

Table of contents

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	ACCMCH -- Compare file spec with [NO]ACCESS entry
36-	1	FNDFIL -- Find file in directory
37-	1	FNDFRE -- Find a free slot for a new file
38-	1	ADDENT -- Add a tentative file entry to directory
39-	1	INSERT -- Add an empty file entry to directory
40-	1	SPLIT -- Split a directory segment
41-	1	CONSOL -- Directory segment consolidator
43-	1	GETDIR -- Locate next directory entry
44-	1	NXTDIR -- Locate next directory entry in current segment
45-	1	DIDDLE -- Allow user to access directory entry
46-	1	SBCALC -- Calculate starting block # for a file entry
47-	1	RDSEG1 -- Read 1st directory segment
48-	1	RDSEG -- Read a directory segment
49-	1	WRTSEG -- Write directory segment
50-	1	SPLDIR -- Directory operations for special devices
51-	1	GETSPD -- Gain exclusive access to special device data base
51-	25	FRESPD -- Free special device data base
52-	1	CSHADD -- Add file entry to directory cache
53-	1	CSHDEL -- Remove a file entry from the directory cache
54-	1	CSHCHK -- Search for file entry in directory cache
55-	1	CSHTST -- Determine if device is to be cached
56-	1	GETDVU -- Get device # & unit # info
57-	1	CDJFLG -- Get user-flag for cached device entry
58-	1	USRXIT -- Exit from USR
59-	1	ERRNAM -- Set name of file spec for error message

```

1          .TITLE  TSUSR  -- File operation EMT's
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  GBL
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, KMONCE
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$$SZ
35         .GLOBL  PO$ALC, PRIVCO, PO$BYP, PO$NFR, PO$NFW
36         .GLOBL  CHKABT, UREGO, NUMDEV, LSTFDT, LSTFDD
37         .GLOBL  TK3SVL, FD$TIM, SYTIMH, SYTIML, FC$CDX
38         .GLOBL  QNSPNX, JCDB, DVFLAG, DX$RAL, DX$NMT
39         .GLOBL  PNAME, CHNNUM, SYINDX, SYUNIT, RUNFLG, AF$BYA
40         .GLOBL  C. CSW, C. SBLK, C. LENG, C. USED, C. NUMQ, C. DEVQ
41         .GLOBL  CS$NMX, CS$ENT, SERFLG, $KINIT, NLCHN
42         .GLOBL  CS$OPN, CHKSD, CS$SPL, CSHCLN, CSHALC
43         .GLOBL  LSW2, RESDEV, OKFILE, OF$DEV, LSW6, OKFAND, OKFNND
44         .GLOBL  OF$UNT, OT$RON, OF$FLG, OF$FIL, OF$$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, $UCLRN, LSW7
48         .GLOBL  FD$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  IQWAIT, GETCXT, FREEXT

```

58	. GLOBL	UACHKW, VALADB, VALADW
59	. GLOBL	HANSIZ, HANENT, DEVSIZ, EMTPS
60	. GLOBL	SYSCHN, LSTSL, 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, QIO
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, CLZERR

```

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  000000  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 002050  000     CNSLXD: .BYTE  0          ;CONSOL routine reorganized segment flag
55         ;
56         ; Internal USR error checks.
57         ; (Numbers correspond to KMON error table offsets)

```

58		,			
59	177762	UERR1	=	-16	; Can't find tentative file entry for file on close
60	177761	UERR2	=	-17	; File length in chan block not = to len in file entry
61	177760	UERR3	=	-20	; Highest block # written > file length
62	177757	UERR4	=	-21	; Empty file entry doesn't follow tentative file entry
63	177756	UERR5	=	-22	; Not tentative file entry during close
64	177755	UERR6	=	-23	; Illegal or uninitialized directory
65				. 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 002052  010246  USRINI: MOV      R2, -(SP)
7 002054  010346          MOV      R3, -(SP)
8          ;
9          ;  Allocate tables within this overlay segment above top of code
10         ;
11 002056  012702  015466'      MOV      #USRTOP, R2      ;Point above top of code in this segment
12         ;
13         ;  Allocate file directory cache table
14         ;
15 002062  010237  000000G      MOV      R2, CSHHD      ;Set pointer to 1st free directory cache entry
16 002066  013700  000000G      MOV      VNFCSH, R0     ;Get # cache entries to allocate
17 002072  010203      1$:  MOV      R2, R3      ;Get address of current entry
18 002074  062703  000000G      ADD      #FC$$SZ, R3   ;Get address of next entry
19 002100  020327  140000      CMP      R3, #140000   ;Will current entry fit in overlay region?
20 002104  101403      BLOS    2$           ;Br if yes
21 002106  162703  000000G      SUB      #FC$$SZ, R3   ;No -- Backup pointer to current entry
22 002112  000406      BR      3$           ;
23 002114  010362  000000G      2$:  MOV      R3, FC$LNK(R2) ;Make current entry point to next one
24 002120  005062  000000G      CLR      FC$CDX(R2)   ;Say entry is currently free
25 002124  010302      MOV      R3, R2      ;Make next entry be current one
26 002126  077017      SOB     R0, 1$       ;Loop if more to allocate
27 002130  162703  000000G      3$:  SUB      #FC$$SZ, R3   ;Point to last entry
28 002134  005063  000000G      CLR      FC$LNK(R3)   ;Make its forward link zero
29         ;
30         ;  Finished
31         ;
32 002140  012603      MOV      (SP)+, R3
33 002142  012602      MOV      (SP)+, R2
34 002144  000207      RETURN

```

.LOOKUP

```

1          .SBTTL .LOOKUP
2          ;-----
3          ; Perform a file lookup.
4          ;
5 002146 004737 007376' LOOKUP: CALL USRCOM ;DO COMMON PARAMETER SETUP
6 002152 005037 000000G CLR URO ;FOR NOW, SAY FILE SIZE = 0 BLOCKS
7          ;
8          ; Now R3 = Address of channel block, R4 = Index into device tables.
9          ;
10         ; See if the device is spooled.
11         ;
12 002156 105737 000000G TSTB NSPLDV ;Are there any spooled devices?
13 002162 001404 BEQ 1$ ;Br if not
14 002164 DCALL CHKSD ;See if we are opening to spooled device
15 002172 103063 BCC 3$ ;Br if this is a spooled device
16         ;
17         ; This is not a spooled device.
18         ; See if this is a magnetic tape.
19         ;
20 002174 032764 000000G 000000G 1$: BIT #DS$NRD, DVSTAT(R4) ;IS THIS A NON-RT-11 DIRECTORY DEVICE?
21 002202 001004 BNE 6$ ;Br if yes
22 002204 032764 000000G 000000G BIT #DX$NRD, DVFLAG(R4); Internal non-RT-11 dir flag set?
23 002212 001404 BEQ 5$ ;BR IF NOT
24 002214 012702 000000G 6$: MOV #DF$LOK, R2 ;CODE FOR LOOKUP
25 002220 000137 014212' JMP SPLDIR ;DO SPECIAL DEVICE LOOKUP
26         ;
27         ; See if this is a file structured device.
28         ;
29 002224 032764 000000G 000000G 5$: BIT #DS$DIR, DVSTAT(R4); DIRECTORY STRUCTURED DEVICE?
30 002232 001443 BEQ 3$ ;Br if not -- We are finished
31         ;
32         ; This is a file structured device.
33         ; See if this is a non-file-structured lookup (null file name).
34         ;
35 002234 005737 000002G TST FILSPC+2 ;IS FILE NAME NULL?
36 002240 001440 BEQ 3$ ;BR IF NON-FILE-STRUCTURED LOOKUP
37         ;
38         ; Lookup file name on a directory structured device.
39         ; See if file info is stored in directory cache.
40         ;
41 002242 012701 000002G MOV #FILSPC+2, R1 ;POINT TO NAME OF FILE
42 002246 004737 015054' CALL CSHCHK ;LOOK FOR FILE IN DIRECTORY CACHE
43 002252 103013 BCC 2$ ;BR IF FOUND IT
44         ;
45         ; File info was not found in directory cache
46         ; so we must look at real directory.
47         ;
48 002254 004737 011376' CALL FNDFIL ;LOOKUP FILE NAME IN DEVICE DIRECTORY
49 002260 103004 BCC 4$ ;BR IF FOUND
50 002262 012700 000001 MOV #1, R0 ;RETURN ERROR CODE OF 1 IF FILE DOESN'T EXIST
51 002266 000137 015356' JMP USRCLS
52         ;
53         ; Add file entry to directory cache.
54         ;
55 002272 010046 4$: MOV R0, -(SP) ;SAVE FILE STARTING BLOCK #
56 002274 004737 014700' CALL CSHADD ;ADD FILE INFO TO DIRECTORY CACHE
57 002300 012600 MOV (SP)+, R0 ;GET BACK FILE STARTING BLOCK #

```

.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 002302 010063 000000G 2$:      MOV      R0,C.SBLK(R3) ;SET FILE STARTING BLOCK #
64 002306 016100 000000G      MOV      FD$LEN(R1),R0 ;GET ALLOCATED SIZE OF FILE
65 002312 010063 000000G      MOV      R0,C.LENG(R3) ;STORE IN CHANNEL BLOCK
66 002316 010037 000000G      MOV      R0,URO ;RETURN TO USER IN R0
67 002322 016137 000000G 000000G      MOV      FD$TIM(R1),LSTFDT;SAVE TIME ENTRY FROM EACH LOOKUP
68 002330 016137 000000G 000000G      MOV      FD$DAT(R1),LSTFDD;ALSO SAVE DATE ENTRY
69 ;
70 ; Allow user to diddle with directory entry if he wants to.
71 ;
72 002336 004737 013456'      CALL      DIDDLE ;ALLOW DIRECTORY ENTRY DIDDLE
73 ;
74 ; Finished
75 ;
76 002342 000137 015404' 3$:      JMP      USRXIT ;EXIT FROM USR PROCESSING

```


.ENTER

```

1          .SBTTL .ENTER
2          ;-----
3          ; Open a new file.
4          ;
5 002346 004737 007376' ENTER: CALL USRCOM ;DO COMMON SETUP
6 002352 005037 000000G CLR URO ;FOR NOW, SAY FILE SIZE = 0
7          ;
8          ; See if we have write access to the file.
9          ;
10 002356 032763 000000G 000000G BIT #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO THE FILE?
11 002364 001406 BEQ 4$ ;BR IF YES
12 002366 004737 015426' CALL ERRNAM ;SET NAME OF FILE FOR TSKMON ERROR MESSAGE
13 002372 012700 177763 MOV #-15,R0 ;SET ABORT CODE
14 002376 000137 015356' JMP USRCLS ;ABORT OPERATION
15          ;
16          ; See if it is a spooled device.
17          ;
18 002402 105737 000000G 4$: TSTB NSPLDV ;Are there any spooled devices?
19 002406 001404 BEQ 3$ ;Br if not
20 002410 OCALL CHKSD ;Are we opening to a spooled device?
21 002416 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 002420 032764 000000G 000000G 3$: BIT #DS$NRD,DVSTAT(R4);MAG TAPE TYPE DIRECTORY DEVICE?
27 002426 001004 BNE 7$ ;Br if yes
28 002430 032764 000000G 000000G BIT #DX$NRD,DVFLAG(R4);Internal flag set?
29 002436 001407 BEQ 5$ ;BR IF NOT
30 002440 052763 000000G 000000G 7$: BIS #CS$ENT,C.CSW(R3);SET FLAG TO CAUSE TO CALL HANDLER ON CLOSE
31 002446 012702 000000G MOV #DF$ENT,R2 ;GET CODE FOR ENTER
32 002452 000137 014212' JMP SPLDIR ;GO DO SPECIAL DEVICE ENTER
33          ;
34          ; See if this is a file structured device.
35          ;
36 002456 032764 000000G 000000G 5$: BIT #DS$DIR,DVSTAT(R4);FILE STRUCTURED DEVICE?
37 002464 001451 BEQ 9$ ;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 002466 013702 002044' MOV ASNSIZ,R2 ;WAS SIZE SPECIFIED WITH ASSIGN STATEMENT?
43 002472 001002 BNE 6$ ;BR IF YES
44 002474 013702 000004G MOV EMTBLK+4,R2 ;GET REQUESTED SIZE FROM EMT ARG BLOCK
45 002500 004737 011460' 6$: CALL FNDFRE ;TRY TO FIND A FREE SLOT
46 002504 103002 BCC 1$ ;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 002506 000137 015356' JMP USRCLS
50          ;
51          ; We have found a free slot.
52          ; Create a tentative file entry.
53          ;
54 002512 010237 000000G 1$: MOV R2,URO ;RETURN FILE SIZE IN USER'S R0
55 002516 004737 012210' CALL ADDENT ;CREATE A TENTATIVE FILE ENTRY
56 002522 103004 BCC 2$ ;BR IF WE WERE SUCCESSFUL
57          ; Error: Directory overflow -- Return error code 1.

