19 006560 120027 000005      CMPB    RO,#5         ;Dismount all LD structures?
20 006564 001412              BEQ     3$           ;Br if yes
21          ;
22          ; This is a request to dismount a file structure
23          ;
24 006566 004737 007120'      CALL    MNTCOM        ;Do common setup
25          ;
26          ; Call DMTSUB subroutine to do real dismount work.
27          ;
28 006572 004737 006732'      CALL    DMTSUB        ;Do the dismount
29          ;
30          ; Finished
31          ;
32 006576 000137 015172'      JMP     USRXIT
33          ;
34          ; Dismount a LD structure
35          ;
36 006602 000137 000000G      1$:    JMP     LDDEMT        ;Dismount a LD
37          ;
38          ; Get information about a LD structure
39          ;
40 006606 000137 000000G      2$:    JMP     LDIEMT        ;Get info about a LD
41          ;
42          ; Dismount all LD structures
43          ;
44 006612 000137 000000G      3$:    JMP     LDAEMT        ;Dismount all LD's
45          ;
46          ;-----
47          ; CSHDMT removes all entries from the directory cache.
48          ;
49 006616 004737 007770'      CSHDMT: CALL    GETUSR        ;GET EXCLUSIVE USE OF USR
50 006622 013700 000000G      MOV    CSHHD,RO      ;POINT TO 1ST CACHED DIRECTORY ENTRY
51 006626 005060 000000G      1$:    CLR    FC$CDX(RO) ;SAY ENTRY IS EMPTY
52 006632 005060 000000G      CLR    FD$NAM(RO)
53 006636 016000 000000G      MOV    FC$LNK(RO),RO ;LINK TO NEXT ENTRY
54 006642 001371              BNE    1$           ;BRANCH BACK IF ANOTHER TO FREE
55 006644 000137 015172'      JMP    USRXIT        ;FINISHED

```

```

1          .SBTTL  DMTALL -- Dismount all mounted devices for job
2          ;-----
3          ; Dismount all devices for this job.
4          ;
5 006650   DMTALL:
6          ;
7          ; Gain exclusive access to USR data base
8          ;
9 006650   004737 007770'      CALL    GETUSR      ;Get exclusive access to USR
10         ;
11        ; Get flag bit that identifies our job in mount table entries
12        ;
13 006654   113703 0000000    MOV     CORUSR,R3      ;Get our job index number
14 006660   005005          CLR     R5              ;Say no entry to point to yet
15 006662   004737 015112'    CALL    CDJFLG       ;Get flag bit for our job
16        ;
17        ; Locate each device this is mounted by our job
18        ;
19 006666   013705 0000000    MOV     CSHDEV,R5     ;Point to 1st cache device entry
20 006672   005765 0000000    1$:    TST     CD$DVU(R5) ;Is this entry in use?
21 006676   001406          BEQ     2$              ;Br if not
22 006700   010500          MOV     R5,R0         ;Get address of entry
23 006702   060200          ADD     R2,R0         ;Point to byte with our job flag bit
24 006704   130310          BITB   R3,(R0)        ;Is this device mounted by our job?
25 006706   001402          BEQ     2$              ;Br if not
26 006710   004737 006752'    CALL    DMTDEV       ;Dismount this entry
27 006714   062705 0000000    2$:    ADD     #CD#$SZ,R5 ;Point to next mount table entry
28 006720   020537 0000000    CMP     R5,CSHDVN    ;Checked all entries?
29 006724   103762          BLO    1$              ;Loop if not
30        ;
31        ; Say job has no logical disks mounted
32        ;
33        ; MOV     #MAXLD,R0      ;Get # of LD units
34        ; MOV     #LDNAME,R5    ;Point to LD name table
35 3$:     CLR     (R5)          ;Say this LD not in use
36        ; ADD     #8.,R5        ;Point to next LD table entry
37        ; SOB     R0,3$         ;Br if more to do
38        ;
39        ; Finished
40        ;
41 006726   000137 015172'    JMP     USRXIT       ;Finished

```

DMTALL -- Dismount all mounted devices for job

```

1 ; -----
2 ; The DMTSUB subroutine is called to do the actual work of dismounting
3 ; a file structure.
4 ;
5 ; Inputs:
6 ; FILDVU = Device/Unit number info for device being dismounted.
7 ;
8 006732 010546 DMTSUB: MOV R5, -(SP)
9 ;
10 ; Locate entry for device in cached-device table.
11 ;
12 006734 004737 014776' CALL CSHTST ;Locate entry for device in cached dev table
13 006740 103402 BCS 9# ;Br if device is not mounted
14 ;
15 ; Device is mounted, dismount it.
16 ;
17 006742 004737 006752' CALL DMTDEV ;Dismount the device
18 ;
19 ; Finished
20 ;
21 006746 012605 9#: MOV (SP)+, R5
22 006750 000207 RETURN

```

```

1          .SBTTL  DMTDEV -- Remove entry from cache table
2          ;-----
3          ; DMTDEV is called to remove a particular entry from the mount table.
4          ;
5          ; Inputs:
6          ; R5 = Pointer to cached device entry to be removed.
7          ;
8 006752 010246 DMTDEV: MOV     R2,-(SP)
9 006754 010346         MOV     R3,-(SP)
10         ;
11         ; Reset mount flag for our job.
12         ;
13 006756 113703 0000000 MOVB   CORUSR,R3      ;Get our job index number
14 006762 004737 015112' CALL   CDJFLG        ;GET MOUNT FLAG FOR OUR JOB
15 006766 140312         BICB   R3,(R2)      ;RESET MOUNT FLAG FOR OUR JOB
16         ;
17         ; See if any other users still have device mounted.
18         ;
19 006770 010503         MOV     R5,R3      ;Point to bytes with mount flags
20 006772 062703 0000000 ADD     #CD$JOB,R3
21 006776 012700 0000000 MOV     #CD$#UB,R0   ;GET # BYTES WITH MOUNT FLAGS
22 007002 105723 1$:    TSTB   (R3)+  ;ANY OTHER JOBS HAVE THIS DEVICE MOUNTED?
23 007004 001042         BNE     9$      ;BR IF YES
24 007006 077003         SOB     R0,1$
25         ;
26         ; No other jobs have this device mounted.
27         ; Check if this is the SY disk.  If so, don't dismount it.
28         ;
29 007010 005765 0000000 TST     CD$BAS(R5)   ;IS THIS A LOGICAL DISK?
30 007014 001011         BNE     2$      ;BR IF YES
31 007016 016500 0000000 MOV     CD$DVU(R5),R0 ;GET DEVICE # AND UNIT #
32 007022 120037 0000000 CMPB   R0,SYINDEX   ;IS THIS SY DEVICE?
33 007026 001004         BNE     2$      ;BR IF NOT
34 007030 000300         SWAB   R0      ;GET UNIT # TO LOW-ORDER BYTE
35 007032 120037 0000010 CMPB   R0,SYUNIT+1   ;IS THIS SY UNIT?
36 007036 001425         BEQ     9$      ;BR IF YES -- DON'T DISMOUNT SY
37         ;
38         ; This is not SY device, do the actual dismount.
39         ; Tell data caching facility to clean out the data cache for this device.
40         ;
41 007040 005737 0000000 2$:    TST     CSHALC   ;Is data caching genned into system?
42 007044 001404         BEQ     5$      ;Br if not
43 007046 016503 0000000 MOV     CD$DVU(R5),R3 ;Get device and unit number for CSHCLN
44 007052 004777 0000000 CALL   @CSHCLN      ;Clean out the data cache
45         ;
46         ; Say device is no longer mounted
47         ;
48 007056 005065 0000000 5$:    CLR     CD$DVU(R5) ;SAY DEVICE IS NOT MOUNTED
49         ;
50         ; Remove file entries for this device from directory cache.
51         ;
52 007062 013700 0000000 MOV     CSHHD,R0    ;POINT TO FIRST CACHED DIRECTORY ENTRY
53 007066 020560 0000000 3$:    CMP     R5,FC$CDX(R0) ;IS FILE ON THIS DEVICE?
54 007072 001004         BNE     4$      ;BR IF NOT
55 007074 005060 0000000 CLR     FC$CDX(R0)  ;REMOVE ENTRY FROM CACHED DIRECTORY
56 007100 005060 0000000 CLR     FD$NAM(R0)
57 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
```

```

1          .SBTTL  MNTCOM -- MOUNT/DISMOUNT common setup
2          ;-----
3          ; Do common setup for Mount/Dismount EMT's.
4          ;
5          ; Outputs:
6          ;   USR data base locked for exclusive access.
7          ;   FILSPC = File spec passed with EMT.
8          ;   FILDVU = Device index # & Unit #.
9          ;
10         MNTCOM: MOV      R1, -(SP)
11         MOV      R4, -(SP)
12         ;
13         ; Gain exclusive access to USR data base.
14         ;
15         CALL     GETUSR          ;GET EXCLUSIVE USE OF USR
16         ;
17         ; Move file spec to FILSPC and perform any assigns.
18         ;
19         MOV      EMTBLK+2, R1    ;POINT TO DEVICE SPEC ARG
20         CALL     GETSPC         ;SET UP FILSPC
21         ;
22         ; Check to see if device is legal.
23         ;
24         CALL     CHKDEV          ;LOOK UP THE DEVICE NAME
25         BCC      1$             ;BR IF DEVICE IS OK
26         CALL     ERRNAM         ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
27         MOV      #-2, R0        ;INVALID DEVICE NAME
28         JMP      USRERR
29         ;
30         ; Build FILDVU word.
31         ;
32         1$:  MOVB   R4, FILDVU    ;SET DEVICE INDEX #
33         MOVB   R0, FILDVU+1     ;SET UNIT #
34         ;
35         ; Finished
36         ;
37         MOV      (SP)+, R4
38         MOV      (SP)+, R1
39         RETURN

```

```

1                                     .SBTTL  CPYMNT -- Copy mount entries from another job
2                                     ;-----
3                                     ;  CPYMNT is called to mount for the current job all of the devices
4                                     ;  mounted by another job.
5                                     ;
6                                     ;  Inputs:
7                                     ;  R2 = Job index number of job whose mounts are to be copied.
8                                     ;
9 007200 010246 CPYMNT: MOV      R2, -(SP)
10 007202 010346      MOV      R3, -(SP)
11 007204 010446      MOV      R4, -(SP)
12 007206 010546      MOV      R5, -(SP)
13 007210 010204      MOV      R2, R4          ; Save job index in R4
14                                     ;
15                                     ;  Begin to search through mount table for mounts for other job
16                                     ;
17 007212 013705 0000000 MOV     CSHDEV, R5          ; Point to start of mount table
18 007216 010403 1$:    MOV     R4, R3          ; Get # of job we are copying from
19 007220 004737 015112' CALL   CDJFLG          ; Get job flag for other job
20 007224 130312      BITB    R3, (R2)          ; Is this entry mounted by other job?
21 007226 001405      BEQ     2$          ; Br if not
22 007230 113703 0000000 MOVB   CORUSR, R3          ; Get our job index #
23 007234 004737 015112' CALL   CDJFLG          ; Get mount bit for our job
24 007240 150312      BISB   R3, (R2)          ; Say device is mounted by our job
25 007242 062705 0000000 2$:    ADD   #CD$$SZ, R5          ; Point to next mount table entry
26 007246 020537 0000000      CMP   R5, CSHDEV          ; Checked all entries?
27 007252 103761      BLO   1$          ; Br if not
28                                     ;
29                                     ;  Finished
30                                     ;
31 007254 012605      MOV   (SP)+, R5
32 007256 012604      MOV   (SP)+, R4
33 007260 012603      MOV   (SP)+, R3
34 007262 012602      MOV   (SP)+, R2
35 007264 000207      RETURN

```

CLRDIR -- Remove directory entries from dir cache

```

1          .SBTTL  CLRDIR -- Remove directory entries from dir cache
2          ;-----
3          ; This routine is called when a user-mode program writes to a directory
4          ; structured device that has been opened in non-file-structured mode.
5          ; It cleans out all directory cache entries for the device that is being
6          ; written to.
7          ;
8          ; Inputs:
9          ; R2 = Device index number
10         ; R3 = Pointer to Channel Status Block being used by write
11         ;
12 007266 010546 CLRDIR: MOV     R5,-(SP)
13 007270 013746 002014'      MOV     FILDVU,-(SP)      ;Save current device/file index info
14         ;
15         ; Set up device and unit numbers in FILDVU
16         ;
17 007274 110237 002014'      MOVB   R2,FILDVU      ;Set device index number
18 007300 116300 0000000     MOVB   C.DEVQ(R3),R0    ;Get device unit number
19 007304 042700 177770     BIC   #^C<7>,R0       ;Clear out all but unit number
20 007310 110037 002015'      MOVB   R0,FILDVU+1    ;Set unit number
21         ;
22         ; See if this device is being cached
23         ;
24 007314 004737 014776'      CALL   CSHTST         ;Is this device being cached?
25 007320 103414             BCS    9$              ;Br if not
26         ;
27         ; Clean out directory cache entries for this device
28         ;
29 007322 013700 0000000     MOV    CSHHD,R0        ;Point to 1st cached dir entry
30 007326 020560 0000000     1$:   CMP    R5,FC$CDX(R0) ;Is file on this device?
31 007332 001004             BNE    2$              ;Br if not
32 007334 005060 0000000     CLR    FC$CDX(R0)     ;Remove entry from directory cache
33 007340 005060 0000000     CLR    FD$NAM(R0)
34 007344 016000 0000000     2$:   MOV    FC$LNK(R0),R0 ;Chain to next cache entry
35 007350 001366             BNE    1$              ;Loop if more to check
36         ;
37         ; Finished
38         ;
39 007352 012637 002014'     9$:   MOV    (SP)+,FILDVU
40 007356 012605             MOV    (SP)+,R5
41 007360 000207             RETURN

```