```

.ENTER

```

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

```

.RENAME

```

1          .SBTTL .RENAME
2          ;-----
3          ; Rename a file.
4          ;
5 002614 004737 007376'  RENAME: CALL  USRCOM          ; DO COMMON SETUP
6          ;
7          ; See if this is a file structured device.
8          ;
9 002620 032764 000000G 000000G      BIT    #DS$DIR,DVSTAT(R4); FILE STRUCTURED DEVICE?
10 002626 001004          BNE    1$          ; BR IF YES
11 002630 012700 000002          MOV    #2,R0          ; INVALID OPERATION
12 002634 000137 015356'          JMP    USRCLS
13          ;
14          ; Make sure we have write access to old file.
15          ;
16 002640 032763 000000G 000000G 1$: BIT    #CS$RON,C.CSW(R3); DO WE HAVE WRITE ACCESS TO OLD FILE?
17 002646 001063          BNE    11$          ; BR IF NOT
18          ;
19          ; Locate directory entry with old name.
20          ; (Note: FILSPC currently contains old file name)
21          ;
22 002650 004737 011376'          CALL   FDNFIL          ; LOOK UP THE ENTRY
23 002654 103004          BCC    2$          ; BR IF FOUND
24 002656 012700 000001          MOV    #1,R0          ; FILE NOT FOUND
25 002662 000137 015356'          JMP    USRCLS
26          ; Save info about old file entry.
27 002666 010146 2$: MOV    R1,-(SP)          ; ADDRESS OF ITS DIRECTORY ENTRY
28 002670 010105          MOV    R1,R5          ; CARRY ADDR OF DIR ENTRY IN R5 FOR A WHILE
29 002672 013746 002002'          MOV    CURSEG,-(SP)          ; SEGMENT NUMBER CONTAINING ENTRY
30 002676 010446          MOV    R4,-(SP)          ; DEVICE TABLE INDEX
31          ;
32          ; Remove old file entry from directory cache.
33          ;
34 002700 004737 015030'          CALL   CSHDEL          ; DELETE FILE ENTRY FROM CACHE TABLE
35          ;
36          ; Set up new file spec.
37          ;
38 002704 013701 000002G          MOV    EMTBLK+2,R1          ; ADDRESS OF FILE SPEC AREA
39 002710 042701 000001          BIC    #1,R1          ; MAKE SURE ADDRESS IS EVEN
40 002714 062701 000010          ADD    #10,R1          ; POINT TO NEW FILE SPEC
41 002720 004737 007546'          CALL   GETSPC          ; MOVE DEVICE-FILE SPEC TO FILSPC
42 002724 004737 010226'          CALL   CHKDEV          ; MAKE SURE DEVICE IS LEGAL
43 002730 103006          BCC    3$          ; BR IF OK
44 002732 004737 015426'          CALL   ERRNAM          ; SET FILE SPEC FOR TSKMON ERROR MESSAGE
45 002736 012700 177776          MOV    #-2,R0          ; ILLEGAL DEVICE
46 002742 000137 015356'          JMP    USRCLS
47          ;
48          ; Make sure user is authorized to access to new file.
49          ;
50 002746 004737 010602' 3$: CALL   CHKACC          ; SEE IF USER IS AUTHORIZED FOR ACCESS
51 002752 032763 000000G 000000G      BIT    #CS$RON,C.CSW(R3); IS USER AUTHORIZED TO WRITE TO NEW FILE?
52 002760 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 002762 020426          CMP      R4,(SP)+      ;COMPARE DEVICE INDEX NUMBERS
59 002764 001405          BEQ      4$             ;BR IF OK
60 002766 022626          5$:    CMP      (SP)+,(SP)+ ;CLEAN OFF STACK
61 002770 012700 000002    MOV      #2,R0        ;INVALID OPERATION
62 002774 000137 015356'  JMP      USRCLS
63 003000 120063 000000G  4$:    CMPB     RO,C.DEVQ(R3) ;DO UNIT NUMBERS MATCH?
64 003004 001370          BNE      5$             ;BR IF DON'T MATCH
65
66 ; Make sure we have write access to new file.
67 ;
68 003006 032763 000000G 000000G BIT      #CS$RON,C.CSW(R3);DO WE HAVE WRITE ACCESS TO NEW FILE?
69 003014 001406          BEQ      10$            ;BR IF YES
70 003016 004737 015426'  11$:   CALL     ERRNAM      ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
71 003022 012700 177763    MOV      #-15,R0      ;SET ABORT CODE
72 003026 000137 015356'  JMP      USRCLS        ;ABORT THE OPERATION
73
74 ; Delete any existing file with new file name.
75 ;
76 ; Temporarily mark old file entry as empty so we won't find it when
77 ; searching for file that has same name as new file.
78 ; Note that this is safe since the USR is locked and noone else can
79 ; access the directory.
80 003032 042765 000000G 000000G 10$:   BIC      #FS$PRM,FD$STA(R5);CLEAR PERMANENT-FILE STATUS FLAG
81 003040 052765 000000G 000000G BIS      #FS$EMP,FD$STA(R5);SET EMPTY-FILE STATUS FLAG
82 003046 023727 002002' 000001    CMP      CURSEG,#1    ;IS THIS DIR ENTRY IN SEGMENT # 1?
83 003054 001402          BEQ      12$            ;BR IF YES -- IT WILL NOT BE REREAD BY FNDFIL
84 003056 004737 014110'  CALL     WRTSEG        ;WRITE OUT MODIFIED ENTRY SO FNDFIL WILL READ IT
85 ; Try to locate directory entry for file being deleted.
86 ; (Note: FILSPC now contains new file name)
87 003062 004737 011376'  12$:   CALL     FNDFIL      ;LOOKUP NEW FILE NAME
88 003066 103442          BCS      9$             ;BR IF DON'T NEED TO DELETE
89
90 ; File with new name already exists.
91 ; See if it is protected.
92 ;
93 003070 032761 000000G 000000G BIT      #FS$PRO,FD$STA(R1);IS FILE PROTECTED?
94 003076 001420          BEQ      13$            ;BR IF NOT
95 ; File is protected so we cannot delete it.
96 ; Restore file status word for old file then return error code 3 for rename.
97 003100 012600          MOV      (SP)+,R0      ;GET DIR SEG # FOR OLD FILE ENTRY
98 003102 004737 013772'  CALL     RDSEG         ;READ IN THE SEGMENT
99 003106 012601          MOV      (SP)+,R1      ;GET ADDRESS OF DIR ENTRY IN SEGMENT
100 003110 042761 000000G 000000G BIC      #FS$EMP,FD$STA(R1);CLEAR EMPTY-FILE FLAG
101 003116 052761 000000G 000000G BIS      #FS$PRM,FD$STA(R1);SET PERMANENT-FILE FLAG
102 003124 004737 014110'  CALL     WRTSEG        ;REWRITE THE DIR SEGMENT
103 003130 012700 000003    MOV      #3,R0        ;RETURN ERROR CODE 3
104 003134 000137 015356'  JMP      USRCLS
105 ; File is not protected.
106 ; Remove its entry from directory cache.
107 003140 004737 015030'  13$:   CALL     CSHDEL      ;DELETE FILE ENTRY FROM CACHE TABLE
108 ; Mark its entry as empty.
109 003144 012761 000000G 000000G MOV      #FS$EMP,FD$STA(R1);FILE IS NOW DELETED
110 ; See if this entry is in same directory segment as the file being renamed.
111 ; If it is, bypass the consolidation & rewrite.
112 003152 023716 002002'  CMP      CURSEG,(SP)   ;IS DELETED FILE ENTRY IN SAME SEG AS RENAMED ENTRY?
113 003156 001002          BNE      6$             ;BR IF NOT
114 003160 012600          MOV      (SP)+,R0      ;POP SEGMENT NUMBER

```

.RENAME

```

115 003162 000407          BR      7$          ;GO FILL IN NEW NAME
116                      ; Consolidate & rewrite segment with deleted file entry.
117 003164 004737 013016' 6$: CALL  CONSOL      ;CONSOLIDATE DIRECTORY SEGMENT
118 003170 004737 014110' CALL  WRTSEG      ;WRITE IT OUT
119                      ;
120                      ; Now change the name in the old file entry.
121                      ;
122 003174 012600          9$: MOV    (SP)+,R0      ;GET DIRECTORY SEGMENT #
123 003176 004737 013772' CALL  RDSEG      ;READ IN THE SEGMENT
124 003202 012601          7$: MOV    (SP)+,R1      ;GET ADDRESS OF ENTRY WITHIN SEGMENT
125 003204 042761 000000G 000000G BIC    #FS$EMP,FD$STA(R1);CLEAR EMPTY-FILE FLAG
126 003212 052761 000000G 000000G BIS    #FS$PRM,FD$STA(R1);SET PERMANENT-FILE FLAG
127 003220 010105          MOV    R1,R5        ;GET ADDRESS OF DIR ENTRY BEING RENAMED
128 003222 062705 000000G ADD    #FD$NAM,R5      ;POINT TO FILE NAME AREA IN ENTRY
129 003226 012702 000002G MOV    #FILSPC+2,R2    ;POINT TO NEW FILE NAME
130 003232 012225          MOV    (R2)+,(R5)+    ;MOVE IN NEW NAME
131 003234 012225          MOV    (R2)+,(R5)+
132 003236 011215          MOV    (R2),(R5)
133                      ;
134                      ; Give user a chance to diddle with the dir entry if he wishes.
135                      ;
136 003240 004737 013456' CALL  DIDDLE      ;ALLOW USER TO DIDDLE
137                      ;
138                      ; Add new file entry to cache table.
139                      ;
140 003244 004737 014700' CALL  CSHADD      ;ADD NEW FILE ENTRY TO CACHE TABLE
141                      ;
142                      ; Consolidate and rewrite directory segment.
143                      ;
144 003250 004737 013016' CALL  CONSOL      ;CONSOLIDATE DIRECTORY SEGMENT
145 003254 004737 014110' CALL  WRTSEG      ;WRITE IT OUT
146                      ;
147                      ; Finished
148                      ;
149 003260 005063 000000G CLR    C.CSW(R3)    ;PURGE THE CHANNEL
150 003264 000137 015404' JMP    USRXIT

```

.DELETE

```

1          .SBTTL .DELETE
2          ;-----
3          ; Delete a permanent file entry.
4          ;
5 003270 004737 007376' DELETE: CALL  USRCOM          ; DO COMMON SETUP
6          ;
7          ; See if we have write access to file being deleted.
8          ;
9 003274 032763 000000G 000000G          BIT    #CS$RON,C.CSW(R3); DO WE HAVE WRITE ACCESS TO FILE?
10 003302 001406          BEQ     2$          ; BR IF YES
11 003304 004737 015426'          CALL   ERRNAM          ; SET FILE SPEC FOR TSKMON ERROR MESSAGE
12 003310 012700 177763          MOV    #-15,R0         ; SET ABORT ERROR CODE
13 003314 000137 015356'          JMP    USRCLS         ; ABORT THE OPERATION
14          ;
15          ; See if device is a magnetic tape.
16          ;
17 003320 032764 000000G 000000G 2$:    BIT    #DS$NRD,DVSTAT(R4); IS THIS A MAG TAPE DEVICE?
18 003326 001004          BNE    6$          ; Br if yes
19 003330 032764 000000G 000000G          BIT    #DX$NRD,DVFLAG(R4); Internal flag set?
20 003336 001404          BEQ    5$          ; BR IF NOT
21 003340 012702 000000G          6$:    MOV    #DF$DEL,R2         ; SET DELETE FUNCTION CODE
22 003344 000137 014212'          JMP    SPLDIR        ; DO SPECIAL DEVICE DELETE
23          ;
24          ; See if this is a file structured device.
25          ;
26 003350 032764 000000G 000000G 5$:    BIT    #DS$DIR,DVSTAT(R4); IS THIS A DIRECTORY STRUCTURED DEVICE?
27 003356 001004          BNE    1$          ; BR IF YES
28          ;
29          ; Delete is invalid on non-file-structured device.
30          ;
31 003360 012700 000002          MOV    #2,R0          ; RETURN INVALID-OPERATION ERROR CODE
32 003364 000137 015356'          JMP    USRCLS
33          ;
34          ; Try to locate directory entry for file being deleted.
35          ;
36 003370 004737 011376'          1$:    CALL   FNDFIL          ; LOOK UP THE FILE
37 003374 103425          BCS    9$          ; BR IF FILE DOES NOT EXIST
38          ;
39          ; See if file is protected.
40          ;
41 003376 032761 000000G 000000G          BIT    #FS$PRO,FD$STA(R1); IS FILE PROTECTED?
42 003404 001404          BEQ    3$          ; BR IF NOT
43 003406 012700 000003          MOV    #3,R0          ; RETURN ERROR CODE 3 IF FILE IS PROTECTED
44 003412 000137 015356'          JMP    USRCLS
45          ;
46          ; Remove file entry from directory cache.
47          ;
48 003416 004737 015030'          3$:    CALL   CSHDEL          ; REMOVE FILE ENTRY FROM DIRECTORY CACHE
49          ;
50          ; File exists. Mark its entry as empty.
51          ;
52 003422 012761 000000G 000000G          MOV    #FS$EMP,FD$STA(R1); MARK DIR ENTRY AS EMPTY
53          ;
54          ; Consolidate and rewrite the directory segment.
55          ;
56 003430 004737 013016'          CALL   CONSOL         ; CONSOLIDATE THE SEGMENT
57 003434 004737 014110'          CALL   WRTSEG        ; WRITE SEGMENT TO DISK

```

.DELETE

```
58 ;  
59 ; Finished  
60 ;  
61 003440 005063 000000G ; CLR C.CSW(R3) ;PURGE THE CHANNEL  
62 003444 000137 015404' ; JMP USRXIT  
63 ;  
64 ; File does not exist.  
65 ;  
66 003450 012700 000001 9#: MOV #1,RO ;FILE DOES NOT EXIST  
67 003454 000137 015356' ; JMP USRCLS
```

.CLOSE

```

1          .SBTTL .CLOSE
2          ;-----
3          ; Close a channel.
4          ;
5 003460 013703 000000G      CLOSE:  MOV      CHNADR,R3      ;GET ADDRESS OF CHANNEL BLOCK
6 003464 032763 000000G 000000G  BIT      #CS$OPN,C.CSW(R3); IS THE CHANNEL OPEN?
7 003472 001002              BNE      7$          ;BR IF CHANNEL IS OPEN
8 003474 000137 004204'      JMP      8$          ;NOP IF NOT OPEN
9          ;
10         ; Wait for all I/O to finish on this channel
11         ;
12 003500 004737 000000G      7$:    CALL      IOWAIT      ;WAIT FOR ALL I/O ON CHANNEL TO FINISH
13         ;
14         ; See if we are closing a spooled device.
15         ;
16 003504 032763 000000G 000000G  BIT      #CS$SPL,C.CSW(R3); IS THIS A SPOOLED DEVICE CHANNEL?
17 003512 001405              BEQ      1$          ;BR IF NOT
18 003514              OCALL     SDCLOS      ;CLOSE A SPOOLED DEVICE CHANNEL
19 003522 000137 004200'      JMP      9$
20         ;
21         ; See if device being closed is a magnetic tape.
22         ;
23 003526 016300 000000G      1$:    MOV      C.CSW(R3),R0      ;GET CSW
24 003532 042700 000000C      BIC      #^C<CS$NMX>,R0 ;MASK OUT ALL BUT DEVICE TABLE INDEX
25 003536 032760 000000G 000000G  BIT      #DS$NRD,DVSTAT(R0); IS DEVICE A MAG TAPE?
26 003544 001004              BNE      18$         ;Br if yes
27 003546 032760 000000G 000000G  BIT      #DX$NRD,DVFLAG(R0); Internal flag set?
28 003554 001404              BEQ      6$          ;BR IF NOT
29 003556 012702 000000G      18$:   MOV      #DF$CLS,R2      ;DO SPECIAL DEVICE CLOSE
30 003562 000137 014212'      JMP      SPLDIR
31         ;
32         ; See if we are closing a shared file.
33         ;
34 003566 005737 000000G      6$:    TST      JCDB          ;Does job have any shared file chans open?
35 003572 001402              BEQ      17$         ;BR IF NOT
36 003574 004777 000000G      CALL     @SFCLS        ;CHECK FOR CLOSING A SHARED FILE
37         ;
38         ; See if this file was opened with a .LOOKUP or .ENTER
39         ;
40 003600 032763 000000G 000000G  17$:   BIT      #CS$ENT,C.CSW(R3); DID WE DO A .ENTER ON THE CHANNEL?
41 003606 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 003610 004737 010004'      CALL     GETUSR        ;CLAIM EXCLUSIVE ACCESS TO USR
47         ;
48         ; Set up device # and unit # in FILDVU.
49         ;
50 003614 116337 000000G 002015'  MOVB     C.DEVQ(R3),FILDVU+1; SET UNIT #
51 003622 016300 000000G      MOV      C.CSW(R3),R0      ;GET CHANNEL STATUS WORD
52 003626 042700 000000C      BIC      #^C<CS$NMX>,R0 ;MASK OUT ALL BUT DEVICE INDEX #
53 003632 110037 002014'      MOVB     R0,FILDVU        ;SET DEVICE INDEX #
54         ;
55         ; Set up info about this directory
56         ;
57 003636 005063 000000G      CLR      C.SBLK(R3)       ;SAY CHANNEL BEING USED FOR DIRECTORY OP

```


.CLOSE

```

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

```

.CLOSE

```

115 004062 010061 000000G      MOV      R0,FD$LEN(R1)      ;SET THIS AS THE ACTUAL FILE LENGTH
116 004066 020002              CMP      R0,R2              ;MAKE SURE HIGHEST BLOCK NOT ABOVE FILE LENGTH
117 004070 101062              BHI     13$                 ;ERROR IF TOO BIG
118 004072 160002              SUB      R0,R2              ;GET # BLOCKS LEFT OVER
119 004074 063701 002006'      ADD      DIRSIZ,R1          ;POINT TO EMPTY FILE ENTRY THAT FOLLOWS
120 004100 032761 000000G 000000G  BIT      #FS$EMP,FD$STA(R1);MAKE SURE THIS REALLY IS AN EMPTY ENTRY
121 004106 001456              BEQ     14$                 ;BAD ERROR IF NOT
122 004110 060261 000000G      ADD      R2,FD$LEN(R1)      ;ADD RESIDUAL BLOCKS TO EMPTY ENTRY
123                               ; Set date and time of file creation in file directory entry.
124 004114 163701 002006'      SUB      DIRSIZ,R1          ;POINT BACK TO NEW DIRECTORY ENTRY
125 004120 005061 000000G      CLR      FD$TIM(R1)         ;SAY CREATION TIME UNKNOWN
126 004124 005761 000000G      TST      FD$DAT(R1)         ;DID DIDDLE ROUTINE SET FILE DATE?
127 004130 001013              BNE     16$                 ;BR IF YES
128 004132 013761 000000G 000000G  MOV      SYSDAT,FD$DAT(R1);SET DATA THAT FILE WAS CREATED (CLOSED)
129 004140 013704 000000G      MOV      SYTIMH,R4          ;GET CURRENT TIME OF DAY (HIGH ORDER)
130 004144 013705 000000G      MOV      SYTIML,R5          ;GET LOW-ORDER TIME OF DAY
131 004150 071437 000000G      DIV      TK3SVL,R4          ;CONVERT TO 3-SECOND UNITS
132 004154 010461 000000G      MOV      R4,FD$TIM(R1)      ;SET TIME OF FILE CREATION
133                               ; Add entry for new file to directory cache.
134 004160 004737 014700'      16$:    CALL     CSHADD        ;ADD TO CACHE TABLE
135                               ; Consolidate and rewrite the directory segment.
136 004164 004737 013016'      CALL     CONSOL             ;CONSOLIDATE THE DIRECTORY
137 004170 004737 014110'      CALL     WRTSEG             ;WRITE OUT THE SEGMENT
138                               ;
139                               ; Free the USR
140                               ;
141 004174 004737 010146'      CALL     FREUSR             ;RELEASE USR DATA BASE
142                               ;
143                               ; Mark the channel as closed.
144                               ;
145 004200 005063 000000G      9$:    CLR      C.CSW(R3)    ;SAY THE CHANNEL IS CLOSED
146                               ;
147                               ; Finished, check for non-fatal errors
148                               ;
149 004204 113700 000000G      8$:    MOV      CLZERR,R0     ;Do we need to report .CLOSZ error?
150 004210 001402              BEQ     10$                 ;Br if not
151 004212 000137 000000G      JMP      SETERR             ;If so, report error to user
152                               ;
153                               ; Successful completion
154                               ;
155 004216 000137 000000G      10$:   JMP      EMTXIT
156                               ;
157                               ; Consistency check errors -- May indicate hardware errors or bugs.
158                               ;
159                               ; Can't find tentative file during close.
160 004222 012700 177762      11$:   MOV      #UERR1,R0
161 004226 000413              BR      20$
162                               ; File size in channel block doesn't agree with size in directory entry.
163 004230 012700 177761      12$:   MOV      #UERR2,R0
164 004234 000410              BR      20$
165                               ; Highest block # written is greater than file length.
166 004236 012700 177760      13$:   MOV      #UERR3,R0
167 004242 000405              BR      20$
168                               ; Empty file entry does not follow tentative file entry.
169 004244 012700 177757      14$:   MOV      #UERR4,R0
170 004250 000402              BR      20$
171                               ; Not pointing to tentative file entry during close.

```

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

.DSTATUS

```

1
2
3
4
5 004262 004737 010004'
6
7 004266 013701 000000G
8 004272 004737 007546'
9
10 004276 004737 010226'
11 004302 103003
12
13 004304 005000
14 004306 000137 015370'
15
16
17
18
19
20 004312 010005
21 004314 013700 000004G
22 004320 010037 000000G
23 004324 004737 000000G
24 004330 016446 000000G
25 004334 106620
26 004336 016446 000000G
27 004342 106620
28 004344 016446 000000G
29 004350 106620
30 004352 016446 000000G
31 004356 106620
32
33
34
35 004360 020437 000000G
36 004364 001004
37 004366 006305
38 004370 016546 000000G
39 004374 106640
40
41
42
43 004376 000137 015404'

```

```

.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

```

.FETCH

```

1          .SBTTL .FETCH
2          ;-----
3          ; .FETCH EMT, This is basically a NOP under TSX-Plus since all device
4          ; handlers are permanently resident. However, we do check to make sure
5          ; the device is installed.
6          ;
7 004402 004737 010004'  FETCH:  CALL    GETUSR          ;Get exclusive access to USR data base
8          ;
9          ; Move device name to FILSPC buffer and perform any assigns
10         ;
11 004406 013701 000000G  MOV     URO,R1          ;Point to device spec
12 004412 004737 007546'  CALL   GETSPC          ;Move to filspc buffer
13         ;
14         ; Look up device in perm name table
15         ;
16 004416 004737 010226'  CALL   CHKDEV          ;Look up the device name
17 004422 103003          BCC    1$              ;Br if found
18         ;
19         ; Invalid device
20         ;
21 004424 005000          CLR    RO              ;Set error code = 0
22 004426 000137 015370'  JMP    USRERR          ;Return with error
23         ;
24         ; The device name is valid.
25         ; Return handler load address to user in RO.
26         ;
27 004432 013700 000004G  1$:   MOV    EMTBLK+4,RO ;Get user's load address
28 004436 005200          INC    RO              ;Round up
29 004440 042700 000001  BIC    #1,RO          ;And force to even address
30 004444 010037 000000G  MOV    RO,URO          ;And return as top of handler
31 004450 000137 015404'  JMP    USRXIT         ;Exit from EMT

```

.SFDAT, .SFTIM, & .FPROT

```

1          .SBTTL .SFDAT, .SFTIM, & .FPROT
2          ;-----
3          ; .SFDAT -- Set date in file entry
4          ;
5 004454 004737 004720' SFDATE: CALL SFCOM ;LOCATE FILE ENTRY
6 004460 004737 004770' CALL SFWRIT ;MAKE SURE WE HAVE WRITE ACCESS
7 004464 013700 000004G MOV EMTBLK+4,R0 ;GET SPECIFIED DATE VALUE
8 004470 001002 BNE 1$ ;BR IF DATE SPECIFIED
9 004472 013700 000000G MOV SYSDAT,R0 ;USE CURRENT DATE
10 004476 016105 000000G 1$: MOV FD$DAT(R1),R5 ;SAVE OLD FILE DATE
11 004502 010061 000000G MOV RO,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 RO,LSTFDD ;ARE WE USING THE OLD FILE'S DATE?
22          ; BNE SFEXIT ;BR IF NOT
23          ; MOV LSTFDT,FD$TIM(R1); USE TIME VALUE FROM LAST .LOOKUP
24          ;
25          ; Add new file entry to directory cache
26          ;
27 004506 004737 014700' SFEXIT: CALL CSHADD ;ADD ENTRY TO CACHE
28          ;
29          ; Rewrite directory segment
30          ;
31 004512 004737 014110' CALL WRTSEG ;REWRITE DIRECTORY SEGMENT
32          ;
33          ; Finished
34          ;
35 004516 005063 000000G CLR C.CSW(R3) ;SAY CHANNEL IS CLOSED
36 004522 000137 015404' JMP USRXIT ;EXIT
37          ;-----
38          ; .SFTIM -- Set file time in file entry
39          ;
40          ;
41 004526 004737 004720' SFTIME: CALL SFCOM ;LOCATE FILE ENTRY
42 004532 004737 004770' CALL SFWRIT ;MAKE SURE WE HAVE WRITE ACCESS
43 004536 013761 000004G 000000G MOV EMTBLK+4,FD$TIM(R1) ;SET TIME VALUE IN DIRECTORY ENTRY
44 004544 000760 BR SFEXIT ;REWRITE DIRECTORY SEGMENT
45          ;-----
46          ; .FPROT -- Set file protection
47          ;
48          ;
49 004546 004737 004720' SFPROT: CALL SFCOM ;DO COMMON SETUP
50 004552 004737 004770' CALL SFWRIT ;MAKE SURE WE HAVE WRITE ACCESS
51 004556 113700 000004G MOVB EMTBLK+4,R0 ;GET PROTECT/UNPROTECT FLAG
52 004562 001407 BEQ 2$ ;BR IF UNPROTECT WANTED
53 004564 020027 000001 CMP RO,#1 ;VALUE MUST BE 0 OR 1
54 004570 001410 BEQ 3$ ;BR IF PROTECT WANTED
55 004572 012700 000003 MOV #3,R0 ;ERROR IF NOT 0 OR 1
56 004576 000137 015356' JMP USRCLS
57 004602 042761 000000G 000000G 2$: BIC #FS$PRO,FD$STA(R1) ;UNPROTECT THE FILE