```

1          .SBTTL  USRCOM --- Common setup
2          ;-----
3          ; USRCOM is called to perform the common setup operation for
4          ; .lookup, .enter, .delete and .rename emts.
5          ; It performs the following functions:
6          ; 1. Claim the USR data base for current user.
7          ; 2. Make sure channel is closed.
8          ; 3. Move file spec to FILSPC buffer and perform any assigns.
9          ; 4. Test that the specified device is legal.
10         ; 5. Mark channel open and set up device index and unit #.
11         ;
12         ; Inputs:
13         ; EMTBLK = Emt argument block.
14         ; CHNADR = Address of CSW for current channel.
15         ;
16         ; Outputs:
17         ; Channel area = Marked as open, device index # and unit # set up.
18         ; FILSPC = File spec after any assign performed.
19         ; FILDVU = Device index # in low-byte, unit # in high byte.
20         ; ASNSIZ = Size specified with ASSIGN command for logical device.
21         ;           0 if no size was specified with assign.
22         ; R0 = Unit number of device being accessed.
23         ; R3 = Address of CSW for channel.
24         ; R4 = Index into device table for device being opened.
25         ;
26 007362 010146 USRCOM: MOV      R1,-(SP)
27 007364 013703 0000000 MOV      CHNADR,R3      ;GET ADDRESS OF CURRENT CHANNEL AREA
28         ;
29         ; Claim USR data base for us
30         ;
31 007370 004737 007770' CALL     GETUSR        ;GAIN EXCLUSIVE ACCESS TO USR DATA BASE
32         ;
33         ; Make sure channel is closed now.
34         ;
35 007374 032763 0000000 0000000 BIT      #CS$OPN,C.CSW(R3); IS CHANNEL OPEN NOW?
36 007402 001403 BEQ      2$           ;BR IF NOT OPEN NOW
37 007404 005000 CLR      R0           ;RETURN ERROR CODE OF 0
38 007406 000137 015156' JMP      USRERR
39         ;
40         ; Move file spec to FILSPC and apply any assigns.
41         ;
42 007412 013701 0000020 2$:      MOV      EMTBLK+2,R1    ;GET ADDRESS OF FILE SPEC IN USER'S AREA
43 007416 042701 000001 BIC      #1,R1        ;MAKE SURE ADDRESS IS EVEN
44 007422 004737 007532' CALL     GETSPC        ;GET FILE SPEC TO FILSPC
45         ;
46         ; Now check to see if the device is legal.
47         ;
48 007426 004737 010212' CALL     CHKDEV        ; IS THE DEVICE LEGAL?
49 007432 103006 BCC      1$           ;BR IF RECOGNIZED DEVICE
50 007434 004737 015214' CALL     ERRNAM        ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
51 007440 012700 177776 MOV      #-2,R0       ;INVALID DEVICE
52 007444 000137 015156' JMP      USRERR
53         ; (At this point R4 has device table index and R0 has device unit #)
54         ; Set up FILDVU word.
55 007450 110437 002014' 1$:      MOVB     R4,FILDVU    ;SET DEVICE #
56 007454 110037 002015' MOVB     R0,FILDVU+1    ;SET UNIT #
57         ;

```

```
58 ; Set up information in channel.
59 ;
60 007460 110063 0000000 MOV  R0,C.DEVQ(R3) ;SET DEVICE UNIT #
61 007464 010463 0000000 MOV  R4,C.CSW(R3) ;SET DEVICE TABLE INDEX # IN CSW
62 007470 052763 0000000 0000000 BIS  #CS#OPN,C.CSW(R3);MARK CHANNEL AS OPEN
63 007476 005063 0000000 CLR  C.SBLK(R3) ;STARTING BLOCK # OF FILE
64 007502 005063 0000000 CLR  C.LENG(R3) ;ALLOCATED LENGTH OF FILE
65 007506 005063 0000000 CLR  C.USED(R3) ;NO BLOCKS WRITTEN TO FILE YET
66 007512 105063 0000000 CLR  C.NUMQ(R3) ;NO I/O OPERATIONS QUEUED ON CHANNEL YET
67 ;
68 ; See if this device is allocated to some other user
69 ;
70 007516 004737 010432' CALL  CHKALC ;Check for allocation problems
71 ;
72 ; See if we are authorized to access this device and file.
73 ;
74 007522 004737 010566' CALL  CHKACC ;CHECK FOR ACCESS AUTHORIZATION
75 ;
76 ; Return successfully
77 ;
78 007526 012601 MOV  (SP)+,R1
79 007530 000207 RETURN
```

GETSPC -- Get file spec from user's area

```

1          .SBTTL  GETSPC -- Get file spec from user's area
2          ;-----
3          ; GETSPC is called to move a file specification from the user's address
4          ; space to the FILSPC buffer area.
5          ;
6          ; Inputs:
7          ; R1 = Address of file spec in user's area.
8          ;
9          ; Outputs:
10         ; FILSPC = File spec after processing.
11         ; ASNSIZ = Size specified with ASSIGN for logical device (0 if not spec)
12         ;
13 007532 010146 GETSPC: MOV     R1,-(SP)
14 007534 010246         MOV     R2,-(SP)
15 007536 005037 002044' CLR     ASNSIZ          ;NO ASSIGN SIZE YET
16         ;
17         ; Copy the file specification from the user's area into FILSPC.
18         ;
19 007542 012702 0000000 MOV     #FILSPC,R2      ;Copy spec into FILSPC
20 007546 004737 007704' CALL    CPYSPC          ;Copy file spec from user's area
21         ;
22         ; Apply any assignments to this file spec.
23         ;
24 007552 013700 0000000 MOV     FILSPC,R0      ;GET DEVICE NAME
25 007556 001447         BEQ     4$          ;BR IF NULL NAME
26 007560 032737 0000000 0000000 BIT     #AF$BYA,RUNFLG ;Should we bypass logical assignment?
27 007566 001043         BNE     4$          ;Br if yes
28 007570 012702 0000000 MOV     #ASNTBL,R2     ;POINT TO START OF ASSIGN TABLE
29 007574 020062 0000000 2$:  CMP     R0,AT$LOG(R2) ;SEARCH FOR DEVICE NAME IN ASSIGN TABLE
30 007600 001406         BEQ     3$          ;BR IF FOUND
31 007602 062702 0000000 ADD     #AT$$SZ,R2     ;TRY NEXT ENTRY
32 007606 020227 0000000 CMP     R2,#ASNEND    ;DONE ALL?
33 007612 103770         BLO     2$          ;BR IF NOT
34 007614 000430         BR      4$          ;NAME WAS NOT ASSIGNED
35         ;
36         ; Name was assigned. Substitute assigned name.
37         ;
38 007616 012701 0000000 3$:  MOV     #FILSPC,R1      ;POINT TO AREA WITH FILE SPEC
39 007622 016211 0000000 MOV     AT$DEV(R2),(R1) ;PUT IN REAL DEVICE NAME
40 007626 016200 0000000 MOV     AT$FIL(R2),R0  ;WAS FILE NAME ASSIGNED?
41 007632 001421         BEQ     4$          ;BR IF NOT
42 007634 010061 0000002 MOV     R0,2(R1)       ;PUT IN FILE NAME
43 007640 016261 0000020 0000004 MOV     AT$FIL+2(R2),4(R1)
44 007646 016200 0000000 MOV     AT$EXT(R2),R0  ;WAS FILE EXTENSION ASSIGNED?
45 007652 001402         BEQ     8$          ;BR IF NOT
46 007654 010061 0000006 MOV     R0,6(R1)       ;PUT IN FILE EXT
47 007660 016200 0000000 8$:  MOV     AT$SIZ(R2),R0 ;WAS FILE SIZE ASSIGNED?
48 007664 001404         BEQ     4$          ;BR IF NOT
49 007666 010061 0000010 MOV     R0,8.(R1)     ;PUT IN FILE SIZE
50 007672 010037 002044' MOV     R0,ASNSIZ     ;RETURN ASSIGN SIZE IN ASNSIZ
51         ;
52         ; finished
53         ;
54 007676 012602 4$:  MOV     (SP)+,R2
55 007700 012601         MOV     (SP)+,R1
56 007702 000207         RETURN

```

```

1          .SBTTL  CPYSPC -- Copy file specification from user's area
2          ;-----
3          ; Copy a 5-word file specification from the user's area into a
4          ; system buffer.
5          ;
6          ; Inputs:
7          ; R1 = Address of file specification in user's area.
8          ; R2 = Address of 5-word buffer to receive specification.
9          ;
10         CPYSPC:
11         ;
12         ; Make sure address of file spec is legal
13         ;
14         007704 010100          MOV      R1,R0          ;GET ADDRESS OF FILE SPEC BUFFER
15         007706 004737 00000000 CALL    UACHKW        ;MAKE SURE ADDRESS IS LEGAL
16         007712 103004          BCC     5$            ;BR IF ADDRESS IS OK
17         007714 012700 177766    MOV     #-12,R0       ;GET ERROR ABORT CODE
18         007720 000137 015144'   JMP     USRCLS       ;ABORT THE JOB
19         ;
20         ; Copy file specification from user's area to FILSPC.
21         ;
22         007724 032737 00000000 00000000 5$:  BIT     #UMODE,EMTPS ;WAS EMT DONE IN USER OR KERNEL MODE?
23         007732 001410          BEQ     6$            ;BR IF KERNEL MODE
24         007734 013700 00000000          MOV     ODTBAS,R0     ;GET THE USER'S HIGH MEMORY LIMIT
25         007740 160100          SUB     R1,R0         ;SUBTRACT ADDRESS OF FILE SPEC AREA
26         007742 006200          ASR     R0            ;CONVERT TO NUMBER OF WORDS
27         007744 001410          BEQ     9$            ;BR IF NOTHING TO MOVE
28         007746 020027 0000004    CMP     R0,#4         ;COMPARE WITH SIZE OF FULL FILE SPEC
29         007752 101402          BLOS   1$            ;BR IF WE MUST MOVE LESS BECAUSE OF MEMORY TOP
30         007754 012700 0000004    6$:    MOV     #4.,R0      ;# WORDS TO MOVE
31         007760 106521          1$:    MFPD    (R1)+     ;GET A WORD FROM USER'S AREA
32         007762 012622          MOV     (SP)+,(R2)+   ;MOVE TO FILSPC
33         007764 077003          SOB     R0,1$
34         ;
35         ; Finished
36         ;
37         007766 000207          9$:    RETURN

```

GETUSR -- Claim usr data base for our job

```

1          .SBTTL  GETUSR -- Claim usr data base for our job
2          ;-----
3          ; GETUSR is called to gain exclusive access to the usr data base.
4          ; If the data base is currently in use by some other job, our job
5          ; is suspended until it is freed.
6          ;
7 007770 010546 GETUSR: MOV      R5, -(SP)
8          ;
9          ; See if we can access the USR
10         ;
11 007772      4$:  DISABL          ;;;DISABLE INTERRUPTS
12 010000 113700 0000000 MOVB   USRJOB, R0      ;;;IS USR AVAILABLE NOW?
13 010004 001426      BEQ    1$          ;;;BR IF YES
14 010006 120037 0000000 CMPB   R0, CORUSR      ;;;DO WE ALREADY OWN USR?
15 010012 001426      BEQ    2$          ;;;BR IF YES
16         ;
17         ; Someone else owns the USR now.
18         ; Suspend our job until we can get it.
19         ;
20 010014      ENABL          ;ENABLE INTERRUPTS
21 010022 113705 0000000 MOVB   EXCJOB, R5      ;REMEMBER IF WE HAVE EXCLUSIVE SYS USE
22 010026 105037 0000000 CLRB   EXCJOB          ;ENABLE OTHER JOBS TO RUN
23 010032 012700 0000000 MOV    #S$QUSR, R0     ;PUT US IN QUEUE FOR USR
24 010036 004737 0000000 CALL   QNSPNX          ;SUSPEND JOB AND WAIT FOR USR
25 010042 004737 0000000 CALL   CHKABT          ;WERE WE ABORTED WHILE ASLEEP?
26 010046 020537 0000000 CMP    R5, CORUSR      ;DO WE WANT EXCLUSIVE ACCESS TO SYSTEM?
27 010052 001347      BNE    4$          ;LOOP IF NOT
28 010054 110537 0000000 MOVB   R5, EXCJOB      ;GET EXCLUSIVE ACCESS TO THE SYSTEM
29 010060 000744      BR     4$          ;GO TRY TO GET USR NOW
30         ;
31         ; We can get USR now.
32         ;
33 010062 113737 0000000 0000000 1$:  MOVB   CORUSR, USRJOB ;;;CLAIM USR FOR US
34 010070      2$:  ENABL          ;ENABLE INTERRUPTS
35 010076 005237 002026'  INC    USRCNT      ;REMEMBER # TIMES WE CLAIMED USR
36         ;
37         ; Make sure USR is not being called from a completion routine.
38         ;
39 010102 105737 0000000      TSTB   CURCP          ;ARE WE IN A COMPLETION ROUTINE NOW?
40 010106 001404      BEQ    3$          ;BR IF NOT
41 010110 012700 177765      MOV    #-13, R0      ;ABORT IF IN COMPLETION ROUTINE
42 010114 000137 015156'  JMP    USRERR
43         ;
44         ; Initialize USR data base.
45         ;
46 010120 012737 177777 002002' 3$:  MOV    #-1, CURSEC      ;NO DIR SEG IN BUFFER NOW
47         ;
48         ; Finished
49         ;
50 010126 012605      MOV    (SP)+, R5
51 010130 000207      RETURN
52         ;
53         .SBTTL  FREUSR -- Free the USR
54         ;-----
55         ; FREUSR is called to release our ownership of the USR data base.
56         ;
57 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$QUSR,RO     ; QUEUE OF WAITING JOBS
68 010174 004737 0000000      CALL   UREGD          ; RESTART WAITING JOBS
69 010200 000207          RETURN
70                                     ;
71                                     ; Finished
72                                     ;
73 010202          1$: ENABL          ; ENABLE INTERRUPTS
74 010210 000207          RETURN
```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 010212 010146
15 010214 010246
16
17
18
19 010216 013701 0000000
20 010222 005000
21 010224 071027 000050
22 010230 012702 177777
23 010234 005701
24 010236 001406
25 010240 162701 000036
26 010244 010102
27 010246 020227 000007
28 010252 101063
29 010254 070027 000050
30
31
32
33
34
35
36
37 010260 020137 002020'
38 010264 001403
39 010266 020137 002016'
40 010272 001007
41 010274 013704 0000000
42 010300 005702
43 010302 002014
44 010304 113702 0000010
45 010310 000411
46
47
48
49 010312 013704 0000000
50 010316 020164 0000000
51 010322 001404
52 010324 162704 000002
53 010330 002372
54 010332 000433
55
56
57

```

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.

```

CHKDEV: MOV     R1, -(SP)
        MOV     R2, -(SP)

```

; Get device name and split off unit number.

```

        MOV     FILSPC, R1      ; GET DEVICE NAME (RAD50)
        CLR     R0              ; SET FOR DIVIDE
        DIV     #50, R0         ; SPLIT OFF LOW-ORDER RAD50 CHARACTER
        MOV     #-1, R2         ; ASSUME NO UNIT NUMBER SPECIFIED
        TST     R1              ; WAS A UNIT NUMBER SPECIFIED?
        BEQ     6$,             ; BR IF NOT
        SUB     #36, R1         ; CONVERT RAD50 DIGIT TO BINARY VALUE
        MOV     R1, R2          ; GET BINARY VALUE OF UNIT NUMBER
        CMP     R2, #7          ; RESTRICT UNIT NUMBER TO RANGE 0-7
        BHI     8$,             ; BR IF INVALID UNIT NUMBER
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.