```

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

.GFINF -- Get information about a file

```

1                                     .SBTTL .GFINF -- Get information about a file
2                                     ;-----
3                                     ; .GFINF Get the following information about a file:
4                                     ; 1. Size of the file.
5                                     ; 2. Protected/Unprotected status.
6                                     ; 3. Date of creation.
7                                     ; 4. Time of creation.
8                                     ; 5. Starting block number of file.
9                                     ;
10 004622 004737 004720' GFINFO: CALL SFCOM ;LOCATE FILE ENTRY
11 004626 013700 000004G MOV EMTBLK+4,R0 ;GET ADDRESS OF USER'S INFO BUFFER
12 004632 004737 000000G CALL VALADW ;MAKE SURE ADDRESS IS VALID
13                                     ; File size
14 004636 016146 000000G MOV FD$LEN(R1),-(SP);FILE SIZE
15 004642 106620 MTPD (R0)+
16                                     ; Protected/Unprotected status
17 004644 005046 CLR -(SP) ;ASSUME FILE IS NOT PROTECTED
18 004646 032761 000000G 000000G BIT #FS$PRO,FD$STA(R1); IS FILE PROTECTED?
19 004654 001401 BEQ 1$ ;BR IF NOT
20 004656 005216 INC (SP) ;RETURN 1 IF FILE IS PROTECTED
21 004660 106620 1$: MTPD (R0)+
22                                     ; Creation date
23 004662 016146 000000G MOV FD$DAT(R1),-(SP);CREATION DATE
24 004666 106620 MTPD (R0)+
25                                     ; Creation time
26 004670 016146 000000G MOV FD$TIM(R1),-(SP);CREATION TIME
27 004674 106620 MTPD (R0)+
28                                     ; Starting block number
29 004676 010002 MOV R0,R2 ;SAVE BUFFER POINTER
30 004700 004737 013622' CALL SBCALC ;CALCULATE STARTING BLOCK NUMBER
31 004704 010046 MOV R0,-(SP)
32 004706 106622 MTPD (R2)+ ;RETURN STARTING BLOCK NUMBER
33                                     ;
34                                     ; Finished
35                                     ;
36 004710 005063 000000G CLR C.CSW(R3) ;SAY CHANNEL IS CLOSED
37 004714 000137 015404' JMP USRXIT ;EXIT

```


.QFINF -- 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 004720 004737 007376' SFCOM: 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 004724 032764 000000G 000000G BIT     #DS$DIR.DVSTAT(R4) ; IS THIS A FILE STRUCTURED DEVICE?
19 004732 001004          BNE     1$              ; BR IF YES
20 004734 012700 000002     MOV     #2,R0          ; RETURN ERROR CODE 2 IF NOT
21 004740 000137 015356'     JMP     USRCLS
22     ;
23     ; Locate directory entry for the file
24     ;
25 004744 004737 011376' 1$:    CALL   FNDFIL           ; LOCATE THE DIRECTORY ENTRY
26 004750 103004          BCC     3$              ; BR IF ENTRY FOR FILE FOUND
27 004752 012700 000001     MOV     #1,R0          ; ERROR 1 IF CAN'T FIND ENTRY
28 004756 000137 015356'     JMP     USRCLS
29     ;
30     ; Remove directory entry from cache
31     ;
32 004762 004737 015030' 3$:    CALL   CSHDEL           ; REMOVE ENTRY FROM CACHE
33     ;
34     ; Finished
35     ;
36 004766 000207          RETURN
37
38     ; -----
39     ; Make sure that we have write access to a file.
40     ; Produce an abort if not.
41     ;
42     ; Inputs:
43     ;   R3 = Address of CSW for channel opened to the file.
44     ;
45 004770 032763 000000G 000000G SFWRIT: BIT     #CS$RON.C.CSW(R3) ; DO WE HAVE WRITE ACCESS TO THE FILE?
46 004776 001406          BEQ     1$              ; BR IF YES
47 005000 004737 015426'     CALL   ERRNAM          ; SET FILE SPEC FOR TSKMON ERROR MESSAGE
48 005004 012700 177763     MOV     #-15,R0       ; ERROR IF NOT
49 005010 000137 015356'     JMP     USRCLS
50 005014 000207          1$:    RETURN           ; SUCCESSFUL RETURN

```

ALCEMT -- Allocate a device

```

1          .SBTTL  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 005016   ALCEMT:
7          ;
8          ; Determine if this is an Allocate, Deallocate, or test allocation EMT
9          ;
10 005016  113700  000000G          MOVB    EMTBLK,R0          ;Get sub-function code
11 005022  001410          BEQ     10$          ;0==>Allocate
12 005024  120027  000001          CMPB    R0,#1          ;1==>Deallocate
13 005030  001475          BEQ     DLCEMT
14 005032  120027  000002          CMPB    R0,#2          ;2==>Test allocation
15 005036  001567          BEQ     TLCEMT
16 005040  000137  000000G          JMP     BADEMT          ;Invalid sub-function code
17          ;
18          ; Allocate a device
19          ;
20 005044  032737  000000G 000000G 10$:  BIT     #PO$ALC,PRIVCO ;Are we authorized to allocate devices?
21 005052  001004          BNE     11$          ;Br if yes
22 005054  012700  000005          MOV     #5,R0          ;Error code 5 if not authorized
23 005060  000137  000000G          JMP     SETERR
24          ;
25          ; Do common allocate setup
26          ;
27 005064  004737  005636' 11$:  CALL    ALCCOM          ;Do common setup
28          ;
29          ; See if the device being allocated is currently allocated to another job
30          ;
31 005070  004737  005436'          CALL    ALCTST          ;Test for allocation to another job
32          ;
33          ; Device is not currently allocated.
34          ; Check to see if the device is already allocated by our job or a
35          ; related job.
36          ;
37 005074  012702  000000G          MOV     #ALCTBL,R2          ;Point to start of allocation table
38 005100  013700  002014'          MOV     FILDVU,R0          ;Get device # and index #
39 005104  020062  000000G 4$:  CMP     R0,AD$DVU(R2)      ;Search for device in allocation table
40 005110  001406          BEQ     5$          ;Br if found it
41 005112  062702  000000G          ADD     #AD$$SZ,R2          ;Point to next entry
42 005116  020227  000000G          CMP     R2,#ALCEND          ;Checked all entries?
43 005122  103770          BLD     4$          ;Br if not
44 005124  000410          BR     7$          ;Br if not currently in allocation table
45          ;
46          ; Device is already in allocation table
47          ;
48 005126  116200  000000G 5$:  MOVB    AD$JOB(R2),R0      ;Get # of job that owns device
49 005132  120160  000000G          CMPB    R1,LNPRIM(R0)      ;Is primary job reallocating device?
50 005136  001030          BNE     9$          ;Br if not
51 005140  110162  000000G          MOVB    R1,AD$JOB(R2)      ;Let primary job take over device
52 005144  000425          BR     9$
53          ;
54          ; Device is not currently allocated.
55          ; Search for a free entry in the allocation table.
56          ;
57 005146  012702  000000G 7$:  MOV     #ALCTBL,R2          ;Point to start of allocation table

```

ALCEMT -- Allocate a device

```

58 005152 105762 000000G      2$:   TSTB   AD$JOB(R2)      ;Is this entry free?
59 005156 001411              BEQ     3$           ;Br if yes
60 005160 062702 000000G      ADD     #AD$$SZ,R2   ;Point to next table entry
61 005164 020227 000000G      CMP     R2,#ALCEND   ;Checked all entries?
62 005170 103770              BLO     2$           ;Loop if more to check
63 005172 012700 0000003      MOV     #3,R0        ;Error 3 -- Allocation table is full
64 005176 000137 015370'      JMP     USRERR       ;Error abort
65                               ;
66                               ; Found a free entry, set it up for this device
67                               ;
68 005202 013762 002014' 000000G 3$:   MOV     FILDVU,AD$DVU(R2);Store device and unit numbers
69 005210 110162 000000G      MOVB   R1,AD$JOB(R2) ;Store job number
70 005214 105062 000000G      CLRB   AD$FLG(R2)   ;No flags yet
71                               ;
72                               ; Finished
73                               ;
74 005220 000137 015404'      9$:   JMP     USRXIT

```

```

1                                     .SBTTL  DLCEMT -- Deallocate a device
2                                     ;-----
3                                     ; Deallocate a device.
4                                     ;
5 005224 032737 000000G 000000G DLCEMT: BIT      #PO$ALC,PRIVCO ;Are we authorized to allocate devices?
6 005232 001004                                     BNE      11$          ;Br if yes
7 005234 012700 000005                                     MOV      #5,R0        ;Error code 5 if not authorized
8 005240 000137 000000G                                     JMP      SETERR
9 005244 004737 005636' 11$: CALL    ALCCOM      ;Do common setup
10 005250 012702 000000G                                     MOV      #ALCTBL,R2   ;Point to start of allocation table
11                                     ;
12                                     ; If the device name is null, deallocate all devices allocated by
13                                     ; this user.
14                                     ;
15 005254 013700 000000G                                     MOV      FILSPC,R0    ;Is device name null?
16 005260 001015                                     BNE      2$          ;Br if not
17 005262 120162 000000G 1$: CMPB   R1,AD$JOB(R2) ;Is this entry an allocation for our job?
18 005266 001004                                     BNE      3$          ;Br if not
19 005270 105062 000000G                                     CLRB    AD$JOB(R2)    ;Say no job using this entry
20 005274 005062 000000G                                     CLR     AD$DVU(R2)    ;Say entry is free
21 005300 062702 000000G 3$: ADD    #AD$$SZ,R2   ;Point to next allocation table entry
22 005304 020227 000000G                                     CMP     R2,#ALCEND    ;Checked all entries?
23 005310 103764                                     BLO     1$          ;Loop if more to check
24 005312 000437                                     BR      9$
25                                     ;
26                                     ; The device name is not null.
27                                     ; Deallocate the specified device.
28                                     ;
29 005314 105737 002014' 2$: TSTB   FILDVU      ;Trying to deallocate TT?
30 005320 001434                                     BEQ     9$          ;Ignore if yes
31 005322 013700 002014'                                     MOV     FILDVU,R0     ;Get device/unit ID for device being dealloc
32 005326 020062 000000G 7$: CMP    R0,AD$DVU(R2) ;Search for specified device in alloc table
33 005332 001406                                     BEQ     5$          ;Br if found it
34 005334 062702 000000G                                     ADD     #AD$$SZ,R2   ;Point to next entry in table
35 005340 020227 000000G                                     CMP     R2,#ALCEND    ;Checked all entries?
36 005344 103770                                     BLO     7$          ;Keep looking if not
37 005346 000421                                     BR      9$          ;Device was not allocated by anyone
38 005350 116200 000000G 5$: MOVB   AD$JOB(R2),R0 ;Get # of job that owns this device
39 005354 126061 000000G 000000G CMPB    LNPRIM(R0),LNPRIM(R1) ;Is device owned by us?
40 005362 001407                                     BEQ     6$          ;Br if yes
41 005364 006200                                     ASR     R0            ;Convert job index # to job #
42 005366 010037 000000G                                     MOV     R0,URO        ;Return owner job # in R0
43 005372 012700 000001                                     MOV     #1,R0         ;Error 1 -- Device is allocated to someone
44 005376 000137 015370'                                     JMP     USRERR
45 005402 005062 000000G 6$: CLR    AD$DVU(R2)   ;Say this table entry is free
46 005406 105062 000000G                                     CLRB   AD$JOB(R2)
47                                     ;
48                                     ; Finished
49                                     ;
50 005412 000137 015404' 9$: 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 005416  005037  000000G  TLCEMT: CLR    URO          ;Clear user's RO in case ALCCOM aborts
6 005422  004737  005636'      CALL   ALCCOM        ;Do common setup
7          ;
8          ; See if device is currently allocated
9          ;
10 005426  004737  005436'      CALL   ALCTST        ;See if device is allocated to another user
11          ;
12          ; Finished -- Device is not allocated to any other user
13          ;
14 005432  000137  015404'      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          ;   R1 = 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 005436 010246 ALCTST: MOV     R2, -(SP)
17 005440 010346      MOV     R3, -(SP)
18         ;
19         ; Clear returned value for R0 in case device is not in use or allocated
20         ;
21 005442 005037 000000G      CLR     URO          ; Say no use of device found yet
22         ;
23         ; See if the device is legal
24         ;
25 005446 005737 000000G      TST     FILSPC        ; Don't allow null device
26 005452 001413      BEQ     11$          ; Error if name null
27 005454 105737 002014'     TSTB    FILDVU        ; Trying to allocate TT?
28 005460 001410      BEQ     11$          ; Br if yes
29 005462 123737 002014' 000000G 6$:  CMPB   FILDVU, SYINDEX  ; Is this the system device?
30 005470 001010      BNE     12$          ; Br if not
31 005472 123737 002015' 000001G     CMPB   FILDVU+1, SYUNIT+1 ; Is this the system unit?
32 005500 001004      BNE     12$          ; Br if not
33 005502 012700 000002     11$:  MOV     #2, R0          ; Invalid device if null
34 005506 000137 015370'     JMP     USRERR
35         ;
36         ; See if the device is currently allocated to any user
37         ;
38 005512 012702 000000G     12$:  MOV     #ALCTBL, R2      ; Point to start of allocation table
39 005516 023762 002014' 000000G 1$:  CMP     FILDVU, AD$DVU(R2) ; Search for entry for this device and unit
40 005524 001423      BEQ     5$           ; Br if found it
41 005526 062702 000000G     ADD     #AD$$SZ, R2      ; Point to next entry
42 005532 020227 000000G     CMP     R2, #ALCEND      ; Searched all entries?
43 005536 103767      BLD     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 005540 004737 005756'     CALL   CHKUSE          ; See if device is being used by another job
49 005544 103007      BCC     13$          ; Br if no other jobs have device open
50 005546 006200      ASR     R0           ; Convert job index # to job #
51 005550 010037 000000G     MOV     R0, URO        ; Return in R0
52 005554 012700 000004     MOV     #4, R0         ; Error 4 -- Device in use by another job
53 005560 000137 015370'     JMP     USRERR
54 005564 006200     13$:  ASR     R0           ; Get # of job that is using the device
55 005566 010037 000000G     MOV     R0, URO        ; Return in R0
56 005572 000416      BR      9$           ;
57         ;

```

ALCTST -- See if device is allocated to another user

```

58          ; The device is allocated.
59          ; See if it is allocated to our job or to another job.
60          ;
61 005574 116203 000000G 5$:      MOV      AD#JOB(R2),R3  ;Get # of job to which dev is allocated
62 005600 010337 000000G      MOV      R3,URO          ;Return job # to user in R0
63 005604 006237 000000G      ASR      URO
64 005610 126361 000000G 000000G  CMPB     LNPRIM(R3),LNPRIM(R1) ;Is dev allocated to our family?
65 005616 001404          BEQ      9$              ;Br if yes
66 005620 012700 000001      MOV      #1,R0          ;Error 1 -- Device is allocated to another job
67 005624 000137 015370'      JMP      USRERR
68          ;
69          ; Device is not allocated to another user
70          ;
71 005630 012603 9$:      MOV      (SP)+,R3
72 005632 012602          MOV      (SP)+,R2
73 005634 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 005636 010146 ALCCOM: MOV      R1,-(SP)
12                                    ;
13                                    ; Gain exclusive access to USR data base
14                                    ;
15 005640 004737 010004' CALL     GETUSR      ;Get exclusive use of USR
16                                    ;
17                                    ; Move file spec to FILSPC and apply any assigns
18                                    ;
19 005644 013701 000002G MOV      EMTBLK+2,R1   ;Get pointer to device spec argument
20 005650 004737 007546' CALL     GETSPC      ;Setup FILSPC
21                                    ;
22                                    ; Check to see if the device is valid
23                                    ;
24 005654 005737 000000G TST      FILSPC      ;Is the device null?
25 005660 001430 BEQ      2$          ;Treat this as a special valid device
26 005662 004737 010226' CALL     CHKDEV      ;See if device is valid
27 005666 103016 BCC      1$          ;Br if CHKDEV says it is valid
28 005670 020437 000000G CMP      R4,CLDEVX    ;Is this a CL unit?
29 005674 001004 BNE      3$          ;Br if not
30 005676 020027 000000G CMP      R0,#CLTOTL   ;Valid CL unit number?
31 005702 103417 BLD      2$          ;Br if yes
32 005704 000412 BR       8$          ;Invalid device
33 005706 020437 000000G 3$:  CMP      R4,C1DEVX    ;Is this a C1 unit?
34 005712 001007 BNE      8$          ;Br if not
35 005714 020027 177770G CMP      R0,#CLTOTL-8. ;Is this a valid C1 unit number?
36 005720 002410 BLT      2$          ;Br if yes
37 005722 000403 BR       8$          ;Br if not
38 005724 020437 000000G 1$:  CMP      R4,LDDEVX    ;Is this a logical disk?
39 005730 001004 BNE      2$          ;Br if not -- Don't allow alloc of LD
40                                    ;
41                                    ; This is an invalid device
42                                    ;
43 005732 012700 000002 8$:  MOV      #2,R0          ;Error 2 -- Invalid device
44 005736 000137 015370' JMP      USRERR
45                                    ;
46                                    ; This is a valid device, Build the FILDVU word
47                                    ;
48 005742 110437 002014' 2$:  MOVB     R4,FILDVU    ;Set device index number
49 005746 110037 002015' MOVB     R0,FILDVU+1  ;Set unit number
50                                    ;
51                                    ; Finished
52                                    ;
53 005752 012601 MOV      (SP)+,R1
54 005754 000207 RETURN

```



```

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         ;  RO = Job index number of job that is accessing the device.
15         ;          (Zero returned if no job is using the device).
16         ;
17 005756 010246  CHKUSE: MOV     R2,-(SP)
18 005760 010346          MOV     R3,-(SP)
19 005762 010446          MOV     R4,-(SP)
20 005764 010546          MOV     R5,-(SP)
21         ;
22         ;  Get exclusive access to job context block buffer
23         ;
24 005766          OCALL  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 005774 005004          CLR     R4          ;Have not found any job accessing device
31 005776 013705 000000G  MOV     CSHDEV,R5          ;Point to start of mount table
32 006002 005765 000000G 6$:  TST     CD$DVU(R5)          ;Is this mount entry in use?
33 006006 001430          BEQ     7$          ;Br if not
34 006010 026537 000000G 002014'  CMP     CD$DVU(R5),FILDVU ;Does this match device of interest?
35 006016 001024          BNE     7$          ;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 006020 012702 000000G  MOV     #LSTSL,R2          ;Get index # of last job
41 006024 020201 8$:  CMP     R2,R1          ;Are we checking current job?
42 006026 001415          BEQ     10$          ;Br if yes -- It is ok
43 006030 010203          MOV     R2,R3          ;Get index number
44 006032 010246          MOV     R2,-(SP)          ;Preserve R2
45 006034 004737 015324'  CALL   CDJFLG          ;Get pointer to flag for this job
46 006040 010200          MOV     R2,RO          ;Get pointer to byte with flag bit
47 006042 012602          MOV     (SP)+,R2          ;Recover job number
48 006044 130310          BITB   R3,(RO)          ;Does this job have device mounted?
49 006046 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 006050 126261 000000G 000000G  CMPB   LNPRIM(R2),LNPRIM(R1) ;Is this job allowed to access the dev?
55 006056 001073          BNE     5$          ;Br if not
56 006060 010204          MOV     R2,R4          ;Remember # of job accessing device
57 006062 162702 000002 10$:  SUB     #2,R2          ;More jobs to check?

```

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

```

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

```

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

```

115 ;
116 ; Check next job
117 ;
118 006220 062702 000002 2$: ADD #2,R2 ;Get # of next job
119 006224 020227 000000G CMP R2,#LSTSL ;Checked all jobs?
120 006230 101727 BLOS 1$ ;Loop if more to check
121 ;
122 ; Finished checking all jobs
123 ; See if a friendly job is using the device
124 ;
125 006232 OCALL FREEXT ;Free context block buffer
126 006240 010500 MOV R5,R0 ;Get # of any job accessing the device
127 006242 000241 CLC ;Set flag saying device is free
128 006244 000405 BR 9$
129 ;
130 ; We found a job accessing the channel who is in conflict with testing job
131 ;
132 006246 5$: OCALL FREEXT ;Free context block buffer
133 006254 010200 MOV R2,R0 ;Return job index number in R0
134 006256 000261 SEC ;Set flag saying that device is in use
135 ;
136 ; Finished
137 ;
138 006260 012605 9$: MOV (SP)+,R5
139 006262 012604 MOV (SP)+,R4
140 006264 012603 MOV (SP)+,R3
141 006266 012602 MOV (SP)+,R2
142 006270 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 006272 004737 007134' MOUNT: CALL  MNTCOM          ; DO COMMON SETUP
6          ;
7          ; First dismount the device to clear the directory file entries
8          ;
9 006276 004737 006746'          CALL  DMTSUB          ; DISMOUNT THE DEVICE
10         ;
11         ; See if this device is mounted by other users.
12         ;
13 006302 004737 015210'          CALL  CSHTST         ; SEARCH FOR DEVICE IN MOUNT TABLE
14 006306 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 006310 113705 002014'          MOVVB FILDVU,R5        ; GET DEVICE INDEX NUMBER
20 006314 016500 000000G          MOV   DVSTAT(R5),R0    ; GET STATUS FLAGS FOR THIS DEVICE
21 006320 032700 000000G          BIT   #DS$DIR,R0        ; IS THIS A DIRECTORY STRUCTURED DEVICE?
22 006324 001503          BEQ   3$              ; BR IF NOT -- DON'T CACHE
23 006326 032700 000000G          BIT   #DS$NRD,R0        ; NON RT-11 DIRECTORY STRUCTURE (MAG TAPE)?
24 006332 001100          BNE   3$              ; BR IF YES -- DON'T CACHE
25 006334 032765 000000G 000000G BIT   #DX$NRD,DVFLAG(R5); Internal non-RT-dir flag set?
26 006342 001074          BNE   3$              ; Br if yes
27 006344 032765 000000G 000000G BIT   #DX$NMT,DVFLAG(R5); Is it flagged for no mount?
28 006352 001070          BNE   3$              ; Br if yes
29         ;
30         ; Look for a free entry in the mount table.
31         ;
32 006354 013705 000000G          MOV   CSHDEV,R5        ; POINT TO TABLE OF STRUCTURES
33 006360 005765 000000G 1$: TST   CD$DVU(R5)        ; SEARCH FOR A FREE SLOT IN THE TABLE
34 006364 001411          BEQ   2$              ; BR IF FOUND ONE
35 006366 062705 000000G          ADD   #CD$$SZ,R5        ; POINT TO NEXT TABLE ENTRY
36 006372 020537 000000G          CMP   R5,CSHDVN        ; HIT END OF TABLE?
37 006376 103770          BLO   1$              ; BR IF NOT
38 006400 012700 000001          MOV   #1,R0           ; NO FREE SLOTS IN TABLE
39 006404 000137 015370'          JMP   USRERR
40         ;
41         ; We found a free entry in the mount table.
42         ; Set up the entry for this device.
43         ;
44 006410 004737 015266' 2$: CALL  GETDVU          ; GET PHYSICAL DEVICE # AND UNIT #
45 006414 005065 000000G          CLR   CD$NAM(R5)        ; CLEAR FILE NAME (ASSUME NOT LOGICAL DISK)
46 006420 005065 000002G          CLR   CD$NAM+2(R5)
47 006424 012765 177777 000000G MOV   #177777,CD$TOP(R5); ASSUME NOT LOGICAL DISK FOR TOP BLOCK
48 006432 010365 000000G          MOV   R3,CD$DVU(R5)    ; STORE PHYSICAL DEVICE # AND UNIT #
49 006436 010465 000000G          MOV   R4,CD$BAS(R5)    ; STORE BASE BLOCK # IF LOGICAL DISK
50 006442 001420          BEQ   6$              ; BR IF NOT A LOGICAL DISK
51 006444 113703 002015'          MOVVB FILDVU+1,R3      ; THIS IS A LOGICAL DISK, GET LD UNIT #
52 006450 006303          ASL   R3              ; CONVERT UNIT # TO WORD TABLE INDEX
53 006452 066304 000000G          ADD   LDSIZE(R3),R4    ; GET # OF BLOCK ABOVE TOP OF LOGICAL DISK
54 006456 010465 000000G          MOV   R4,CD$TOP(R5)    ; SET TOP BLOCK NUMBER OF LOGICAL DISK
55 006462 072327 000002          ASH   #2,R3           ; TIMES 8 BYTES PER ENTRY
56 006466 062703 000000G          ADD   #LDNAME,R3      ; POINT TO ENTRY FOR NAME OF THIS LD FILE
57 006472 005723          TST   (R3)+          ; SKIP OVER DEVICE NAME

```

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

DISMNT -- Dismount a file structure

```

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 006540   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 006540  113700  000000G      MOVB    EMTBLK,RO      ;Get sub-function code
11 006544  120027  000001      CMPB    RO,#1         ;Clean out directory cache?
12 006550  001430                BEQ     CSHDMT        ;Br if yes
13 006552  120027  000002      CMPB    RO,#2         ;Dismount all devices for job?
14 006556  001442                BEQ     DMTALL        ;Br if yes
15 006560  120027  000003      CMPB    RO,#3         ;Dismount a LD structure?
16 006564  001414                BEQ     1$           ;Br if yes
17 006566  120027  000004      CMPB    RO,#4         ;Get information about a LD structure?
18 006572  001413                BEQ     2$           ;Br if yes
19 006574  120027  000005      CMPB    RO,#5         ;Dismount all LD structures?
20 006600  001412                BEQ     3$           ;Br if yes
21          ;
22          ; This is a request to dismount a file structure
23          ;
24 006602  004737  007134'      CALL    MNTCOM        ;Do common setup
25          ;
26          ; Call DMTSUB subroutine to do real dismount work.
27          ;
28 006606  004737  006746'      CALL    DMTSUB        ;Do the dismount
29          ;
30          ; Finished
31          ;
32 006612  000137  015404'      JMP     USRXIT
33          ;
34          ; Dismount a LD structure
35          ;
36 006616  000137  000000G      1$:    JMP     LDDEMT      ;Dismount a LD
37          ;
38          ; Get information about a LD structure
39          ;
40 006622  000137  000000G      2$:    JMP     LDIEMT      ;Get info about a LD
41          ;
42          ; Dismount all LD structures
43          ;
44 006626  000137  000000G      3$:    JMP     LDAEMT      ;Dismount all LD's
45          ;
46          ;-----
47          ; CSHDMT removes all entries from the directory cache.
48          ;
49 006632  004737  010004'      CSHDMT: CALL    GETUSR      ;GET EXCLUSIVE USE OF USR
50 006636  013700  000000G      MOV     CSHHD,RO      ;POINT TO 1ST CACHED DIRECTORY ENTRY
51 006642  005060  000000G      1$:    CLR     FC$CDX(RO) ;SAY ENTRY IS EMPTY
52 006646  005060  000000G      CLR     FD$NAM(RO)
53 006652  016000  000000G      MOV     FC$LNK(RO),RO ;LINK TO NEXT ENTRY
54 006656  001371                BNE     1$           ;BRANCH BACK IF ANOTHER TO FREE
55 006660  000137  015404'      JMP     USRXIT        ;FINISHED

```