```

        CMP     R1, R50DK       ; IS DEVICE NAME "DK"?
        BEQ     2$,             ; BR IF YES
        CMP     R1, R50SY       ; IS DEVICE NAME "SY"?
        BNE     3$,             ; BR IF NOT
2$:     MOV     SYINDX, R4       ; GET SY DEVICE INDEX NUMBER
        TST     R2              ; DID USER SPECIFY A UNIT NUMBER?
        BGE     7$,             ; BR IF YES
        MOVB   SYUNIT+1, R2     ; GET SYSTEM DEVICE UNIT NUMBER
        BR     7$,

```

; Look up device name in permanent device name table.

```

3$:     MOV     NUMDEV, R4       ; GET INDEX NUMBER OF LAST DEVICE
5$:     CMP     R1, PNAME(R4)    ; SEARCH FOR NAME IN TABLE
        BEQ     7$,             ; BR IF FOUND
        SUB     #2, R4          ; TRY NEXT ENTRY
        BGE     5$,             ; LOOP IF MORE TO CHECK
        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

```

CHKALC -- Check for device allocation

```

1          .SBTTL  CHKALC -- Check for device allocation
2          ;-----
3          ;  CHKALC is called to determine if a device being accessed is available
4          ;  to the current job due to allocation considerations.
5          ;  If an allocation failure is detected, a fatal error return is taken.
6          ;
7          ;  Inputs:
8          ;  FILDVU = Device index # and unit # of device to be checked.
9          ;
10         010432  010046  CHKALC:  MOV      R0,-(SP)
11         010434  010246          MOV      R2,-(SP)
12         ;
13         ;  Don't do allocation test for TT
14         ;
15         010436  105737  002014'  TSTB    FILDVU      ;Device # 0 ==> TT
16         010442  001446          BEQ     9$           ;Br if this device is TT
17         ;
18         ;  See if this device is in the allocation table.
19         ;
20         010444  012702  0000000  MOV     #ALCTBL,R2   ;Point to start of allocation table
21         010450  013700  002014'  MOV     FILDVU,R0    ;Get device/unit id
22         010454  020062  0000000  1$:    CMP     R0,AD$DVU(R2) ;Is this entry for this device?
23         010460  001421          BEQ     2$           ;Br if yes
24         010462  062702  0000000  ADD     #AD$SZ,R2    ;Point to next table entry
25         010466  020227  0000000  CMP     R2,#ALCEND   ;Have we checked all entries?
26         010472  103770          BLO    1$           ;Br if not
27         ;
28         ;  This device is not in the allocation table.
29         ;  See if it must be allocated before it can be accessed.
30         ;
31         010474  113700  002014'  MOVB   FILDVU,R0    ;Get the device index number
32         010500  032760  0000000  0000000  BIT     #DX$RAL,DVFLAG(R0);Is allocation required before access?
33         010506  001424          BEQ     9$           ;Br if not
34         010510  004737  015214'  CALL   ERRNAM       ;Set file spec for Kmon error message
35         010514  012700  177751    MOV     #-27,R0      ;Set error code
36         010520  000137  015144'  JMP    USRCLS       ;Error return from USR
37         ;
38         ;  This device is in the allocation table.
39         ;  See if it is allocated to our job or to another job.
40         ;
41         010524  116202  0000000  2$:    MOVB   AD$JOB(R2),R2 ;Get # of job that owns the device
42         010530  113700  0000000  MOVB   CORUSR,R0    ;Get current job index number
43         010534  126260  0000000  0000000  CMPB   LNPRIM(R2),LNPRIM(R0);Do we own the device?
44         010542  001406          BEQ     9$           ;Br if yes
45         010544  004737  015214'  CALL   ERRNAM       ;Set file spec for Kmon error message
46         010550  012700  177752    MOV     #-26,R0      ;Set error code
47         010554  000137  015144'  JMP    USRCLS       ;Error return
48         ;
49         ;  Device access is ok
50         ;
51         010560  012602  9$:    MOV     (SP)+,R2
52         010562  012600          MOV     (SP)+,R0
53         010564  000207          RETURN

```

```

1          .SBTTL  CHKACC -- Check legality of file access
2          ;-----
3          ;  CHKACC is called to see if a user is authorized to access a
4          ;  certain device and file.
5          ;
6          ;  Inputs:
7          ;  R0 = Unit number of device being accessed.
8          ;  R4 = Device index number of device being accessed.
9          ;  FILSPC = Device-file specification of file being accessed.
10         ;  R3 = Address of channel block associated with operation.
11         ;
12         ;  Outputs:
13         ;  Abort if access denied.
14         ;  CS#RON set in channel CSW if read-only access authorized.
15         ;
16 010566 010046  CHKACC:  MOV     R0,-(SP)
17 010570 010146      MOV     R1,-(SP)
18 010572 010246      MOV     R2,-(SP)
19 010574 010346      MOV     R3,-(SP)
20 010576 010446      MOV     R4,-(SP)
21 010600 010546      MOV     R5,-(SP)
22 010602 113701 0000000  MOVB   CORUSR,R1      ;GET JOB INDEX NUMBER
23         ;
24         ;  See if we are accessing a logical disk with NOWRITE attribute set.
25         ;
26 010606 020437 0000000      CMP     R4,LDDEVX      ;IS THIS A LOGICAL DISK?
27 010612 001011      BNE     13$          ;BR IF NOT
28 010614 010002      MOV     R0,R2          ;GET UNIT NUMBER
29 010616 006302      ASL     R2          ;CONVERT TO WORD TABLE INDEX
30 010620 032762 0000000 0000000  BIT     #LD#RON,LDFLAG(R2) ;WAS /NOWRITE SPECIFIED FOR DISK?
31 010626 001403      BEQ     13$          ;BR IF NOT
32 010630 052763 0000000 0000000  BIS     #CS#RON,C.CSW(R3);SAY READ-ONLY ACCESS ALLOWED
33         ;
34         ;  Allow access to all devices if user has BYPASS privilege
35         ;
36 010636 032737 0000000 0000000 13$:  BIT     #PO#BYP,PRIVCO ;Does user have BYPASS privilege?
37 010644 001004      BNE     19$          ;Br if yes
38         ;
39         ;  See if TSXUCL is trying to access its command file
40         ;
41 010646 032761 0000000 0000000  BIT     #UCLRN,LSW7(R1); IS TSXUCL RUNNING?
42 010654 001402      BEQ     16$          ;Br if not
43 010656 000137 011224' 19$:  JMP     9$          ;Allow full file access
44         ;
45         ;  See if a non-privileged user is trying to access a SYS or TSX file on SY:
46         ;
47 010662 032737 0000000 0000000 16$:  BIT     #PO#SYS,PRIVCO ; IS THIS A PRIVILEGED USER?
48 010670 001022      BNE     1$          ;BR IF PRIVILEGED USER
49 010672 032761 0000000 0000000  BIT     #N0IN,LSW3(R1) ;ARE WE RUNNING IN START-UP COMMAND FILE?
50 010700 001016      BNE     1$          ;BR IF YES -- GIVE FULL ACCESS DURING STARTUP
51 010702 120437 0000000      CMPB   R4,SYINDEX      ;Is device = SY?
52 010706 001013      BNE     1$          ;Br if not SY
53 010710 120037 0000010      CMPB   R0,SYUNIT+1    ;Is unit = SY?
54 010714 001010      BNE     1$          ;Br if not
55 010716 013702 0000060      MOV     FILSPC+6,R2  ;GET FILE EXTENSION
56 010722 020237 002022'      CMP     R2,R50SYS    ; IS EXTENSION "SYS"?
57 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 0000020      1$:     TST      FILSPC+2      ; Non-file-structured lookup?
64 010742 001017              BNE     15$              ; Br if not
65 010744 032764 0000000 0000000  BIT     #DS$DIR,DVSTAT(R4) ; Is this a directory structured device?
66 010752 001413              BEQ     15$              ; Br if not
67 010754 032737 0000000 0000000  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 0000000 0000000  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 0000000 0000000  BIS     #CS$RON,C.CSW(R3) ; Force read-only access
72                               ;
73                               ; See if ACCESS command was used to restrict access to any devices or files.
74                               ;
75 011002 005737 0000000      15$:    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'      MOV     R0,CKAUNT          ; SAVE DEVICE UNIT #
82 011014 110437 002046'      MOV     R4,CKADEV          ; SAVE DEVICE INDEX NUMBER
83 011020 005002              CLR     R2                  ; SET NO-ACCESS STATE
84 011022 012705 0000000      MOV     #DKFILE,R5         ; POINT TO START OF ALLOWED DEVICE SPECS
85                               ; Check next authorized file spec.
86 011026 010504      6$:     MOV     R5,R4          ; GET ADDRESS OF FILE SPEC AREA
87                               ; Check device number and unit number.
88 011030 116400 0000000      MOV     OF$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'      CMP     R0,CKADEV          ; DOES IT MATCH THAT OF FILE SPEC BEING OPENED?
92 011044 001026              BNE     4$                  ; BR IF NOT
93 011046 123764 002047' 0000000  CMP     CKAUNT,OF$UNT(R4) ; COMPARE DEVICE UNIT NUMBERS
94 011054 001022              BNE     4$                  ; BR IF MISMATCH
95                               ; Check file names.
96 011056 062704 0000000      2$:     ADD     #OF$FIL,R4      ; POINT TO FILE NAME
97 011062 012701 0000020      MOV     #FILSPC+2,R1      ; POINT TO FILE NAME IN SPEC BEING OPENED
98 011066 012700 0000003      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 0000000  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 0000000 0000000  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 0000000      4$:     ADD     #OF$$SZ,R5      ; POINT TO NEXT AUTHORIZED FILE SPEC
111 011126 020527 0000000      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.

```

```

115 ;
116 ; Allow implicit access to some devices and files.
117 ; Allow full access to TT.
118 ;
119 011134 105737 002046' TSTB CKADEV ;TT DEVICE INDEX IS ALWAYS ZERO
120 011140 001431 BEQ 9# ;BR IF TT
121 ;
122 ; See if we were granted read-only access.
123 ;
124 011142 005702 TST R2 ;WERE WE GRANTED READ-ONLY ACCESS TO FILE?
125 011144 001024 BNE 7# ;BR IF YES
126 ;
127 ; Allow DUP to access SY: on a read-only basis
128 ;
129 011146 113701 0000000 MOV B CORUSR,R1 ;GET JOB INDEX NUMBER
130 011152 032761 0000000 0000000 BIT #DUPRN,LSW6(R1); IS DUP RUNNING?
131 011160 001410 BEQ 8# ;BR IF NOT
132 011162 123737 002046' 0000000 CMPB CKADEV,SYINDX ; IS DEVICE = SY?
133 011170 001004 BNE 8# ;BR IF NOT
134 011172 123737 002047' 0000010 CMPB CKAUNT,SYUNIT+1 ; IS UNIT = SY?
135 011200 001406 BEQ 7# ;BR IF SY -- ALLOW READ ONLY ACCESS
136 ;
137 ; User is not allowed to access this device or file.
138 ;
139 011202 004737 015214' 8#: CALL ERRNAM ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
140 011206 012700 177764 MOV #-14,R0 ;SET ERROR CODE
141 011212 000137 015144' JMP USRCLS ;ABORT
142 ;
143 ; User is allowed read-only access to this device or file.
144 ; Set CS#RON flag in CSW.
145 ;
146 011216 052763 0000000 0000000 7#: BIS #CS#RON,C.CSW(R3);SET READ-ONLY ACCESS FLAG IN CSW
147 ;
148 ; User is allowed full access to file.
149 ;
150 011224 012605 9#: MOV (SP)+,R5
151 011226 012604 MOV (SP)+,R4
152 011230 012603 MOV (SP)+,R3
153 011232 012602 MOV (SP)+,R2
154 011234 012601 MOV (SP)+,R1
155 011236 012600 MOV (SP)+,R0
156 011240 000207 RETURN

```

```

1          .SBTTL  FNDFIL --- Find file in directory
2          ;-----
3          ; FNDFIL is called to locate the directory entry for a permanent file.
4          ;
5          ; Inputs:
6          ;   FILSPC = File spec to be searched for.
7          ;
8          ; Outputs:
9          ;   RO = Starting block number of file.
10         ;   R1 = Address of directory entry (located in USRBUF).
11         ;   USRBUF = Directory segment containing entry.
12         ;   CURSEG = Number of directory segment.
13         ;   C-flag set if file not found.
14         ;
15 011242 010246 FNDFIL: MOV     R2,-(SP)
16 011244 010346         MOV     R3,-(SP)
17         ;
18         ; Read in directory segment number 1 and set up info about directory.
19         ;
20 011246 004737 013454' CALL    RDSEG1          ;READ IN FIRST DIRECTORY SEGMENT
21         ;
22         ; Search for permanent file entry that matches our name.
23         ;
24 011252 012700 0000000 2$:    MOV     #FS$PRM,R0      ;LOCATE NEXT PERM FILE ENTRY
25 011256 004737 013166'     CALL    GETDIR
26 011262 103415             BCS     3$                ;BR IF HIT END OF DIRECTORY
27         ; Compare file names.
28 011264 010103             MOV     R1,R3                ;GET ADDRESS OF DIRECTORY ENTRY
29 011266 062703 0000000     ADD     #FD$NAM,R3        ;POINT TO FILE NAME IN DIR ENTRY
30 011272 012702 0000020     MOV     #FILSPC+2,R2     ;POINT TO NAME WE WANT TO FIND
31 011276 012700 0000003     MOV     #3,R0           ;COMPARE 3 WORDS
32 011302 022223 1$:    CMP     (R2)+,(R3)+        ;DO NAMES MATCH?
33 011304 001362             BNE     2$                ;BR IF NOT
34 011306 077003             SOB     R0,1$          ;CHECK ALL OF NAME
35         ;
36         ; Found the file entry -- Calculate starting block number.
37         ;
38 011310 004737 013420'     CALL    SBCALC          ;CALCULATE STARTING BLOCK # FOR ENTRY
39         ;
40         ; Finished
41         ;
42 011314 000241 4$:    CLC                ;SIGNAL GOOD RETURN
43 011316 012603 3$:    MOV     (SP)+,R3
44 011320 012602         MOV     (SP)+,R2
45 011322 000207         RETURN

```

```

1          .SBTTL  FNDFRE -- Find a free slot for a new file
2          ;-----
3          ; FNDFRE is called to locate a free file entry for a new file.
4          ;
5          ; Inputs:
6          ;   R2 = Desired size (-1==>largest; 0==>Default algorithm)
7          ;   FILSPC = Device-file spec for file being entered.
8          ;
9          ; Outputs:
10         ;   R0 = Error code if C-flag set on return.
11         ;       (1==>No free space of adequate size; 3==>Protected file conflict).
12         ;   R1 = Address of free file directory entry.
13         ;   R2 = Actual # of blocks to be used for file.
14         ;   USRBUF = Directory segment that has free slot entry.
15         ;   C-flag set if insufficient disk space or protected file conflict.
16         ;
17 011324 010346 FNDFRE: MOV     R3,-(SP)
18 011326 010446      MOV     R4,-(SP)
19         ;
20         ; Initialize tables that hold info about 2 largest free slots.
21         ;
22 011330 005037 002030'      CLR     L1SIZ      ;SIZE OF LARGEST FREE SLOT
23 011334 005037 002036'      CLR     L2SIZ      ;SIZE OF 2ND LARGEST FREE SLOT
24         ;
25         ; Read in 1st directory segment and set up info about directory parameters.
26         ;
27 011340 004737 013454'      CALL    RDSEG1      ;READ IN 1ST DIRECTORY SEGMENT
28         ;
29         ; Consolidate entries in directory segment.
30         ;
31 011344 004737 012634'      7$:    CALL    CONSOL      ;CONSOLIDATE DIRECTORY SEGMENT
32         ;
33         ; Find next directory entry for a permanent file or a free slot.
34         ;
35 011350 063701 002006'      4$:    ADD     DIRSIZ,R1      ;POINT TO NEXT DIRECTORY ENTRY
36 011354 016100 000000G      MOV     FD$STA(R1),R0 ;PICK UP STATUS WORD FROM DIRECTORY ENTRY
37 011360 032700 000000C      BIT     #FS$PRM+FS$EMP+FS$EDS,R0;PERM FILE, EMPTY SLOT OR END OF SEGMENT?
38 011364 001771              BEQ     4$              ;BR IF NONE OF ABOVE
39 011366 032700 000000G      BIT     #FS$EMP,R0      ;IS THIS AN EMPTY SPACE ENTRY?
40 011372 001024              BNE     13$             ;BR IF YES
41 011374 032700 000000G      BIT     #FS$EDS,R0      ;IS THIS THE END OF SEGMENT MARKER ENTRY?
42 011400 001106              BNE     1$              ;BR IF YES
43         ;
44         ; Found permanent file entry. See if this is a protected file entry
45         ; with the same name as the file being entered.
46         ;
47 011402 032700 000000G      BIT     #FS$PRO,R0      ;IS THIS FILE PROTECTED?
48 011406 001760              BEQ     4$              ;BR IF NOT
49 011410 010103              MOV     R1,R3              ;POINT TO NAME IN DIR ENTRY
50 011412 062703 000000G      ADD     #FD$NAM,R3
51 011416 012704 000002G      MOV     #FILSPC+2,R4      ;POINT TO NAME OF FILE BEING ENTERED
52 011422 012700 000003      MOV     #3,R0           ;GET # WORDS TO COMPARE
53 011426 022324      14$:    CMP     (R3)+,(R4)+      ;COMPARE FILE NAMES
54 011430 001347              BNE     4$              ;BR IF DIFFERENT
55 011432 077003              SOB     R0,14$          ;COMPARE ALL OF NAMES
56         ; Error: Protected file with same name.
57         ; 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 0000000        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      L1$IZ,R3     ;GET SIZE OF BEST FIT SO FAR
72 011472 001402                BEQ     15$          ;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                BHS     4$           ;BR IF NOT
75 011500 010037 002030'        15$:    MOV      R0,L1$IZ    ;REMEMBER SIZE OF BEST FIT FOUND SO FAR
76 011504 010137 002032'        MOV      R1,L1$OFF   ;OFFSET OF DIR ENTRY WITHIN SEGMENT
77 011510 013737 002002' 002034' MOV      CURSEG,L1$SEG ;# 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$:    CMP      R0,L1$IZ    ;IS NEW SLOT LARGER THAN LARGEST WE'VE SEEN?
82 011524 101421                BLOS   5$           ;BR IF NOT
83 011526 013737 002030' 002036' MOV      L1$IZ,L2$IZ  ;MOVE INFO INTO 2ND LARGEST ENTRY
84 011534 013737 002034' 002042' MOV      L1$SEG,L2$SEG
85 011542 013737 002032' 002040' MOV      L1$OFF,L2$OFF
86 011550 010037 002030'        MOV      R0,L1$IZ    ;SAVE SIZE OF NEW LARGEST ENTRY
87 011554 010137 002032'        MOV      R1,L1$OFF   ;SAVE ADDRESS OF DIR ENTRY
88 011560 013737 002002' 002034' MOV      CURSEG,L1$SEG ;SAVE # OF SEGMENT WITH ENTRY
89 011566 000670                BR      4$           ;KEEP LOOKING FOR LARGER FREE SLOTS
90 011570 020037 002036'        5$:    CMP      R0,L2$IZ    ;IS NEW SLOT LARGER THAN 2ND LARGEST SO FAR?
91 011574 101665                BLOS   4$           ;BR IF NOT
92 011576 010037 002036'        MOV      R0,L2$IZ    ;SAVE NEW 2ND LARGEST SIZE
93 011602 010137 002040'        MOV      R1,L2$OFF   ;SAVE ADDRESS OF DIR ENTRY
94 011606 013737 002002' 002042' MOV      CURSEG,L2$SEG ;SAVE DIR SEGMENT #
95 011614 000655                BR      4$           ;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 0000000        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$:    TST      L1$IZ      ;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

```

```

115 011652 013702 002030'      MOV      L1SIZ,R2      ;RETURN SIZE OF LARGEST FREE SLOT
116 011656 013700 002034'      10$:    MOV      L1SEG,R0      ;GET SEGMENT # WITH ENTRY
117 011662 013701 002032'      MOV      L1OFF,R1     ;GET ADDRESS OF ENTRY
118 011666 000435              BR       23$          ;GO REREAD SEGMENT WITH EMPTY ENTRY
119                          ; User wants default allocation.
120                          ; (i.e., Larger of (1/2 largest slot or 2nd largest slot) )
121 011670 013702 002030'      8$:    MOV      L1SIZ,R2      ;GET SIZE OF LARGEST SLOT
122 011674 020227 000001      CMP      R2,#1        ;IS LARGEST SLOT 1 BLOCK LONG?
123 011700 001414              BEQ      11$          ;IF SO THEN USE ALL OF IT
124 011702 000241              CLC                    ;DIVIDE BY 2
125 011704 006002              ROR      R2
126 011706 020237 002036'      CMP      R2,L2SIZ     ;COMPARE TO 2ND LARGEST SLOT
127 011712 101007              BHI      11$          ;USE 1/2 OF LARGEST
128 011714 013700 002042'      MOV      L2SEG,R0     ;GET # OF SEGMENT WITH 2ND LARGEST FREE BLOCK
129 011720 013701 002040'      MOV      L2OFF,R1     ;GET OFFSET TO ENTRY WITHIN SEGMENT
130 011724 013702 002036'      MOV      L2SIZ,R2     ;GET SIZE OF ENTRY
131 011730 000404              BR       12$
132 011732 013700 002034'      11$:    MOV      L1SEG,R0      ;GET # OF SEGMENT WITH LARGEST FREE BLOCK
133 011736 013701 002032'      MOV      L1OFF,R1     ;GET OFFSET TO ENTRY WITHIN SEGMENT
134                          ;
135                          ; Constrain the largest file size returned in response to a 0-size request
136                          ; to a sysgen supplied parameter (MAXFIL).
137                          ;
138 011742 005737 0000000      12$:    TST      VMXFIL      ;Did user specify a size to constrain alloc?
139 011746 001405              BEQ      23$          ;Br if not
140 011750 020237 0000000      CMP      R2,VMXFIL     ;IS FREE SLOT LARGER THAN MAXFIL PARAMETER?
141 011754 101402              BLOS    23$          ;BR IF NOT
142 011756 013702 0000000      MOV      VMXFIL,R2     ;ONLY USE THIS MUCH SPACE
143                          ;
144                          ; Reread and consolidate the segment that has the empty entry we
145                          ; are going to use unless that segment is the current segment.
146                          ;
147 011762 020037 002002'      23$:    CMP      R0,CURSEG     ;IS SEG WE WANT THE CURRENT SEGMENT?
148 011766 001413              BEQ      21$          ;BR IF YES
149 011770 010146              MOV      R1,-(SP)      ;SAVE ADDRESS OF DIRECTORY ENTRY
150 011772 004737 013570'      CALL    RDSEG         ;REREAD THE SEGMENT WITH THE EMPTY ENTRY
151 011776 012601              MOV      (SP)+,R1     ;GET ADDRESS OF DIRECTORY ENTRY
152 012000 004737 012634'      CALL    CONSOL        ;CONSOLIDATE IT
153 012004 000404              BR       21$          ;RETURN SUCCESSFULLY
154                          ;
155                          ; We could not satisfy the request.
156                          ;
157 012006 012700 000001      9$:    MOV      #1,R0      ;RETURN ERROR CODE 1
158 012012 000261      24$:    SEC                    ;SIGNAL ERROR ON RETURN
159 012014 000401              BR       22$
160                          ;
161                          ; Successful return
162                          ;
163 012016 000241      21$:    CLC                    ;SIGNAL GOOD RETURN
164 012020 012604      22$:    MOV      (SP)+,R4
165 012022 012603      MOV      (SP)+,R3
166 012024 000207      RETURN

```

ADDENT -- Add a tentative file entry to directory

```

1          .SBTTL  ADDENT -- Add a tentative file entry to directory
2          ;-----
3          ; ADDENT is called to add a tentative file entry to the current
4          ; directory segment.  This may cause the directory segment to be
5          ; split if it overflows.
6          ;
7          ; Inputs:
8          ; R1 = Address of empty-file directory entry to be converted to tentative file.
9          ; R2 = # blocks to be allocated to tentative file.
10         ; FILSPC = File spec for file being created.
11         ;
12         ; Outputs:
13         ; R1 = Address of tentative file directory entry.
14         ; C-flag set if directory overflows.
15         ;
16 012026 010246 ADDENT: MOV      R2,-(SP)
17 012030 010346      MOV      R3,-(SP)
18         ;
19         ; See if there is room in the current directory segment for three new entries
20         ; (free-0, tentative, free-remainder)
21         ;
22 012032 010146      MOV      R1,-(SP)      ; SAVE ADDRESS OF ENTRY WE ARE WORKING ON
23 012034 005000      CLR      R0          ; FIND END OF SEGMENT ENTRY
24 012036 004737 013222' CALL    NXTDIR
25 012042 010103      MOV      R1,R3      ; GET ADDRESS OF END OF SEGMENT ENTRY
26 012044 012601      MOV      (SP)+,R1    ; GET BACK ADDRESS OF EMPTY FILE ENTRY
27 012046 013700 002006' MOV      DIRSIZ,R0    ; GET SIZE OF A DIRECTORY ENTRY
28 012052 060003      ADD      R0,R3      ; ADD SPACE NEEDED FOR 3 ENTRIES
29 012054 060003      ADD      R0,R3
30 012056 060003      ADD      R0,R3
31 012060 020327 002002' CMP      R3,#USRBUF+1024. ; IS THERE ROOM IN THIS SEGMENT?
32 012064 103403      BLO     1$          ; BR IF THERE IS ROOM
33         ;
34         ; Directory segment is full.
35         ; Attempt to split it.
36         ;
37 012066 004737 012326' CALL    SPLIT    ; SPLIT THE SEGMENT
38 012072 103461      BCS     9$          ; BR IF DIRECTORY OVERFLOW
39         ;
40         ; There is room in the current directory segment for the new entries.
41         ; Convert the empty file entry to a tentative file entry.
42         ;
43 012074 016100 000000G 1$:  MOV      FD$LEN(R1),R0    ; GET SIZE OF EMPTY FILE ENTRY
44 012100 010261 000000G  MOV      R2,FD$LEN(R1)    ; PUT IN ALLOCATED SIZE OF TENTATIVE FILE
45 012104 160200      SUB      R2,R0      ; CALC # BLOCKS LEFT OVER IN EMPTY AREA
46 012106 010002      MOV      R0,R2
47 012110 012761 000000G 000000G MOV      #FS$TEN,FD$STA(R1); MARK ENTRY AS TENTATIVE
48 012116 113761 000000G 000000G MOVVB   CORUSR,FD$JOB(R1); PUT IN JOB NUMBER
49 012124 113761 000000G 000000G MOVVB   CHNNUM,FD$CHN(R1); PUT IN CHANNEL NUMBER
50 012132 005061 000000G      CLR      FD$DAT(R1)    ; SAY NO CREATION DATE YET -- SET IN CLOSE
51 012136 010103      MOV      R1,R3      ; SAVE ADDRESS OF TENTATIVE ENTRY
52 012140 062703 000000G      ADD      #FD$NAM,R3    ; POINT TO FIELD WHERE NAME GOES
53 012144 012700 000002G      MOV      #FILSPC+2,R0    ; POINT TO FILE SPEC WE ARE CREATING FOR
54 012150 012023      MOV      (R0)+,(R3)+    ; MOVE IN FILE NAME
55 012152 012023      MOV      (R0)+,(R3)+
56 012154 012023      MOV      (R0)+,(R3)+    ; AND EXTENSION
57         ;

```

ADDENT -- Add a tentative file entry to directory