```

1          .SBTTL  DMTALL -- Dismount all mounted devices for job
2          ;-----
3          ; Dismount all devices for this job.
4          ;
5 006664   DMTALL:
6          ;
7          ; Gain exclusive access to USR data base
8          ;
9 006664   004737  010004'   CALL    GETUSR      ;Get exclusive access to USR
10         ;
11         ; Get flag bit that identifies our job in mount table entries
12         ;
13 006670   113703  000000G   MOVB   CORUSR,R3    ;Get our job index number
14 006674   005005                CLR    R5           ;Say no entry to point to yet
15 006676   004737  015324'   CALL   CDJFLG      ;Get flag bit for our job
16         ;
17         ; Locate each device this is mounted by our job
18         ;
19 006702   013705  000000G   MOV    CSHDEV,R5   ;Point to 1st cache device entry
20 006706   005765  000000G  1$:    TST    CD$DVU(R5) ;Is this entry in use?
21 006712   001406                BEQ    2$          ;Br if not
22 006714   010500                MOV    R5,R0       ;Get address of entry
23 006716   060200                ADD    R2,R0       ;Point to byte with our job flag bit
24 006720   130310                BITB   R3,(R0)     ;Is this device mounted by our job?
25 006722   001402                BEQ    2$          ;Br if not
26 006724   004737  006766'   CALL   DMTDEV      ;Dismount this entry
27 006730   062705  000000G  2$:    ADD    #CD$$SZ,R5 ;Point to next mount table entry
28 006734   020537  000000G   CMP    R5,CSHDVN  ;Checked all entries?
29 006740   103762                BLO   1$          ;Loop if not
30         ;
31         ; Say job has no logical disks mounted
32         ;
33         ;
34         ; MOV    #MAXLD,R0   ;Get # of LD units
35         ; MOV    #LDNAME,R5 ;Point to LD name table
36         ; 3$:  CLR    (R5)    ;Say this LD not in use
37         ; ADD    #8.,R5     ;Point to next LD table entry
38         ; SOB    R0,3$     ;Br if more to do
39         ; Finished
40         ;
41 006742   000137  015404'   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 006746 010546 DMTSUB: MOV      R5, -(SP)
9          ;
10         ;   Locate entry for device in cached-device table.
11         ;
12 006750 004737 015210'          CALL    CSHTST          ;Locate entry for device in cached dev table
13 006754 103402          BCS      9$              ;Br if device is not mounted
14         ;
15         ;   Device is mounted, dismount it.
16         ;
17 006756 004737 006766'          CALL    DMTDEV          ;Dismount the device
18         ;
19         ;   Finished
20         ;
21 006762 012605          9$:    MOV      (SP)+, R5
22 006764 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 006766 010246 DMTDEV: MOV     R2, -(SP)
9 006770 010346         MOV     R3, -(SP)
10         ;
11         ; Reset mount flag for our job.
12         ;
13 006772 113703 000000G         MOV     CORUSR, R3         ;Get our job index number
14 006776 004737 015324'         CALL    CDJFLG         ;GET MOUNT FLAG FOR OUR JOB
15 007002 140312                 BICB   R3, (R2)         ;RESET MOUNT FLAG FOR OUR JOB
16         ;
17         ; See if any other users still have device mounted.
18         ;
19 007004 010503                 MOV     R5, R3         ;Point to bytes with mount flags
20 007006 062703 000000G         ADD     #CD$JOB, R3
21 007012 012700 000000G         MOV     #CD$$UB, R0         ;GET # BYTES WITH MOUNT FLAGS
22 007016 105723 1#: TST     (R3)+         ;ANY OTHER JOBS HAVE THIS DEVICE MOUNTED?
23 007020 001042                 BNE    9$             ;BR IF YES
24 007022 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 007024 005765 000000G         TST    CD$BAS(R5)         ;IS THIS A LOGICAL DISK?
30 007030 001011                 BNE    2$             ;BR IF YES
31 007032 016500 000000G         MOV     CD$DVU(R5), R0         ;GET DEVICE # AND UNIT #
32 007036 120037 000000G         CMPB   R0, SYINDX         ;IS THIS SY DEVICE?
33 007042 001004                 BNE    2$             ;BR IF NOT
34 007044 000300                 SWAB   R0             ;GET UNIT # TO LOW-ORDER BYTE
35 007046 120037 000001G         CMPB   R0, SYUNIT+1         ;IS THIS SY UNIT?
36 007052 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 007054 005737 000000G 2#: TST    CSHALC         ;Is data caching genned into system?
42 007060 001404                 BEQ    5$             ;Br if not
43 007062 016503 000000G         MOV     CD$DVU(R5), R3         ;Get device and unit number for CSHCLN
44 007066 004777 000000G         CALL   @CSHCLN         ;Clean out the data cache
45         ;
46         ; Say device is no longer mounted
47         ;
48 007072 005065 000000G 5#: CLR    CD$DVU(R5)         ;SAY DEVICE IS NOT MOUNTED
49         ;
50         ; Remove file entries for this device from directory cache.
51         ;
52 007076 013700 000000G         MOV     CSHHD, R0         ;POINT TO FIRST CACHED DIRECTORY ENTRY
53 007102 020560 000000G 3#: CMP     R5, FC$CDX(R0)         ;IS FILE ON THIS DEVICE?
54 007106 001004                 BNE    4$             ;BR IF NOT
55 007110 005060 000000G         CLR    FC$CDX(R0)         ;REMOVE ENTRY FROM CACHED DIRECTORY
56 007114 005060 000000G         CLR    FD$NAM(R0)
57 007120 016000 000000G 4#: MOV     FC$LNK(R0), R0         ;CHAIN TO NEXT ENTRY

```

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

```

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 007214 010246 CPYMNT: MOV      R2,-(SP)
10 007216 010346      MOV      R3,-(SP)
11 007220 010446      MOV      R4,-(SP)
12 007222 010546      MOV      R5,-(SP)
13 007224 010204      MOV      R2,R4          ;Save job index in R4
14                                     ;
15                                     ; Begin to search through mount table for mounts for other job
16                                     ;
17 007226 013705 000000G      MOV      CSHDEV,R5          ;Point to start of mount table
18 007232 010403 1$:      MOV      R4,R3          ;Get # of job we are copying from
19 007234 004737 015324'      CALL     CDJFLG          ;Get job flag for other job
20 007240 130312      BITB     R3,(R2)          ;Is this entry mounted by other job?
21 007242 001405      BEQ      2$          ;Br if not
22 007244 113703 000000G      MOVB    CORUSR,R3          ;Get our job index #
23 007250 004737 015324'      CALL     CDJFLG          ;Get mount bit for our job
24 007254 150312      BISB     R3,(R2)          ;Say device is mounted by our job
25 007256 062705 000000G  2$:      ADD     #CD$$SZ,R5          ;Point to next mount table entry
26 007262 020537 000000G      CMP     R5,CSHDVN          ;Checked all entries?
27 007266 103761      BLO     1$          ;Br if not
28                                     ;
29                                     ; Finished
30                                     ;
31 007270 012605      MOV     (SP)+,R5
32 007272 012604      MOV     (SP)+,R4
33 007274 012603      MOV     (SP)+,R3
34 007276 012602      MOV     (SP)+,R2
35 007300 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 007302 010546 CLRDIR: MOV     R5, -(SP)
13 007304 013746 002014'      MOV     FILDVU, -(SP)      ; Save current device/file index info
14         ;
15         ; Set up device and unit numbers in FILDVU
16         ;
17 007310 110237 002014'      MOVVB   R2, FILDVU      ; Set device index number
18 007314 116300 000000G      MOVVB   C.DEVQ(R3), R0    ; Get device unit number
19 007320 042700 177770      BIC     #^C<7>, R0      ; Clear out all but unit number
20 007324 110037 002015'      MOVVB   R0, FILDVU+1    ; Set unit number
21         ;
22         ; See if this device is being cached
23         ;
24 007330 004737 015210'      CALL    CSHTST          ; Is this device being cached?
25 007334 103414              BCS     9$              ; Br if not
26         ;
27         ; Clean out directory cache entries for this device
28         ;
29 007336 013700 000000G      MOV     CSHHD, R0      ; Point to 1st cached dir entry
30 007342 020560 000000G 1$:  CMP     R5, FC$CDX(R0)  ; Is file on this device?
31 007346 001004              BNE     2$              ; Br if not
32 007350 005060 000000G      CLR     FC$CDX(R0)     ; Remove entry from directory cache
33 007354 005060 000000G      CLR     FD$NAM(R0)
34 007360 016000 000000G 2$:  MOV     FC$LNK(R0), R0  ; Chain to next cache entry
35 007364 001366              BNE     1$              ; Loop if more to check
36         ;
37         ; Finished
38         ;
39 007366 012637 002014' 9$:  MOV     (SP)+, FILDVU
40 007372 012605              MOV     (SP)+, R5
41 007374 000207              RETURN

```

USRCOM -- Common setup

```

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 007376 010146 USRCOM: MOV      R1,-(SP)
27 007400 013703 000000G      MOV      CHNADR,R3      ;GET ADDRESS OF CURRENT CHANNEL AREA
28         ;
29         ; Claim USR data base for us
30         ;
31 007404 004737 010004'      CALL     GETUSR      ;GAIN EXCLUSIVE ACCESS TO USR DATA BASE
32         ;
33         ; Make sure channel is closed now.
34         ;
35 007410 032763 000000G 000000G      BIT      #CS$OPN,C.CSW(R3); IS CHANNEL OPEN NOW?
36 007416 001403      BEQ      2$      ;BR IF NOT OPEN NOW
37 007420 005000      CLR      R0      ;RETURN ERROR CODE OF 0
38 007422 000137 015370'      JMP      USRERR
39         ;
40         ; Move file spec to FILSPC and apply any assigns.
41         ;
42 007426 013701 000002G 2$:      MOV      EMTBLK+2,R1      ;GET ADDRESS OF FILE SPEC IN USER'S AREA
43 007432 042701 000001      BIC      #1,R1      ;MAKE SURE ADDRESS IS EVEN
44 007436 004737 007546'      CALL     GETSPC      ;GET FILE SPEC TO FILSPC
45         ;
46         ; Now check to see if the device is legal.
47         ;
48 007442 004737 010226'      CALL     CHKDEV      ; IS THE DEVICE LEGAL?
49 007446 103006      BCC      1$      ;BR IF RECOGNIZED DEVICE
50 007450 004737 015426'      CALL     ERRNAM      ;SET FILE SPEC FOR TSKMON ERROR MESSAGE
51 007454 012700 177776      MOV      #-2,R0      ;INVALID DEVICE
52 007460 000137 015370'      JMP      USRERR
53         ; (At this point R4 has device table index and R0 has device unit #)
54         ; Set up FILDVU word.
55 007464 110437 002014' 1$:      MOVB     R4,FILDVU      ;SET DEVICE #
56 007470 110037 002015'      MOVB     R0,FILDVU+1      ;SET UNIT #
57         ;

```