```

58      ; Add an empty file entry following the tentative file entry to hold
59      ; any left over blocks.
60      ;
61 012156 063701 002006'      ADD      DIRSIZ,R1      ;POINT TO FOLLOWING DIR ENTRY
62 012162 004737 012244'      CALL     INSERT      ;INSERT AN EMPTY FILE ENTRY
63 012166 010261 0000000     MOV      R2,FD#LEN(R1) ;FILL IN # LEFT OVER BLOCKS
64      ;
65      ; If the empty file entry we turned into a tentative file immediately followed
66      ; a tentative file entry, add a zero length empty file entry in front of
67      ; our new tentative file entry (this entry may be used when the earlier
68      ; tentative file entry is closed).
69      ;
70 012172 163701 002006'      SUB      DIRSIZ,R1      ;POINT TO OUR NEW TENTATIVE FILE ENTRY
71 012176 163701 002006'      SUB      DIRSIZ,R1      ;POINT TO PREVIOUS ENTRY
72 012202 020127 0000000     CMP      R1,#USRBUF+DH##SZ;WAS THERE A PREVIOUS ENTRY?
73 012206 103410              BLO     3$              ;BR IF NOT
74 012210 032761 0000000 0000000 BIT     #FS#TEN,FD#STA(R1);WAS PREVIOUS ENTRY A TENTATIVE FILE ENTRY?
75 012216 001404              BEQ     3$              ;BR IF NOT
76 012220 063701 002006'      ADD      DIRSIZ,R1      ;POINT BACK TO OUR NEW TENTATIVE ENTRY
77 012224 004737 012244'      CALL     INSERT      ;INSERT A ZERO-LENGTH EMPTY FILE ENTRY
78 012230 063701 002006'      3$:    ADD      DIRSIZ,R1      ;POINT TO TENTATIVE FILE ENTRY WE CREATED
79      ;
80      ; Finished
81      ;
82 012234 000241              CLC                      ;SIGNAL GOOD RETURN
83 012236 012603              9$:    MOV      (SP)+,R3
84 012240 012602              MOV      (SP)+,R2
85 012242 000207              RETURN

```

INSERT -- Add an empty file entry to directory

```

1          .SBTTL  INSERT -- Add an empty file entry to directory
2          ;-----
3          ; INSERT is called to create a new empty file directory entry and insert
4          ; it into the current directory segment.
5          ; Note: Tests before calling us guarantee that there is room in the
6          ;       current directory segment.
7          ;
8          ; Inputs:
9          ;   R1 = Address of entry new empty file entry is to go in front of.
10         ;
11         ; Outputs:
12         ;   R1 = Address of new empty file entry.
13         ;
14 012244 010146  INSERT: MOV      R1, -(SP)
15         ;
16         ;   Locate end-of-segment entry.
17         ;
18 012246 012700 0000000  MOV      #FS#EOS, R0      ; SEARCH FOR END-OF-SEGMENT MARKER
19 012252 030061 0000000  BIT      R0, FD#STA(R1)  ; ARE WE POINTING TO IT NOW?
20 012256 001002  BNE      2$              ; BR IF YES
21 012260 004737 013222'  CALL    NXTDIR           ; FIND END OF SEGMENT ENTRY
22         ;
23         ;   Shove down all entries beyond insert point.
24         ;
25 012264 005721 2$:      TST      (R1)+      ; POINT BEYOND END-OF-SEGMENT WORD
26 012266 010100  MOV      R1, R0
27 012270 063700 002006'  ADD     DIRSIZ, R0       ; MAKE ROOM FOR NEW ENTRY
28 012274 014140 1$:      MOV      -(R1), -(R0) ; SHOVE DOWN ALL ENTRIES
29 012276 020116  CMP      R1, (SP)       ; TILL WE REACH INSERT POINT
30 012300 101375  BHI     1$
31         ;
32         ;   Initialize the new entry.
33         ;
34 012302 013700 002006'  MOV     DIRSIZ, R0       ; SIZE OF ENTRY (BYTES)
35 012306 006200  ASR     R0              ; GET # WORDS
36 012310 005021 3$:      CLR      (R1)+      ; ZERO THE ENTRY
37 012312 077002  SDB     R0, 3$
38 012314 012601  MOV     (SP)+, R1       ; GET ADDRESS OF NEW ENTRY
39 012316 012761 0000000 0000000  MOV     #FS#EMP, FD#STA(R1); SAY ENTRY IS EMPTY
40         ;
41         ;   Finished
42         ;
43 012324 000207  RETURN

```

SPLIT -- Split a directory segment

```

1          .SBTTL  SPLIT  -- Split a directory segment
2          ;-----
3          ; SPLIT is called to split a directory segment when it overflows.
4          ; SPLIT moves approximately half of the entries in the current
5          ; segment to a new segment and links the new segment into the directory
6          ; chain.  The number of open directory segments (stored in segment # 1)
7          ; is updated also.
8          ;
9          ; Inputs:
10         ; R1 = Address of a directory entry of interest in current segment.
11         ;
12         ; Outputs:
13         ; R1 = Address of directory entry of interest after split.
14         ; USRBUF = Directory segment containing directory entry pointed to by R1.
15         ; C-flag set on return if there are no available free directory segments.
16         ;
17 012326 010244 SPLIT:  MOV     R2,-(SP)
18 012330 010344      MOV     R3,-(SP)
19 012332 010444      MOV     R4,-(SP)
20 012334 010544      MOV     R5,-(SP)
21 012336 010105      MOV     R1,R5      ;SAVE ADDRESS OF ENTRY WE ARE INTERESTED IN
22         ;
23         ; See if there is a free segment for us to split into.
24         ;
25 012340 023737 002010' 002004'      CMP     DIRNSG,DIRHIS  ;IS THERE A FREE SEGMENT?
26 012346 101524      BLOS    10$      ;BR IF NOT
27 012350 013704 002002'      MOV     CURSEG,R4      ;REMEMBER CURRENT DIR SEGMENT #
28         ;
29         ; There is a free segment.
30         ; Determine where to split this segment.
31         ;
32         ; Count # of entries in this segment.
33         ;
34 012354 012701 000000C      MOV     #USRBUF+DH*$SZ,R1;POINT TO FIRST ENTRY IN SEGMENT
35 012360 005000      CLR     R0      ;COUNT # ENTRIES IN R0
36 012362 032761 0000000 0000000 1$:  BIT     #FS$E0S,FD$STA(R1);IS THIS THE END OF SEGMENT MARKER?
37 012370 001004      BNE     2$      ;BR IF YES
38 012372 005200      INC     R0      ;COUNT ANOTHER ENTRY
39 012374 063701 002006'      ADD     DIRSIZ,R1      ;POINT TO NEXT ENTRY
40 012400 000770      BR     1$
41         ; Leave approximately half the entries in the current segment.
42 012402 006200 2$:  ASR     R0      ;GET 1/2 # ENTRIES
43 012404 013702 000000C      MOV     USRBUF+DH$BLK,R2;GET BASE BLOCK # OF 1ST FILE IN SEGMENT
44 012410 012701 000000C      MOV     #USRBUF+DH*$SZ,R1;POINT TO 1ST ENTRY
45 012414 066102 0000000 3$:  ADD     FD$LEN(R1),R2  ;KEEP TRACK OF BLOCK #'S OF FILES
46 012420 063701 002006'      ADD     DIRSIZ,R1      ;POINT TO NEXT DIRECTORY ENTRY
47 012424 077005      SOB     R0,3$      ;SKIP OVER 1/2 OF THE ENTRIES
48         ; Split on 1st permanent or tentative entry beyond mid-point.
49 012426 032761 000000C 0000000 4$:  BIT     #<FS$PRM+FS$TEN>,FD$STA(R1);IS THIS A PERM OR TENTATIVE ENTRY?
50 012434 001005      BNE     7$      ;BR IF YES
51 012436 066102 0000000      ADD     FD$LEN(R1),R2  ;KEEP TRACK OF STARTING BLOCK NUMBERS
52 012442 063701 002006'      ADD     DIRSIZ,R1      ;POINT TO NEXT ENTRY
53 012446 000767      BR     4$
54         ;
55         ; Found split point.
56         ; R1 = Address of 1st entry to go in new segment.
57         ; R2 = Base block number of 1st file in new segment.

```

```

58 ;
59 ; Calculate address of dir entry we are interested in after split.
60 012450 020501 7#: CMP R5,R1 ; WILL ENTRY GO IN OLD OR NEW SEGMENT?
61 012452 103404 BLD 5# ; BR IF GOING IN OLD SEGMENT
62 012454 160105 SUB R1,R5 ; CALCULATE ITS ADDRESS IN NEW SEGMENT
63 012456 062705 000000C ADD #USRBUF+DH*SZ,R5
64 012462 005004 CLR R4 ; REMEMBER ENTRY GOES IN NEW SEGMENT
65 ;
66 ; Truncate current segment and set link to new segment.
67 ;
68 012464 011146 5#: MOV (R1),-(SP) ; SAVE FD*STAT FOR ENTRY FOLLOWING SPLIT
69 012466 012711 000000C MOV #FS#EOS,(R1) ; SET END-OF-SEGMENT MARKER
70 012472 013746 000000C MOV USRBUF+DH#NXT,-(SP); SAVE LINK TO NEXT SEGMENT
71 012476 013703 002004' MOV DIRHIS,R3 ; GET # OF HIGHEST SEGMENT CURRENTLY IN USE
72 012502 005203 INC R3 ; GET # OF 1ST FREE SEGMENT
73 012504 010337 000000C MOV R3,USRBUF+DH#NXT; SET AS NEW LINK
74 012510 004737 013706' CALL WRTSEG ; WRITE OUT TRUNCATED OLD SEGMENT
75 ;
76 ; Now set up new segment.
77 ;
78 ; Set new header.
79 012514 012637 000000C MOV (SP)+,USRBUF+DH#NXT; SET FLINK
80 012520 010337 002002' MOV R3,CURSEG ; SET OUR SEGMENT #
81 012524 010237 000000C MOV R2,USRBUF+DH#BLK; STARTING BLOCK # FOR FILES IN SEGMENT
82 ; Slide up directory entries from end of segment.
83 012530 012611 MOV (SP)+,(R1) ; FIX FD*STA OF 1ST ENTRY
84 012532 012700 000000C MOV #USRBUF+DH*SZ,R0; POINT TO TOP OF AREA FOR ENTRIES
85 012536 012702 002002' MOV #USRBUF+1024.,R2; POINT PAST END OF BUFFER
86 012542 160102 SUB R1,R2 ; GET # BYTES TO MOVE
87 012544 006202 ASR R2 ; GET # WORDS TO MOVE
88 012546 012120 6#: MOV (R1)+,(R0)+ ; MOVE UP ENTRIES
89 012550 077202 SOB R2,6#
90 ; Write out the new segment.
91 012552 004737 013706' CALL WRTSEG
92 ;
93 ; Update information in header of 1st (master) directory segment.
94 ;
95 012556 004737 013454' CALL RDSEG1 ; READ IN DIR SEG 1
96 012562 010337 000000C MOV R3,USRBUF+DH#HIS; SET # OF HIGHEST SEG IN USE
97 012566 010337 002004' MOV R3,DIRHIS
98 012572 004737 013706' CALL WRTSEG ; WRITE OUT SEG # 1
99 ;
100 ; Now read back in the segment that has the directory entry we are
101 ; interested in.
102 ;
103 012576 005704 TST R4 ; IS ENTRY IN OLD OR NEW DIRECTORY SEGMENT?
104 012600 001401 BEQ 9# ; BR IF IN NEW SEGMENT
105 012602 010403 MOV R4,R3 ; GET NUMBER OF OLD SEGMENT
106 012604 010300 9#: MOV R3,R0 ; SET AS ARGUMENT TO RDSEG
107 012606 004737 013570' CALL RDSEG ; READ IN SEGMENT WITH ENTRY WE WANT
108 ;
109 ; Finished
110 ;
111 012612 010501 MOV R5,R1 ; GET ADDRESS OF ENTRY OF INTEREST
112 012614 000241 CLC ; SIGNAL GOOD RETURN
113 012616 000401 BR 11#
114 ;
    
```

```
115          ; Directory overflow
116          ;
117 012620 000261 10$: SEC          ; SIGNAL ERROR ON RETURN
118 012622 012605 11$: MOV      (SP)+, R5
119 012624 012604      MOV      (SP)+, R4
120 012626 012603      MOV      (SP)+, R3
121 012630 012602      MOV      (SP)+, R2
122 012632 000207      RETURN
```

CONSOL --- Directory segment consolidator

```

1          .SBTTL  CONSOL --- Directory segment consolidator
2          ;-----
3          ; CONSOL is called to remove unnecessary entries from the current directory
4          ; segment.  The unnecessary entries that are removed are:
5          ; 1. Tentative files that are not currently open.
6          ; 2. Multiple consecutive empty entries.
7          ; 3. Empty entries of length 0 that follow a permanent entry.
8          ;
9          ; Inputs:
10         ; USRBUF = Current directory segment.
11         ;
12 012634 010146 CONSOL: MOV     R1,-(SP)
13 012636 010246         MOV     R2,-(SP)
14 012640 010346         MOV     R3,-(SP)
15 012642 010446         MOV     R4,-(SP)
16         ;
17         ; Get exclusive access to job context block buffer
18         ;
19 012644         DCALL  GETCXT      ;Get exclusive access to context block buffer
20         ;
21         ; Pass 1: Convert unopen tentative entries into empty entries.
22         ;
23 012652 012701 000000C         MOV     #USRBUF+DH##SZ,R1;POINT TO 1ST ENTRY
24 012656 013704 002006'         MOV     DIRSIZ,R4      ;CARRY DIRECTORY ENTRY SIZE IN R4
25 012662 160401         SUB     R4,R1      ;POINT IN FRONT OF 1ST ENTRY FOR NXTDIR
26         ; Find the next tentative file entry.
27 012664 012700 000000G 3#:  MOV     #FS$TEN,R0      ;LOOK FOR A TENTATIVE FILE ENTRY
28 012670 004737 013222'         CALL    NXTDIR      ;DO SEARCH
29 012674 103436         BCS     5$      ;BR IF NO MORE TENTATIVE FILE ENTRIES
30         ; We found a tentative file entry.
31         ; R1 points to the entry.
32         ; See if the entry is open now.
33 012676 116102 000000G         MOVVB  FD$JOB(R1),R2  ;GET # OF JOB USING ENTRY
34 012702 001427         BEQ     1$      ;BR IF NOT TSX JOB
35 012704 032702 000001         BIT     #1,R2      ;TSX JOB NUMBER CAN NEVER BE ODD
36 012710 001024         BNE     1$      ;BR IF NOT TSX JOB USING CHANNEL
37 012712 020227 000000G         CMP     R2,#LSTSL  ;MAKE SURE JOB NUMBER IS NOT TOO BIG
38 012716 101021         BHI     1$      ;BR IF NOT TSX JOB
39 012720 032762 000000G 000000G BIT     ##DILUP,LSW(R2) ;IS THAT JOB LOGGED ON?
40 012726 001415         BEQ     1$      ;BR IF NOT
41 012730 116103 000000G         MOVVB  FD$CHN(R1),R3  ;GET CHANNEL # ASSOCIATED WITH FILE
42 012734         DCALL  GETCHA      ;GET POINTER TO CHANNEL BLOCK
43 012742 032760 000000G 000000G BIT     #CS$OPN,C.CSW(R0);IS THAT CHANNEL OPEN?
44 012750 001404         BEQ     1$      ;BR IF NOT
45 012752 032760 000000G 000000G BIT     #CS$ENT,C.CSW(R0);IS CHANNEL OPEN FOR CREATING A NEW FILE?
46 012760 001003         BNE     2$      ;BR IF YES
47         ; Convert this tentative file entry to an empty file entry.
48 012762 012761 000000G 000000G 1#: MOV     #FS$EMP,FD$STA(R1);SAY THIS IS AN EMPTY FILE ENTRY
49         ; Check next entry.
50 012770 000735 2#:  BR     3$

```