```
58 ; Set up information in channel.
59 ;
60 007474 110063 0000000 MOVB R0,C.DEVQ(R3) ;SET DEVICE UNIT #
61 007500 010463 0000000 MOV R4,C.CSW(R3) ;SET DEVICE TABLE INDEX # IN CSW
62 007504 052763 0000000 0000000 BIS #CS$OPN,C.CSW(R3);MARK CHANNEL AS OPEN
63 007512 005063 0000000 CLR C.SBLK(R3) ;STARTING BLOCK # OF FILE
64 007516 005063 0000000 CLR C.LENG(R3) ;ALLOCATED LENGTH OF FILE
65 007522 005063 0000000 CLR C.USED(R3) ;NO BLOCKS WRITTEN TO FILE YET
66 007526 105063 0000000 CLRB 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 007532 004737 010446' CALL CHKALC ;Check for allocation problems
71 ;
72 ; See if we are authorized to access this device and file.
73 ;
74 007536 004737 010602' CALL CHKACC ;CHECK FOR ACCESS AUTHORIZATION
75 ;
76 ; Return successfully
77 ;
78 007542 012601 MOV (SP)+,R1
79 007544 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 007546 010146 GETSPC: MOV      R1,-(SP)
14 007550 010246      MOV      R2,-(SP)
15 007552 005037 002044'      CLR      ASNSIZ          ;NO ASSIGN SIZE YET
16         ;
17         ; Copy the file specification from the user's area into FILSPC.
18         ;
19 007556 012702 000000G      MOV      #FILSPC,R2      ;Copy spec into FILSPC
20 007562 004737 007720'      CALL     CPYSPC          ;Copy file spec from user's area
21         ;
22         ; Apply any assignments to this file spec.
23         ;
24 007566 013700 000000G      MOV      FILSPC,R0      ;GET DEVICE NAME
25 007572 001447              BEQ      4$              ;BR IF NULL NAME
26 007574 032737 000000G 000000G BIT      #AF$BYA,RUNFLG  ;Should we bypass logical assignment?
27 007602 001043              BNE      4$              ;Br if yes
28 007604 012702 000000G      MOV      #ASNTBL,R2      ;POINT TO START OF ASSIGN TABLE
29 007610 020062 000000G      2$:    CMP      R0,AT$LOG(R2)  ;SEARCH FOR DEVICE NAME IN ASSIGN TABLE
30 007614 001406              BEQ      3$              ;BR IF FOUND
31 007616 062702 000000G      ADD      #AT$$SZ,R2      ;TRY NEXT ENTRY
32 007622 020227 000000G      CMP      R2,#ASNEND      ;DONE ALL?
33 007626 103770              BLO      2$              ;BR IF NOT
34 007630 000430              BR       4$              ;NAME WAS NOT ASSIGNED
35         ;
36         ; Name was assigned. Substitute assigned name.
37         ;
38 007632 012701 000000G      3$:    MOV      #FILSPC,R1      ;POINT TO AREA WITH FILE SPEC
39 007636 016211 000000G      MOV      AT$DEV(R2),(R1) ;PUT IN REAL DEVICE NAME
40 007642 016200 000000G      MOV      AT$FIL(R2),R0    ;WAS FILE NAME ASSIGNED?
41 007646 001421              BEQ      4$              ;BR IF NOT
42 007650 010061 000002      MOV      R0,2(R1)          ;PUT IN FILE NAME
43 007654 016261 000002G 000004      MOV      AT$FIL+2(R2),4(R1)
44 007662 016200 000000G      MOV      AT$EXT(R2),R0    ;WAS FILE EXTENSION ASSIGNED?
45 007666 001402              BEQ      8$              ;BR IF NOT
46 007670 010061 000006      MOV      R0,6(R1)          ;PUT IN FILE EXT
47 007674 016200 000000G      8$:    MOV      AT$SIZ(R2),R0    ;WAS FILE SIZE ASSIGNED?
48 007700 001404              BEQ      4$              ;BR IF NOT
49 007702 010061 000010      MOV      R0,8.(R1)        ;PUT IN FILE SIZE
50 007706 010037 002044'      MOV      R0,ASNSIZ      ;RETURN ASSIGN SIZE IN ASNSIZ
51         ;
52         ; finished
53         ;
54 007712 012602      4$:    MOV      (SP)+,R2
55 007714 012601      MOV      (SP)+,R1
56 007716 000207      RETURN

```


CPYSPC -- Copy file specification from user's area

```

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         007720 010100          MOV      R1,R0          ;GET ADDRESS OF FILE SPEC BUFFER
15         007722 004737 000000G  CALL     UACHKW         ;MAKE SURE ADDRESS IS LEGAL
16         007726 103004          BCC     5$              ;BR IF ADDRESS IS OK
17         007730 012700 177766   MOV     #-12,R0        ;GET ERROR ABORT CODE
18         007734 000137 015356'  JMP     USRCLS         ;ABORT THE JOB
19         ;
20         ; Copy file specification from user's area to FILSPC.
21         ;
22         007740 032737 000000G 000000G 5$:  BIT     #UMODE,EMTPS   ;WAS EMT DONE IN USER OR KERNEL MODE?
23         007746 001410          BEQ     6$              ;BR IF KERNEL MODE
24         007750 013700 000000G  MOV     ODTBAS,R0      ;GET THE USER'S HIGH MEMORY LIMIT
25         007754 160100          SUB     R1,R0          ;SUBTRACT ADDRESS OF FILE SPEC AREA
26         007756 006200          ASR     R0              ;CONVERT TO NUMBER OF WORDS
27         007760 001410          BEQ     9$              ;BR IF NOTHING TO MOVE
28         007762 020027 000004   CMP     R0,#4          ;COMPARE WITH SIZE OF FULL FILE SPEC
29         007766 101402          BLOS   1$              ;BR IF WE MUST MOVE LESS BECAUSE OF MEMORY TOP
30         007770 012700 000004   6$:  MOV     #4.,R0          ;# WORDS TO MOVE
31         007774 106521          1$:  MFPD   (R1)+        ;GET A WORD FROM USER'S AREA
32         007776 012622          MOV     (SP)+,(R2)+   ;MOVE TO FILSPC
33         010000 077003          SOB     R0,1$
34         ;
35         ; Finished
36         ;
37         010002 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 010004 010546 GETUSR: MOV      R5, -(SP)
8          ;
9          ; See if we can access the USR
10         ;
11 010006 4$:      DISABL          ;;; DISABLE INTERRUPTS
12 010014 113700 000000G MOVB     USRJOB, R0      ;;; IS USR AVAILABLE NOW?
13 010020 001426 BEQ      1$          ;;; BR IF YES
14 010022 120037 000000G CMPB     R0, CORUSR      ;;; DO WE ALREADY OWN USR?
15 010026 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 010030          ENABL          ; ENABLE INTERRUPTS
21 010036 113705 000000G MOVB     EXCJOB, R5      ; REMEMBER IF WE HAVE EXCLUSIVE SYS USE
22 010042 105037 000000G CLR      EXCJOB      ; ENABLE OTHER JOBS TO RUN
23 010046 012700 000000G MOV      #S$QUSR, R0    ; PUT US IN QUEUE FOR USR
24 010052 004737 000000G CALL     QNSPNX      ; SUSPEND JOB AND WAIT FOR USR
25 010056 004737 000000G CALL     CHKABT      ; WERE WE ABORTED WHILE ASLEEP?
26 010062 020537 000000G CMP      R5, CORUSR      ; DO WE WANT EXCLUSIVE ACCESS TO SYSTEM?
27 010066 001347 BNE      4$          ; LOOP IF NOT
28 010070 110537 000000G MOVB     R5, EXCJOB      ; GET EXCLUSIVE ACCESS TO THE SYSTEM
29 010074 000744 BR       4$          ; GO TRY TO GET USR NOW
30         ;
31         ; We can get USR now.
32         ;
33 010076 113737 000000G 000000G 1$:     MOVB     CORUSR, USRJOB ;;; CLAIM USR FOR US
34 010104          2$:     ENABL          ; ENABLE INTERRUPTS
35 010112 005237 002026' INC      USRCNT      ; REMEMBER # TIMES WE CLAIMED USR
36         ;
37         ; Make sure USR is not being called from a completion routine.
38         ;
39 010116 105737 000000G TSTB     CURCP          ; ARE WE IN A COMPLETION ROUTINE NOW?
40 010122 001404 BEQ      3$          ; BR IF NOT
41 010124 012700 177765 MOV      #-13, R0      ; ABORT IF IN COMPLETION ROUTINE
42 010130 000137 015370' JMP      USRERR
43         ;
44         ; Initialize USR data base.
45         ;
46 010134 012737 177777 002002' 3$:     MOV      #-1, CURSEG      ; NO DIR SEG IN BUFFER NOW
47         ;
48         ; Finished
49         ;
50 010142 012605 MOV      (SP)+, R5
51 010144 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 010146 FREUSR: DISABL          ;;; DISABLE INTERRUPTS

```

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

CHKDEV -- See if requested device is legal

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 010226 010146
15 010230 010246
16
17
18
19 010232 013701 000000G
20 010236 005000
21 010240 071027 000050
22 010244 012702 177777
23 010250 005701
24 010252 001406
25 010254 162701 000036
26 010260 010102
27 010262 020227 000007
28 010266 101063
29 010270 070027 000050
30
31
32
33
34
35
36
37 010274 020137 002020'
38 010300 001403
39 010302 020137 002016'
40 010306 001007
41 010310 013704 000000G
42 010314 005702
43 010316 002014
44 010320 113702 000001G
45 010324 000411
46
47
48
49 010326 013704 000000G
50 010332 020164 000000G
51 010336 001404
52 010340 162704 000002
53 010344 002372
54 010346 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.
;

```

CHKDEV -- See if requested device is legal

```

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

```

CHKALC -- Check for device allocation

```

1
2
3
4
5
6
7
8
9
10 010446 010046
11 010450 010246
12
13
14
15 010452 105737 002014'
16 010456 001446
17
18
19
20 010460 012702 000000G
21 010464 013700 002014'
22 010470 020062 000000G
23 010474 001421
24 010476 062702 000000G
25 010502 020227 000000G
26 010506 103770
27
28
29
30
31 010510 113700 002014'
32 010514 032760 000000G 000000G
33 010522 001424
34 010524 004737 015426'
35 010530 012700 177751
36 010534 000137 015356'
37
38
39
40
41 010540 116202 000000G
42 010544 113700 000000G
43 010550 126260 000000G 000000G
44 010556 001406
45 010560 004737 015426'
46 010564 012700 177752
47 010570 000137 015356'
48
49
50
51 010574 012602
52 010576 012600
53 010600 000207

```

```

.SBTTL  CHKALC -- Check for device allocation
-----
;  CHKALC is called to determine if a device being accessed is available
;  to the current job due to allocation considerations.
;  If an allocation failure is detected, a fatal error return is taken.
;
;  Inputs:
;  FILDVU = Device index # and unit # of device to be checked.
;
CHKALC: MOV     R0,-(SP)
        MOV     R2,-(SP)
;
;  Don't do allocation test for TT
;
        TSTB   FILDVU           ;Device # 0 ==> TT
        BEQ    9$              ;Br if this device is TT
;
;  See if this device is in the allocation table.
;
        MOV    #ALCTBL,R2      ;Point to start of allocation table
        MOV    FILDVU,R0       ;Get device/unit id
1$:     CMP    R0,AD$DVU(R2)    ;Is this entry for this device?
        BEQ    2$              ;Br if yes
        ADD    #AD$$SZ,R2     ;Point to next table entry
        CMP    R2,#ALCEND     ;Have we checked all entries?
        BLO   1$              ;Br if not
;
;  This device is not in the allocation table.
;  See if it must be allocated before it can be accessed.
;
        MOVB   FILDVU,R0       ;Get the device index number
        BIT    #DX$RAL,DVFLAG(R0); Is allocation required before access?
        BEQ    9$              ;Br if not
        CALL   ERRNAM          ;Set file spec for Kmon error message
        MOV    #-27,R0         ;Set error code
        JMP    USRCLS          ;Error return from USR
;
;  This device is in the allocation table.
;  See if it is allocated to our job or to another job.
;
2$:     MOVB   AD$JOB(R2),R2    ;Get # of job that owns the device
        MOVB   CORUSR,R0       ;Get current job index number
        CMPB   LNPRIM(R2),LNPRIM(R0); Do we own the device?
        BEQ    9$              ;Br if yes
        CALL   ERRNAM          ;Set file spec for Kmon error message
        MOV    #-26,R0         ;Set error code
        JMP    USRCLS          ;Error return
;
;  Device access is ok
;
9$:     MOV    (SP)+,R2
        MOV    (SP)+,R0
        RETURN

```

CHKACC -- Check legality of file access

```

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$RON set in channel CSW if read-only access authorized.
15         ;
16 010602 010046  CHKACC: MOV     R0, -(SP)
17 010604 010146          MOV     R1, -(SP)
18 010606 010246          MOV     R2, -(SP)
19 010610 010346          MOV     R3, -(SP)
20 010612 010446          MOV     R4, -(SP)
21 010614 010546          MOV     R5, -(SP)
22 010616 113701 000000G  MOVB   CORUSR, R1          ; GET JOB INDEX NUMBER
23         ;
24         ;  See if we are accessing a logical disk with NOWRITE attribute set.
25         ;
26 010622 020437 000000G  CMP     R4, LDDEVX          ; IS THIS A LOGICAL DISK?
27 010626 001011          BNE     13$                ; BR IF NOT
28 010630 010002          MOV     R0, R2                ; GET UNIT NUMBER
29 010632 006302          ASL     R2                    ; CONVERT TO WORD TABLE INDEX
30 010634 032762 000000G 000000G  BIT     #LD$RON, LDFLAG(R2) ; WAS /NOWRITE SPECIFIED FOR DISK?
31 010642 001403          BEQ     13$                ; BR IF NOT
32 010644 052763 000000G 000000G  BIS     #CS$RON, C.CSW(R3) ; SAY READ-ONLY ACCESS ALLOWED
33         ;
34         ;  Allow access to all devices if user has BYPASS privilege
35         ;
36 010652 032737 000000G 000000G 13$:  BIT     #PO$BYP, PRIVCO    ; Does user have BYPASS privilege?
37 010660 001004          BNE     19$                ; Br if yes
38         ;
39         ;  See if TSXUCL is trying to access its command file
40         ;
41 010662 032761 000000G 000000G  BIT     #UCLRN, LSW7(R1) ; IS TSXUCL RUNNING?
42 010670 001402          BEQ     16$                ; Br if not
43 010672 000137 011250' 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 010676 032737 000000G 000000G 16$:  BIT     #PO$SYS, PRIVCO    ; IS THIS A PRIVILEGED USER?
48 010704 001026          BNE     1$                    ; BR IF PRIVILEGED USER
49 010706 032761 000000G 000000G  BIT     #NOIN, LSW3(R1) ; ARE WE RUNNING IN START-UP COMMAND FILE?
50 010714 001022          BNE     1$                    ; BR IF YES -- GIVE FULL ACCESS DURING STARTUP
51 010716 120437 000000G  CMPB   R4, SYINDX          ; Is device = SY?
52 010722 001017          BNE     1$                    ; Br if not SY
53 010724 120037 000001G  CMPB   R0, SYUNIT+1        ; Is unit = SY?
54 010730 001014          BNE     1$                    ; Br if not
55 010732 013702 000006G  MOV     FILSPC+6, R2        ; GET FILE EXTENSION
56 010736 020237 002022'  CMP     R2, R5OSYS          ; IS EXTENSION "SYS"?
57 010742 001531          BEQ     8$                    ; BR IF YES -- DISALLOW ACCESS