```

1
2 ; Pass 2: Consolidate consecutive empty entries and empty entries
3 ; of zero length preceded by a permanent file entry.
4 ;
5 012772 012701 0000000 5$: MOV #USRBUF+DH*$$SZ,R1;POINT TO 1ST ENTRY
6 012776 160401 SUB R4,R1 ;POINT IN FRONT OF 1ST ENTRY
7 013000 012700 0000000 11$: MOV #FS$EMP,R0 ;SEARCH FOR NEXT EMPTY FILE ENTRY
8 013004 004737 013222' CALL NXTDIR
9 013010 103456 BCS 12$ ;BR IF THERE ARE NO MORE
10 ; We have found an empty file entry.
11 ; Combine it with the following entry if it is empty too.
12 013012 010102 7$: MOV R1,R2 ;GET ADDRESS OF CURRENT ENTRY
13 013014 060402 ADD R4,R2 ;POINT TO NEXT ENTRY
14 013016 032762 0000000 0000000 BIT #FS$EMP,FD$STA(R2);IS NEXT ENTRY EMPTY TOO?
15 013024 001417 BEQ 8$ ;BR IF NOT
16 ; Combine the two entries.
17 013026 066261 0000000 0000000 ADD FD$LEN(R2),FD$LEN(R1);COMBINE ALLOCATED SIZES
18 ; Remove the second empty entry.
19 013034 010146 MOV R1,-(SP) ;REMEMBER ADDRESS OF 1ST EMPTY ENTRY
20 013036 012700 0000000 MOV #FS$EOS,R0 ;SEARCH FOR END OF SEGMENT MARKER
21 013042 004737 013222' CALL NXTDIR
22 ; R1 now points to end of segment marker.
23 ; Move up entries over the one being removed (pointed to by R2).
24 013046 010200 MOV R2,R0 ;GET ADDRESS OF ENTRY TO BE REMOVED
25 013050 060400 ADD R4,R0 ;POINT TO FOLLOWING ENTRY
26 013052 012022 6$: MOV (R0)+,(R2)+ ;MOVE UP FOLLOWING ENTRIES OVER ONE BEING REMOVED
27 013054 020001 CMP R0,R1 ;MOVED END-OF-SEGMENT MARKER YET?
28 013056 101775 BLOS 6$ ;BR IF NOT
29 013060 012601 MOV (SP)+,R1 ;GET BACK ADDRESS OF 1ST EMPTY ENTRY
30 013062 000753 BR 7$ ;SEE IF THERE ARE MORE EMPTIES TO BE MERGED
31 ;
32 ; This empty is not followed by any other empty entries.
33 ; See if it is of zero length and follows a permanent file entry.
34 ;
35 013064 005761 0000000 8$: TST FD$LEN(R1) ;IS THIS EMPTY OF ZERO LENGTH?
36 013070 001025 BNE 9$ ;BR IF NOT
37 013072 010102 MOV R1,R2 ;GET ADDRESS OF ENTRY
38 013074 160402 SUB R4,R2 ;GET ADDRESS OF PREVIOUS ENTRY
39 013076 020227 0000000 CMP R2,#USRBUF+DH*$$SZ;ARE WE 1ST ENTRY?
40 013102 103420 BLO 9$ ;BR IF YES
41 013104 032762 0000000 0000000 BIT #FS$PRM,FD$STA(R2);IS PREVIOUS ENTRY A PERMANENT FILE?
42 013112 001414 BEQ 9$ ;BR IF NOT
43 ; Remove a zero length empty entry following a permanent file entry.
44 013114 010246 MOV R2,-(SP) ;SAVE ADDRESS OF PERMANENT FILE ENTRY
45 013116 010102 MOV R1,R2 ;GET ADDRESS OF EMPTY ENTRY
46 013120 012700 0000000 MOV #FS$EOS,R0 ;SEARCH FOR END OF SEGMENT ENTRY
47 013124 004737 013222' CALL NXTDIR
48 013130 010200 MOV R2,R0 ;GET ADDRESS OF EMPTY ENTRY
49 013132 060400 ADD R4,R0 ;POINT TO FOLLOWING ENTRY
50 013134 012022 10$: MOV (R0)+,(R2)+ ;MOVE UP FOLLOWING ENTRIES OVER EMPTY ENTRY
51 013136 020001 CMP R0,R1 ;HAVE WE MOVED END-OF-SEGMENT MARKER YET?
52 013140 101775 BLOS 10$ ;BR IF NOT
53 013142 012601 MOV (SP)+,R1 ;GET BACK ADDRESS OF PERMANENT FILE ENTRY
54 ; Look for next empty entry.
55 013144 000715 9$: BR 11$
56 ;
57 ; Finished

```

```
58  
59 013146      ;  
60 013154 012604 12$: DCALL  FREEXT      ;Release context block buffer  
61 013156 012603      MOV   (SP)+,R4  
62 013160 012602      MOV   (SP)+,R3  
63 013162 012601      MOV   (SP)+,R2  
64 013164 000207      MOV   (SP)+,R1  
                      RETURN
```

GETDIR -- Locate next directory entry

```

1          .SBTTL  GETDIR -- Locate next directory entry
2          ;-----
3          ; GETDIR is called to get the address of the next directory entry of
4          ; a specified type.  Directory segments are read if necessary to find
5          ; the next such entry.
6          ;
7          ; Inputs:
8          ; R1 = Address of last directory entry.
9          ; R0 = FS$xxx entry type flags.
10         ; USRBUF = Current directory segment.
11         ;
12         ; Outputs:
13         ; R1 = Address of next directory entry of the specified type.
14         ; USRBUF = Directory segment that contains the entry.
15         ; C-flag set if no entry of the specified type can be found.
16         ;
17 013166  GETDIR:
18         ;
19         ; See if we can find entry of specified type in the current directory segment.
20         ;
21 013166  004737  013222' 1$:      CALL    NXTDIR      ;SEARCH FOR ENTRY OF SPECIFIED TYPE
22 013172  103012          BCC      9$          ;BR IF WE FOUND ONE
23         ;
24         ; Hit end of current directory segment.  See if there are more segments.
25         ;
26 013174  010046          MOV      RO,-(SP)
27 013176  013700  000000C  MOV      USRBUF+DH#NXT,R0;GET # OF NEXT SEGMENT IN LIST
28 013202  001404          BEQ      2$          ;BR IF REACHED END
29 013204  004737  013570'  CALL      RDSEG      ;READ IN NEXT SEGMENT
30 013210  012600          MOV      (SP)+,R0    ;GET BACK TYPE FLAGS
31 013212  000765          BR       1$          ;SEARCH THIS SEGMENT
32         ;
33         ; No entry of the specified type was found.
34         ;
35 013214  012600  2$:      MOV      (SP)+,R0    ;CLEAN OFF STACK
36 013216  000261          SEC          ;SIGNAL FAILURE ON RETURN
37         ;
38         ; finished
39         ;
40 013220  000207  9$:      RETURN

```

NXTDIR -- Locate next directory entry in current segment

```

1          .SBTTL  NXTDIR -- Locate next directory entry in current segment
2          ;-----
3          ; NXTDIR is called to get the address of the next directory entry of a
4          ; specified type within the current directory segment.
5          ;
6          ; Inputs:
7          ; RO = FS$xxx directory entry type flags.
8          ; R1 = Address of last directory entry.
9          ;
10         ; Outputs:
11         ; R1 = Address of directory entry found.
12         ; C-flag set if no entry of specified type could be found.
13         ;
14 013222 063701 002006'  NXTDIR: ADD     DIRSIZ,R1      ;POINT TO NEXT DIRECTORY ENTRY
15 013226 030061 0000000  ;         BIT     RO,FD$STA(R1)  ;IS THIS ENTRY OF THE RIGHT TYPE?
16 013232 001006  ;         BNE     1$           ;BR IF YES
17 013234 032761 0000000 0000000  ;         BIT     #FS$EOS,FD$STA(R1);IS THIS THE END-OF-SEGMENT MARKER?
18 013242 001767  ;         BEQ     NXTDIR       ;KEEP SEARCHING IF NOT
19          ; Hit end of segment.
20 013244 000261  ;         SEC           ;SIGNAL FAILURE ON RETURN
21 013246 000401  ;         BR      9$
22          ; Found entry.
23 013250 000241 1$:      CLC           ;SIGNAL SUCCESS ON RETURN
24 013252 000207 9$:      RETURN

```

DIDDLE -- Allow user to access directory entry

```

1          .SBTTL  DIDDLE -- Allow user to access directory entry
2          ;-----
3          ; DIDDLE is called to give the user (especially PIP) a chance to access
4          ; and possible alter a directory entry.
5          ; A program requests this feature by storing the address of a routine to
6          ; be called in the fifth word of the argument block for .ENTER, .LOOKUP,
7          ; or .RENAME emt's and setting the name argument word odd.
8          ; The specified routine is called with R1 pointing to the date word of
9          ; the directory entry.
10         ;
11         ; Inputs:
12         ; R1 = Address of the directory entry.
13         ; EMTBLK = Argument list for the emt.
14         ;
15         ; Outputs:
16         ; Directory entry may have been modified by the user.
17         ;
18 013254 032737 000001 0000026 DIDDLE: BIT    #1,EMTBLK+2    ; DOES USER WANT TO DIDDLE WITH THE ENTRY?
19 013262 001455                BEQ      9$              ; BR IF NOT
20 013264 010246                MOV     R2,-(SP)
21 013266 010346                MOV     R3,-(SP)
22 013270 010446                MOV     R4,-(SP)
23 013272 010546                MOV     R5,-(SP)
24 013274 010146                MOV     R1,-(SP)    ; SAVE ADDRESS OF DIR ENTRY ON TOP OF STACK
25         ;
26         ; User does want to diddle with the directory entry.
27         ; Move directory entry to user's address space.
28         ;
29 013276 012703 000300                MOV     #USRUCA,R3    ; GET ADDRESS OF AREA IN USER'S AREA
30 013302 012700 000000C                MOV     #FD#$SZ/2,R0  ; GET # WORDS TO MOVE
31 013306 012146                1$:  MOV     (R1)+,-(SP)    ; PICK UP WORD FROM DIRECTORY ENTRY
32 013310 106623                MTPD   (R3)+          ; MOVE TO AREA IN USER'S SPACE
33 013312 077003                SOB    R0,1$         ; MOVE ALL OF DIRECTORY ENTRY
34         ;
35         ; Use special EMT to get control back from user's routine.
36         ;
37 013314 013703 000000G                MOV     EMTCAD,R3     ; Get pointer to top of return addr stack
38 013320 012743 013364'                MOV     #5$,-(R3)    ; Push our return address
39 013324 010337 000000G                MOV     R3,EMTCAD    ; Save updated stack pointer
40 013330 106506                MFPD   SP            ; GET USER'S STACK POINTER
41 013332 012603                MOV     (SP)+,R3
42 013334 012746 000000G                MOV     #CPLEMT,-(SP) ; PUSH ADDR OF EMT INSTRUCTION ON USER'S STACK
43 013340 106643                MTPD   -(R3)
44 013342 010346                MOV     R3,-(SP)    ; NOW STORE UPDATED USER STACK POINTER
45 013344 106606                MTPD   SP
46         ;
47         ; Now enter user's routine in user mode.
48         ; (Use RTI to do this)
49         ; On entry to user's routine, R1 points to the directory entry.
50         ;
51 013346 012701 000300G                MOV     #USRUCA+FD$DAT,R1 ; MAKE R1 POINT TO DATE WORD OF DIRECTORY ENTRY
52 013352 013746 000000G                MOV     EMTPS,-(SP)  ; PS
53 013356 013746 000010G                MOV     EMTBLK+10,-(SP) ; PC = Address of user's routine
54 013362 000002                RTI                    ; ENTER USER'S ROUTINE
55         ;
56         ; Return here from user's routine (when EMT is done).
57         ;

```

```
58          ; Move directory entry back to directory buffer.
59          ;
60 013364 011602          5$:      MOV      (SP), R2          ; GET ADDRESS WHERE WE SHOULD PUT DIR ENTRY
61 013366 012703 000300      MOV      #USRUCA, R3        ; POINT TO AREA IN USER'S AREA WITH DIR ENTRY
62 013372 012700 000000C      MOV      #FD##SZ/2, R0       ; GET # WORDS TO MOVE
63 013376 106523          2$:      MFPD     (R3)+          ; GET A WORD FROM USER'S DIRECTORY ENTRY
64 013400 012622          MOV      (SP)+, (R2)+        ; MOVE BACK TO OUR INTERNAL AREA
65 013402 077003          SOB      R0, 2$          ; MOVE ALL OF ENTRY
66          ;
67          ; Finished
68          ;
69 013404 012601          MOV      (SP)+, R1
70 013406 012605          MOV      (SP)+, R5
71 013410 012604          MOV      (SP)+, R4
72 013412 012603          MOV      (SP)+, R3
73 013414 012602          MOV      (SP)+, R2
74 013416 000207          9$:      RETURN
```

SBCALC -- Calculate starting block # for a file entry

```

1          .SBTTL  SBCALC -- Calculate starting block # for a file entry
2          ;-----
3          ; SBCALC is called to calculate the starting block number of a file.
4          ;
5          ; Inputs:
6          ; R1 = Address of directory entry for file.
7          ; USRBUF = Directory segment containing entry.
8          ;
9          ; Outputs:
10         ; RO = Starting block number of the file.
11         ;
12 013420 010246 SBCALC: MOV      R2, -(SP)
13 013422 013700      MOV      USRBUF+DH$BLK, RO; GET BLOCK # OF 1ST FILE IN THIS DIR SEGMENT
14 013426 012702 000000C      MOV      #USRBUF+DH$$SZ, R2; POINT TO 1ST DIR ENTRY IN THIS SEGMENT
15 013432 020201 1$:      CMP      R2, R1          ; HAVE WE REACHED OUR ENTRY?
16 013434 001405      BEQ      2$          ; BR IF YES
17 013436 066200 0000000      ADD      FD$LEN(R2), RO      ; ADD SPACE ALLOCATED TO THAT FILE ENTRY
18 013442 063702 002006'      ADD      DIRSIZ, R2        ; POINT TO NEXT ENTRY
19 013446 000771      BR      1$
20 013450 012602 2$:      MOV      (SP)+, R2
21 013452 000207      RETURN

```

RDSEG1 -- Read 1st directory segment

```

1          .SBTTL  RDSEG1 -- Read 1st directory segment
2          ;-----
3          ; RDSEG1 is called to read the first directory segment on a device
4          ; and set up parameters about the directory.
5          ;
6          ; Inputs:
7          ;   CHNNUM = Number of channel we are working on.
8          ;
9          ; Outputs:
10         ;   R1 = Pointer to dummy directory entry in front of 1st real one (for NXTDIR).
11         ;   USRBUF = 1st directory segment.
12         ;   CURSEG = 0
13         ;   DIRHIS = Number of highest directory segment in use.
14         ;   DIRSIZ = Number of bytes per directory entry.
15         ;
16 013454  RDSEG1:
17         ;
18         ; Read in directory segment # 1
19         ;
20 013454  012700  000001          MOV     #1,R0          ;SAY WE WANT SEG # 1
21 013460  004737  013570'       CALL    RDSEG        ;READ THE SEGMENT IN
22         ;
23         ; Set up parameters about the directory
24         ;
25 013464  012701  000002'       MOV     #USRBUF,R1    ;POINT TO BUFFER WITH DIRECTORY SEGMENT
26 013470  016137  000000G 002004'  MOV     DH#HIS(R1),DIRHIS;HIGHEST SEGMENT # USED
27 013476  016137  000000G 002010'  MOV     DH#NSG(R1),DIRNSG;NUMBER OF DIRECTORY SEGMENTS AVAILABLE
28 013504  001425          BEQ     5$           ;BR IF INVALID DIRECTORY
29 013506  023727  002010' 000037  CMP     DIRNSG,#31.   ;CHECK TO SEE IF VALID
30 013514  101021          BHI     5$           ;BR IF INVALID
31 013516  016100  000000G       MOV     DH#NEB(R1),R0 ;# EXTRA BYTES PER DIRECTORY ENTRY
32 013522  020027  001000          CMP     R0,#512.     ;CHECK IF REASONABLE
33 013526  101014          BHI     5$           ;BR IF INVALID
34 013530  062700  000000G       ADD     #FD#OPT,R0   ;# STANDARD BYTES PER DIRECTORY ENTRY
35 013534  010037  002006'       MOV     R0,DIRSIZ    ;TOTAL # BYTES PER DIRECTORY ENTRY
36 013540  026127  000000G 000000G  CMP     DH#BLK(R1),#DH##BS;MAKE SURE BASE BLOCK # IS REASONABLE
37 013546  103404          BLO     5$           ;BR IF INVALID
38         ;
39         ; Point R1 in front of 1st directory entry so NXTDIR will get 1st real entry.
40         ;
41 013550  062701  000000G       ADD     #DH##SZ,R1   ;POINT TO 1ST REAL DIRECTORY ENTRY
42 013554  160001          SUB     R0,R1        ;POINT BACK 1 ENTRY
43         ;
44         ; Finished
45         ;
46 013556  000207          RETURN
47         ;
48         ; Invalid directory
49         ;
50 013560  012700  177755          5$:    MOV     #UERR6,R0   ;FATAL ERROR
51 013564  000137  015144'       JMP     USRCLS

```

RDSEG -- Read a directory segment

```

1          .SBTTL  RDSEG  -- Read a directory segment
2          ;-----
3          ; RDSEG is called to read a directory segment.
4          ;
5          ; Inputs:
6          ;   RO = Number of directory segment to be read.
7          ;   CHNNUM = Number of current channel being used.
8          ;
9          ; Outputs:
10         ;   R1 = Address of dummy directory in front of first real one.
11         ;   USRBUF = Segment read in.
12         ;   CURSEG = Number of segment read in.
13         ;
14 013570 020037 002002' RDSEG:  CMP      RO,CURSEG      ; IS DESIRED SEGMENT ALREADY IN BUFFER?
15 013574 001437         BEQ      1$              ; BR IF YES
16 013576 010001         MOV      RO,R1          ; GET SEG #
17         ; Convert segment # to absolute block #.
18 013600 010137 002002' 2$:   MOV      R1,CURSEG      ; SAVE CURRENT SEGMENT #
19 013604 005301         DEC      R1              ; 1ST DIRECTORY SEGMENT IS # 1
20 013606 006301         ASL      R1              ; CVT SEGMENT # TO BLOCK #
21 013610 062701 000000G ADD      #DH##BS,R1      ; BASE BLOCK # OF 1ST SEGMENT
22         ; Read in the segment.
23 013614         .READW  #USREMT,CHNNUM,#USRBUF,#512,R1
24 013656 103006         BCC      1$              ; BR IF NO ERROR
25 013660 004737 015214' CALL     ERRNAM          ; SET FILE SPEC FOR TSKMON ERROR MESSAGE
26 013664 012700 177775 MOV      #-3,RO          ; DIRECTORY READ ERROR
27 013670 000137 015144' JMP      USRCLS
28         ; Set R1 to point to dummy directory entry in front of first real one.
29 013674 012701 000000C 1$:   MOV      #USRBUF+DH##SZ,R1; POINT TO FIRST REAL ENTRY IN SEGMENT
30 013700 163701 002006' SUB      DIRSIZ,R1       ; POINT IN FRONT OF FIRST ONE
31 013704 000207         RETURN

```

```

1
2
3
4
5
6
7
8
9 013706 010146
10 013710 013701 002002'
11 013714 005301
12 013716 006301
13 013720 062701 000000G
14 013724
15 013766 103006
16 013770 004737 015214'
17 013774 012700 177775
18 014000 000137 015144'
19 014004 012601
20 014006 000207
  
```

```

.SBTTL WRTSEG -- Write directory segment
-----
; WRTSEG writes the current directory segment to disk.
;
; Inputs:
;   USRBUF = Current segment.
;   CURSEG = Number of current segment in USRBUF.
;
WRTSEG: MOV     R1, -(SP)
        MOV     CURSEG, R1      ; GET CURRENT SEGMENT #
        DEC     R1             ; 1ST SEGMENT IS # 1
        ASL     R1             ; CVT SEGMENT # TO BLOCK #
        ADD     #DH*#BS, R1    ; BASE BLOCK # OF 1ST SEGMENT
        .WRITW  #USREMT, CHNNUM, #USRBUF, #512., R1
        BCC     1$            ; BR IF WRITE OK
        CALL    ERRNAM        ; SET FILE SPEC FOR TSKMON ERROR MESSAGE
        MOV     #-3, R0       ; DIRECTORY WRITE ERROR
        JMP     USRCLS        ; ABORT OPERATION
1$:     MOV     (SP)+, R1
        RETURN
  
```

```

1          .SBTTL  SPLDIR -- Directory operations for special devices
2          ;-----
3          ; SPLDIR is jumped to from the normal USR directory routines to perform
4          ; a directory operation on a special type device such as a magnetic
5          ; tape or cassette. SPLDIR releases the USR, performs the operations
6          ; and then returns from the emt.
7          ;
8          ; Inputs:
9          ;   R2 = Operation function code (DF#xxx).
10         ;   R3 = Address of CSW for channel.
11         ;   FILSPC = File spec.
12         ;
13 014010   SPLDIR:
14         ;
15         ; Free the USR module.
16         ;
17 014010   004737  010132'      CALL    FREUSR          ;FREE THE USR DATA BASE FOR OTHERS TO USE
18         ;
19         ; Obtain exclusive access to special device data base.
20         ;
21 014014   004737  014376'      CALL    GETSPD          ;GAIN EXCLUSIVE ACCESS TO SPD DATA BASE
22         ;
23         ; Copy the untranslated device/file specification from the user's area
24         ; to a system buffer.
25         ;
26 014020   013701  000002G      MOV     EMTBLK+2,R1      ;Point to file spec in user's area
27 014024   042701  000001      BIC     #1,R1          ;Make sure address is even
28 014030   010246              MOV     R2,-(SP)        ;Save operation function code
29 014032   012702  000000G      MOV     #SPFLDV,R2     ;Point to buffer where name is to be stored
30 014036   004737  007704'      CALL    CPYSPC         ;Copy file spec to SPFLDV
31 014042   012602              MOV     (SP)+,R2       ;Restore operation function code
32 014044   005037  000000G      CLR     SPSIZE        ;HANDLER RETURNS FILE SIZE HERE
33 014050   005037  000000G      CLR     SPUSR         ;HANDLER RETURNS ERROR CODE HERE
34         ;
35         ; Build an I/O queue entry to do the operation.
36         ;
37 014054   004737  000000G      CALL    GETQ          ;GET A FREE I/O QUEUE ENTRY
38 014060   010361  000000G      MOV     R3,Q.UCSW(R1) ;SET ADDRESS OF CHANNEL CSW
39 014064   013761  000000G 000000G  MOV     @#KPAR6,Q.PA6(R1);Save mapping for context block
40 014072   010105              MOV     R1,R5          ;Get address of Q element
41 014074   062705  000000G      ADD     #Q.ICSW,R5    ;Point to internal CSW cell
42 014100   010561  000000G      MOV     R5,Q.CSW(R1) ;Set up pointer to internal CSW cell
43 014104   010300              MOV     R3,R0         ;Get pointer to original channel block
44 014106   012025              MOV     (R0)+,(R5)+   ;Copy original channel block to internal one
45 014110   012025              MOV     (R0)+,(R5)+
46 014112   012025              MOV     (R0)+,(R5)+
47 014114   012025              MOV     (R0)+,(R5)+
48 014116   011015              MOV     (R0),(R5)
49 014120   016300  000000G      MOV     C.CSW(R3),R0  ;Get the channel status word
50 014124   042700  177701      BIC     #^C76,R0      ;Isolate the device index number
51 014130   110061  000000G      MOVVB  R0,Q.DEVX(R1)  ;store the device index number
52 014134   013700  000006G      MOV     EMTBLK+6,R0   ;GET FILE SEQUENCE # FOR .ENTER
53 014140   020227  000000G      CMP     R2,#DF#ENT    ;ARE WE DOING A .ENTER?
54 014144   001402              BEQ     6$            ;BR IF YES
55 014146   013700  000004G      MOV     EMTBLK+4,R0   ;GET SEQUENCE # FOR .LOOKUP, .DELETE, ETC.
56 014152   010061  000000G      6$:  MOV     R0,Q.BLKN(R1) ;SET SEQUENCE # IN Q.BLKN FIELD
57 014156   013761  000004G 000000G  MOV     EMTBLK+4,Q.WCNT(R1);SET FILE ENTER SIZE IN Q.WCNT FIELD

```

```

58 014164 110261 0000000 MOV B R2,Q.FUNC(R1) ;SET OPERATION FUN CODE (LOOKUP, ENTER, ETC.)
59 014170 116361 0000000 0000000 MOV B C.DEVQ(R3),Q.UNIT(R1);SET UNIT NUMBER
60 014176 113700 0000000 MOV B CDRUSR,R0 ;Get current job index number
61 014202 006200 ASR R0 ;Convert to job number
62 014204 110061 0000000 MOV B R0,Q.JOB(R1) ;Set job number in queue element
63 014210 072027 0000003 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 BIS B 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$: MOV SPSIZE,UR0 ;RETURN HANDLER REPORTED FILE SIZE IN USER'S R0
92 014326 013705 0000000 MOV SPUSR,R5 ;GET HANDLER REPORTED ERROR CODE
93 ;
94 ; Release special device data base area.
95 ;
96 014332 004737 014440' CALL FRESPD ;FREE SPD AREA
97 ;
98 ; See if we should purge the channel.
99 ;
100 014336 020227 0000000 CMP R2,#DF$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

```

GETSPD -- Gain exclusive access to special device data base

```

1          .SBTTL  GETSPD -- Gain exclusive access to special device data base
2          ;-----
3          ; GETSPD is called to gain exclusive access to the special device data base.
4          ; If the data base is currently in use by another job, our job is suspended
5          ; until the data base becomes free.
6          ;
7 014376 113700 0000000 GETSPD: MOVB   SPDJOB,RO      ; IS SPD DATA BASE FREE NOW?
8 014402 001412          BEQ     1$          ; BR IF YES
9 014404 120037 0000000      CMPB   RO,CORUSR      ; HAVE WE ALREADY GOT IT?
10 014410 001412          BEQ     2$          ; BR IF YES
11          ;
12          ; Someone else owns SPD data base.
13          ; Suspend our job until it becomes free.
14          ;
15 014412 012700 0000000      MOV     #$QSPD,RO      ; QUEUE FOR SPD DATA BASE
16 014416 004737 0000000      CALL   QNSPNX          ; PUT US AT TAIL OF QUEUE FOR IT
17 014422 004737 0000000      CALL   CHKABT          ; SEE IF WE WERE ABORTED WHILE ASLEEP
18 014426 000763          BR      GETSPD          ; NOW TRY TO GET SPD
19          ;
20          ; We can get SPD data base now.
21          ;
22 014430 113737 0000000 0000000 1$: MOVB   CORUSR,SPDJOB ; CLAIM SPD DATA BASE FOR US
23 014436 000207          2$: RETURN
24          ;
25          .SBTTL  FRESPD -- Free special device data base
26          ;-----
27          ; Free the special device data base and restart any user who is waiting
28          ; for it.
29          ;
30 014440 123737 0000000 0000000 FRESPD: CMPB   CORUSR,SPDJOB ; DO WE CURRENTLY OWN SPD?
31 014446 001006          BNE     1$          ; BR IF NOT
32 014450 105037 0000000      CLRB   SPDJOB          ; FREE IT
33          ;
34          ; Restart any job that is waiting for SPD data base.
35          ;
36 014454 012700 0000000      MOV     #$QSPD,RO      ; GET SPD WAIT QUEUE STATE CODE
37 014460 004737 0000000      CALL   UREGO          ; RESTART ANY WAITING JOB
38          ;
39          ; Finished
40          ;
41 014464 000207          1$: RETURN

```

CSHADD -- Add file entry to directory cache

```

1          .SBTTL  CSHADD -- Add file entry to directory cache
2          ;-----
3          ; CSHADD is called to add a new file entry to the directory cache.
4          ;
5          ; Inputs:
6          ; R1 = Pointer to directory entry for file in USRBUF.
7          ; FILDVU = Device # / unit #
8          ;
9 014466 010246 CSHADD: MOV     R2,-(SP)
10 014470 010346      MOV     R3,-(SP)
11 014472 010446      MOV     R4,-(SP)
12 014474 010546      MOV     R5,-(SP)
13          ;
14          ; See if we are caching file entries for this device
15          ;
16 014476 004737 014776'      CALL    CSHTST      ;ARE WE TO CACHE ENTRIES FOR THIS DEVICE?
17 014502 103440      BCS     9$          ;BR IF NOT
18          ;
19          ; We are caching entries for this device.
20          ; Find least recently used entry in cache table (or 1st unused entry).
21          ;
22 014504 012702 0000000      MOV     #CSHHD-FC$LNK,R2;DUMMY POINTER TO PHANTOM FIRST ENTRY
23 014510 010203 1$:      MOV     R2,R3          ;REMEMBER ADDRESS OF PREVIOUS ENTRY
24 014512 016202 0000000      MOV     FC$LNK(R2),R2  ;CHAIN FORWARD TO NEXT ENTRY IN LIST
25 014516 005762 0000000      TST     FC$CDX(R2)    ;IS THIS ENTRY UNUSED?
26 014522 001403          BEQ     2$          ;BR IF YES
27 014524 005762 0000000      TST     FC$LNK(R2)    ;IS THIS THE LAST ENTRY IN LIST?
28 014530 001367          BNE     1$          ;BR IF NOT
29          ;
30          ; Relink cache entry at head of chain (it is most recently used).
31          ;
32 014532 016263 0000000 0000000 2$:      MOV     FC$LNK(R2),FC$LNK(R3);RELINK CHAIN AROUND US
33 014540 013762 0000000 0000000      MOV     CSHHD,FC$LNK(R2);NOW LINK US AT FRONT OF LIST
34 014546 010237 0000000      MOV     R2,CSHHD
35          ;
36          ; Copy info from file directory entry to cache entry.
37          ;
38 014552 010203          MOV     R2,R3          ;GET ADDRESS OF CACHE ENTRY
39 014554 010104          MOV     R1,R4          ;GET ADDRESS OF DIRECTORY ENTRY
40 014556 012700 0000000      MOV     #FD*$SZ,R0    ;GET # BYTES TO MOVE
41 014562 006200          ASR     R0             ;GET # WORDS TO MOVE
42 014564 012423 3$:      MOV     (R4)+,(R3)+    ;COPY DIRECTORY INFO TO CACHE ENTRY
43 014566 077002          SOB     R0,3$
44          ;
45          ; Store pointer to cached-device table entry into file cache entry
46          ;
47 014570 010562 0000000      MOV     R5,FC$CDX(R2) ;REMEMBER WHICH DEVICE FILE IS ASSOC WITH
48          ;
49          ; Store starting block number of file
50          ;
51 014574 004737 013420'      CALL    SBCALC        ;CALCULATE STARTING BLOCK # OF FILE
52 014600 010062 0000000      MOV     R0,FC$SBL(R2) ;SAVE STARTING BLOCK #
53          ;
54          ; Finished
55          ;
56 014604 012605 9$:      MOV     (SP)+,R5
57 014606 012604          MOV     (SP)+,R4

```

58	014610	012603	MOV	(SP)+, R3
59	014612	012602	MOV	(SP)+, R2
60	014614	000207	RETURN	

CSHDEL -- Remove a file entry from the directory cache

```

1          .SBTTL  CSHDEL -- Remove a file entry from the directory cache
2          ;-----
3          ; CSHDEL is called to remove an individual file entry from the cache table.
4          ;
5          ; Inputs:
6          ; R1 = Pointer to directory entry for file to be removed.
7          ; FILDVU = Device # / unit # of device.
8          ;
9 014616 010146 CSHDEL: MOV      R1,-(SP)
10         ;
11         ; Search for file entry in cache table.
12         ;
13 014620 062701 0000000  ADD      #FD$NAM,R1      ;POINT TO FILE NAME IN DIRECTORY ENTRY
14 014624 004737 014642'  CALL    CSHCHK        ;SEE IF ENTRY IS IN DIRECTORY CACHE
15 014630 103402          BCS     4$            ;BR IF FILE ENTRY IS NOT IN CACHE
16         ;
17         ; Found entry in cache table.
18         ; Mark it as deleted.
19         ;
20 014632 005061 0000000  CLR     FC$CDX(R1)    ;MARK ENTRY AS DELETED
21         ;
22         ; Finished
23         ;
24 014636 012601          4$:   MOV     (SP)+,R1
25 014640 000207          RETURN

```

```

1          .SBTTL  CSHCHK -- Search for file entry in directory cache
2          ;-----
3          ; CSHCHK is called to try to locate an entry for a file in
4          ; the directory cache.  If it is found the file is set as the most
5          ; recently used.
6          ;
7          ; Inputs:
8          ; R1 = Pointer to file name (3 words, rad50: name, name, extension)
9          ; FILDVU = Device & unit #.
10         ;
11         ; Outputs:
12         ; C-flag cleared if file entry is found.
13         ; R1 = Pointer to cache table entry for file
14         ; (same format as directory entry).
15         ; R0 = Starting block # of file.
16         ;
17 014642 010246 CSHCHK: MOV     R2,-(SP)
18 014644 010346      MOV     R3,-(SP)
19 014646 010546      MOV     R5,-(SP)
20 014650 010103      MOV     R1,R3          ;CARRY FILE NAME POINTER IN R3
21         ;
22         ; See if this device is being cached and get pointer to device table entry
23         ;
24 014652 004737 014776' CALL    CSHTST          ;SEE IF THIS DEVICE IS BEING CACHED
25 014656 103442      BCS     4$             ;BR IF DEVICE IS NOT BEING CACHED
26         ;
27         ; Search for file entry in cache table.
28         ;
29 014660 012701 000000C      MOV     #CSHHD-FC$LNK,R1;GET POINTER TO DUMMY STARTING POINT
30 014664 010102      1$:    MOV     R1,R2          ;REMEMBER ADDRESS OF PREVIOUS ENTRY
31 014666 016101 000000G      MOV     FC$LNK(R1),R1  ;GET ADDRESS OF NEXT ENTRY IN LIST
32 014672 020561 000000G      CMP     R5,FC$CDX(R1)  ;IS FILE ON CACHED DEVICE?
33 014676 001012      BNE     2$             ;BR IF NOT
34 014700 010300      MOV     R3,R0          ;POINT TO FILE NAME
35 014702 022061 000000G      CMP     (R0)+,FD$NAM(R1);COMPARE FILE NAMES
36 014706 001006      BNE     2$             ;
37 014710 022061 000002G      CMP     (R0)+,FD$NAM+2(R1)
38 014714 001003      BNE     2$             ;
39 014716 021061 000004G      CMP     (R0),FD$NAM+4(R1)
40 014722 001404      BEQ     3$             ;BR IF FOUND ENTRY FOR FILE
41 014724 005761 000000G      2$:    TST     FC$LNK(R1)  ;IS THIS LAST ENTRY IN LIST?
42 014730 001355      BNE     1$             ;BR IF NOT
43 014732 000414      BR      4$             ;FILE ENTRY IS NOT IN CACHE TABLE
44         ;
45         ; Found entry -- Relink at head of list (it is most recently used).
46         ;
47 014734 016162 000000G 000000G 3$:    MOV     FC$LNK(R1),FC$LNK(R2);RELINK LIST AROUND US
48 014742 013761 000000G 000000G      MOV     CSHHD,FC$LNK(R1);LINK US IN AT HEAD OF LIST
49 014750 010137 000000G      MOV     R1,CSHHD
50         ;
51         ; Return with R1 = address of cache entry, R0 = Starting block #.
52         ;
53 014754 016100 000000G      MOV     FC$SBL(R1),R0  ;GET STARTING BLOCK # FOR FILE
54 014760 000241      CLC                    ;SIGNAL SUCCESSFUL RETURN
55 014762 000401      BR      5$             ;
56         ;
57         ; 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
```

CSHTST -- Determine if device is to be cached

```

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 000000G         MOV     CSHDEV,R5         ;POINT TO TABLE OF MOUNTED DEVICES
23 015012 020365 000000G 2$:    CMP     R3,CD$DVU(R5)     ;DO DEVICE # AND UNIT #'S MATCH?
24 015016 001003         BNE     5$                 ;BR IF NOT
25 015020 020465 000000G         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 000000G 5$:    ADD     #CD$$SZ,R5         ;POINT TO NEXT TABLE ENTRY
28 015032 020537 000000G         CMP     R5,CSHDVN        ;REACHED END OF TABLE?
29 015036 103765         BLD     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

```

GETDVU -- Get device # & unit # info

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 015054 013703 002014'
15 015060 005004
16 015062 120337 0000000
17 015066 001010
18 015070 000303
19 015072 042703 177770
20 015076 006303
21 015100 016304 0000000
22 015104 016303 0000000
23
24
25
26 015110 000207

```

```

.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)
;
GETDVU: MOV FILDVU,R3 ;GET DEVICE # AND UNIT #
        CLR R4 ;ASSUME DEVICE IS NOT A LOGICAL DISK
        CMPB R3,LDDEVX ;IS THIS DEVICE A LOGICAL DISK?
        BNE 4$ ;BR IF NOT
        SWAB R3 ;GET LOGICAL UNIT # TO LOW ORDER BYTE
        BIC #^C7,R3 ;MASK OUT ALL BUT UNIT #
        ASL R3 ;CONVERT TO WORD TABLE INDEX
        MOV LDBASE(R3),R4 ;GET STARTING POSITION OF LOGICAL DISK
        MOV LDPDEV(R3),R3 ;GET PHYSICAL DEVICE # AND UNIT #
;
; Finished
;
4$: RETURN

```

CDJFLG -- Get user-flag for cached device entry

```

1          .SBTTL  CDJFLG -- Get user-flag for cached device entry
2          ;-----
3          ; CDJFLG is called to locate within a cached-device table entry the
4          ; specific mount-flag that corresponds to the current job.
5          ;
6          ; Inputs:
7          ;   R3 = Job index number of job whose flag is to be identified.
8          ;   R5 = Pointer to cached device table entry.
9          ;
10         ; Outputs:
11         ;   R2 = Address of byte that contains mount-flag.
12         ;   R3 = Mount-flag bit positioned correctly within byte.
13         ;
14 015112 006203  CDJFLG: ASR      R3          ; Convert job index to index by 1
15 015114 005303          DEC      R3          ; Make base job number 0
16 015116 005002          CLR      R2          ; Clear for divide
17 015120 071227 000010  DIV      #8,R2        ; Divide by 8 jobs per byte
18 015124 060502          ADD      R5,R2        ; Get address of byte within
19 015126 062702 000000G  ADD      #CD$JOB,R2      ; cached-device table entry
20 015132 012700 000001  MOV      #1,R0          ; Get a mount flag
21 015136 072003          ASH      R3,R0          ; Position flag according to job number
22 015140 010003          MOV      R0,R3          ; Return flag in R3
23 015142 000207          RETURN

```

```

1          .SBTTL  USRXIT -- Exit from USR
2          ;-----
3          ; USRXIT is the exit point for EMT processing within the USR.
4          ; USRCLS sets an I/O error code, purges the channel then aborts the emt.
5          ; USRERR sets an I/O error code but does not purge the channel.
6          ;
7          ; Inputs:
8          ;   R0 = error code [USRERR, and USRCLS only]
9          ;
10         015144  013703  0000000  USRCLS:  MOV     CHNADR,R3      ;GET ADDRESS OF CURRENT CHANNEL
11         015150  001402                BEQ     USRERR      ;BR IF NONE
12         015152  005063  0000000      CLR     C.CSW(R3)   ;PURGE THE CHANNEL
13         015156  010046                USRERR:  MOV     R0,-(SP)  ;SAVE ERROR CODE
14         015160  004737  010132'      CALL   FREUSR      ;FREE USR DATA BASE IF WE HAVE IT
15         015164  012600                MOV     (SP)+,R0    ;GET BACK ERROR CODE
16         015166  000137  0000000      JMP     SETERR     ;RETURN ERROR CODE FOR EMT
17         ;
18         ; Normal exit from USR
19         ;
20         015172  004737  010132'  USRXIT:  CALL   FREUSR      ;FREE USR MODULE
21         015176  000137  0000000      JMP     EMTXIT     ;EXIT FROM EMT
22         ;
23         ; Fatal abort due to internal consistency check.
24         ;
25         ; Inputs:
26         ;   R0 = Abort code.
27         ;
28         015202  010005                UABORT:  MOV     R0,R5      ;GET ERROR CODE FOR ABORT
29         015204  013704  0000000      MOV     EMTADR,R4   ;GET ADDRESS OF ABORTED EMT
30         015210  000137  0000000      JMP     ABORT      ;GIVE A FATAL ABORT

```

ERRNAM -- Set name of file spec for error message

```

1          .SBTTL  ERRNAM -- Set name of file spec for error message
2          ;-----
3          ; ERRNAM is called to move the current device/file specification to a
4          ; cell that will cause the spec to be printed with an error message
5          ; when the job reenters TSKMON.
6          ;
7          ; Inputs:
8          ;   FILSPC = Device/file specification in RAD50 form.
9          ;
10         ; Outputs:
11         ;   ERRSPC = Device/file specification saved for TSKMON.
12         ;
13 015214  ERRNAM:
14         ;
15         ; Don't set file spec for error message if a .SERR has been done
16         ;
17 015214  105737  000000G      TSTB   SERFLG      ; Has a .SERR been done?
18 015220  001014              BNE     9$         ; Br if yes
19         ;
20         ; Move file spec to holding area
21         ;
22 015222  013737  000000G  000000G      MOV    FILSPC,ERRSPC      ; Device name
23 015230  013737  000002G  000002G      MOV    FILSPC+2,ERRSPC+2  ; File name part 1
24 015236  013737  000004G  000004G      MOV    FILSPC+4,ERRSPC+4  ; File name part 2
25 015244  013737  000006G  000006G      MOV    FILSPC+6,ERRSPC+6  ; Extension
26         ;
27         ; Finished
28         ;
29 015252  000207      9$:   RETURN
30         ;
31         ; Top of TSUSR module
32         ;
33 015254  USRTOP:
34         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

\$DILUP	1-55	40-39										
\$DUPRN	1-71	34-130										
\$KINIT	1-41	19-77										
\$NDIN	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							

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						

... 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	