```

CHKACC -- Check legality of file access

```

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

```


CHKACC -- Check legality of file access

```

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

```

ACCMCH -- Compare file spec with [NOJACCESS] entry

```

1          .SBTTL  ACCMCH -- Compare file spec with [NOJACCESS] entry
2          ;-----
3          ; Check to see if dev/file specification matches a [NOJACCESS] table entry
4          ;
5          ; Inputs:
6          ;   R5      points to [NOJACCESS] entry to be tested
7          ;   CKADEV  contains device index in question
8          ;   CKAUNT  contains device unit # in question
9          ;   FILSPC  contains ^R/FILNAMEXT/ in question
10         ;
11         ; Outputs:
12         ;   C-clear entry matches
13         ;   C-set  entry does not match
14         ;
15 011266  ACCMCH:
16 011266  010146      MOV     R1, -(SP)
17 011270  010546      MOV     R5, -(SP)
18 011272  005765 000000G  TST     OF$FIL(R5)      ; Is this entry in use?
19 011276  001433      BEQ     7$      ; If not, certainly cannot match
20         ;
21         ; Check for match on device index and unit number
22         ;
23 011300  116500 000000G  MOVB    OF$DEV(R5), R0  ; Get [NOJACCESS] device index
24 011304  100411      BMI     2$      ; Wildcard always matches
25 011306  123700 002046'  CMPB    CKADEV, R0      ; Does the device index match?
26 011312  001025      BNE     7$      ; Br to no match if not
27 011314  116500 000000G  MOVB    OF$UNT(R5), R0  ; Get [NOJACCESS] unit number
28 011320  100403      BMI     2$      ; Wildcard always matches
29 011322  123700 002047'  CMPB    CKAUNT, R0      ; Does the unit # match?
30 011326  001017      BNE     7$      ; Br to no match if not
31         ;
32         ; Device and unit are an acceptable match
33         ; Check file specification
34         ;
35 011330  062705 000000G  2$:     ADD     #OF$FIL, R5  ; Point to [NOJACCESS] file spec
36 011334  012701 000002G  MOV     #FILSPC+2, R1    ; Point to file spec in question
37 011340  012700 0000003  MOV     #3, R0           ; Get # words to compare (FIL NAM EXT)
38 011344  022125      5$:     CMP     (R1)+, (R5)+    ; Compare FIL NAM EXT
39 011346  001404      BEQ     3$      ; Br if this one matches
40 011350  026527 177776 000000G  CMP     -2(R5), #WLDNAM  ; Doesn't match, was [NOJACCESS] wild?
41 011356  001003      BNE     7$      ; If not, no match
42 011360  077007      3$:     SOB     R0, 5$      ; This part matches, check rest?
43         ;
44 011362  000241      CLC                     ; Entry matches, signal success
45 011364  000401      BR      9$
46         ;
47 011366  000261      7$:     SEC                     ; Entry does not match, signal no match
48         ;
49 011370  012605      9$:     MOV     (SP)+, R5
50 011372  012601      MOV     (SP)+, R1
51 011374  000207      RETURN

```

FNDFIL -- Find file in directory

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15 011376 010246
16 011400 010346
17
18
19
20 011402 004737 013656'
21
22
23
24 011406 012700 000000G
25 011412 004737 013370'
26 011416 103415
27
28 011420 010103
29 011422 062703 000000G
30 011426 012702 000002G
31 011432 012700 000003
32 011436 022223
33 011440 001362
34 011442 077003
35
36
37
38 011444 004737 013622'
39
40
41
42 011450 000241
43 011452 012603
44 011454 012602
45 011456 000207

```

```

.SBTTL FNDFIL -- Find file in directory
-----
; FNDFIL is called to locate the directory entry for a permanent file.
;
; Inputs:
;   FILSPC = File spec to be searched for.
;
; Outputs:
;   R0 = Starting block number of file.
;   R1 = Address of directory entry (located in USRBUF).
;   USRBUF = Directory segment containing entry.
;   CURSEQ = Number of directory segment.
;   C-flag set if file not found.
;
FNDFIL: MOV     R2, -(SP)
        MOV     R3, -(SP)
;
; Read in directory segment number 1 and set up info about directory.
;
        CALL    RDSEG1          ; READ IN FIRST DIRECTORY SEGMENT
;
; Search for permanent file entry that matches our name.
;
2$:     MOV     #FS$PRM, R0      ; LOCATE NEXT PERM FILE ENTRY
        CALL    GETDIR
        BCS     3$              ; BR IF HIT END OF DIRECTORY
;
; Compare file names.
        MOV     R1, R3          ; GET ADDRESS OF DIRECTORY ENTRY
        ADD     #FD$NAM, R3     ; POINT TO FILE NAME IN DIR ENTRY
        MOV     #FILSPC+2, R2   ; POINT TO NAME WE WANT TO FIND
        MOV     #3, R0          ; COMPARE 3 WORDS
1$:     CMP     (R2)+, (R3)+     ; DO NAMES MATCH?
        BNE     2$              ; BR IF NOT
        SOB    R0, 1$          ; CHECK ALL OF NAME
;
; Found the file entry -- Calculate starting block number.
;
        CALL    SBCALC          ; CALCULATE STARTING BLOCK # FOR ENTRY
;
; Finished
;
4$:     CLC                     ; SIGNAL GOOD RETURN
3$:     MOV     (SP)+, R3
        MOV     (SP)+, R2
        RETURN

```

FNDFRE -- Find a free slot for a new file

```

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 011460   010346  FNDFRE: MOV     R3,-(SP)
18 011462   010446         MOV     R4,-(SP)
19         ;
20         ; Initialize tables that hold info about 2 largest free slots.
21         ;
22 011464   005037   002030'   CLR     L1SIZ      ;SIZE OF LARGEST FREE SLOT
23 011470   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 011474   004737   013656'   CALL    RDSEG1     ;READ IN 1ST DIRECTORY SEGMENT
28         ;
29         ; Consolidate entries in directory segment.
30         ;
31 011500   004737   013016'   7$:    CALL    CONSOL      ;CONSOLIDATE DIRECTORY SEGMENT
32         ;
33         ; Find next directory entry for a permanent file or a free slot.
34         ;
35 011504   063701   002006'   4$:    ADD     DIRSIZ,R1     ;POINT TO NEXT DIRECTORY ENTRY
36 011510   016100   000000G   MOV     FD$STA(R1),R0    ;PICK UP STATUS WORD FROM DIRECTORY ENTRY
37 011514   032700   000000C   BIT     #FS$PRM+FS$EMP+FS$EOS,R0;PERM FILE, EMPTY SLOT OR END OF SEGMENT?
38 011520   001771           BEQ     4$           ;BR IF NONE OF ABOVE
39 011522   032700   000000G   BIT     #FS$EMP,R0      ;IS THIS AN EMPTY SPACE ENTRY?
40 011526   001024           BNE     13$         ;BR IF YES
41 011530   032700   000000G   BIT     #FS$EOS,R0     ;IS THIS THE END OF SEGMENT MARKER ENTRY?
42 011534   001106           BNE     1$         ;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 011536   032700   000000G   BIT     #FS$PRO,R0     ;IS THIS FILE PROTECTED?
48 011542   001760           BEQ     4$         ;BR IF NOT
49 011544   010103           MOV     R1,R3       ;POINT TO NAME IN DIR ENTRY
50 011546   062703   000000G   ADD     #FD$NAM,R3
51 011552   012704   000002G   MOV     #FILSPC+2,R4   ;POINT TO NAME OF FILE BEING ENTERED
52 011556   012700   000003   MOV     #3,R0         ;GET # WORDS TO COMPARE
53 011562   022324   14$:    CMP     (R3)+,(R4)+   ;COMPARE FILE NAMES
54 011564   001347           BNE     4$         ;BR IF DIFFERENT
55 011566   077003           SOB     R0,14$     ;COMPARE ALL OF NAMES
56         ; Error: Protected file with same name.
57         ; Return error code 3.

```

FNDFRE -- Find a free slot for a new file

```

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

```

FNDFRE -- Find a free slot for a new file

```

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

```

172 012176 000401
173
174
175
176 012200 000241
177 012202 012604
178 012204 012603
179 012206 000207

BR 22\$
;
; Successful return
;
21\$: CLC ; SIGNAL GOOD RETURN
22\$: MOV (SP)+, R4
MOV (SP)+, R3
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 012210 010246 ADDENT: MOV     R2,-(SP)
17 012212 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 012214 010146      MOV     R1,-(SP)      ;SAVE ADDRESS OF ENTRY WE ARE WORKING ON
23 012216 005000      CLR     R0          ;FIND END OF SEGMENT ENTRY
24 012220 004737 013424' CALL    NXTDIR
25 012224 010103      MOV     R1,R3          ;GET ADDRESS OF END OF SEGMENT ENTRY
26 012226 012601      MOV     (SP)+,R1      ;GET BACK ADDRESS OF EMPTY FILE ENTRY
27 012230 013700 002006' MOV     DIRSIZ,R0      ;GET SIZE OF A DIRECTORY ENTRY
28 012234 060003      ADD     R0,R3          ;ADD SPACE NEEDED FOR 3 ENTRIES
29 012236 060003      ADD     R0,R3
30 012240 060003      ADD     R0,R3
31 012242 020327 002002' CMP     R3,#USRBUF+1024. ;IS THERE ROOM IN THIS SEGMENT?
32 012246 103403      BLD     1$          ;BR IF THERE IS ROOM
33         ;
34         ; Directory segment is full.
35         ; Attempt to split it.
36         ;
37 012250 004737 012510' CALL    SPLIT          ;SPLIT THE SEGMENT
38 012254 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 012256 016100 000000G 1$: MOV     FD$LEN(R1),R0      ;GET SIZE OF EMPTY FILE ENTRY
44 012262 010261 000000G MOV     R2,FD$LEN(R1)      ;PUT IN ALLOCATED SIZE OF TENTATIVE FILE
45 012266 160200      SUB     R2,R0          ;CALC # BLOCKS LEFT OVER IN EMPTY AREA
46 012270 010002      MOV     R0,R2
47 012272 012761 000000G 000000G MOV     #FS$TEN,FD$STA(R1); MARK ENTRY AS TENTATIVE
48 012300 113761 000000G 000000G MOVB   CORUSR,FD$JOB(R1); PUT IN JOB NUMBER
49 012306 113761 000000G 000000G MOVB   CHNNUM,FD$CHN(R1); PUT IN CHANNEL NUMBER
50 012314 005061 000000G CLR     FD$DAT(R1)      ;SAY NO CREATION DATE YET -- SET IN CLOSE
51 012320 010103      MOV     R1,R3          ;SAVE ADDRESS OF TENTATIVE ENTRY
52 012322 062703 000000G ADD     #FD$NAM,R3      ;POINT TO FIELD WHERE NAME GOES
53 012326 012700 000002G MOV     #FILSPC+2,R0      ;POINT TO FILE SPEC WE ARE CREATING FOR
54 012332 012023      MOV     (R0)+,(R3)+      ;MOVE IN FILE NAME
55 012334 012023      MOV     (R0)+,(R3)+
56 012336 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 012340 063701 002006'          ADD    DIRSIZ,R1          ;POINT TO FOLLOWING DIR ENTRY
62 012344 004737 012426'          CALL   INSERT           ;INSERT AN EMPTY FILE ENTRY
63 012350 010261 000000G          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 012354 163701 002006'          SUB    DIRSIZ,R1          ;POINT TO OUR NEW TENTATIVE FILE ENTRY
71 012360 163701 002006'          SUB    DIRSIZ,R1          ;POINT TO PREVIOUS ENTRY
72 012364 020127 000000C          CMP    R1,#USRBUF+DH$$SZ; WAS THERE A PREVIOUS ENTRY?
73 012370 103410                    BLO    3$                ;BR IF NOT
74 012372 032761 000000G 000000G BIT    #FS$TEN,FD$STA(R1); WAS PREVIOUS ENTRY A TENTATIVE FILE ENTRY?
75 012400 001404                    BEQ    3$                ;BR IF NOT
76 012402 063701 002006'          ADD    DIRSIZ,R1          ;POINT BACK TO OUR NEW TENTATIVE ENTRY
77 012406 004737 012426'          CALL   INSERT           ;INSERT A ZERO-LENGTH EMPTY FILE ENTRY
78 012412 063701 002006'          3$:   ADD    DIRSIZ,R1          ;POINT TO TENTATIVE FILE ENTRY WE CREATED
79          ;
80          ; Finished
81          ;
82 012416 000241                    CLC                      ;SIGNAL GOOD RETURN
83 012420 012603                    9$:   MOV    (SP)+,R3
84 012422 012602                    MOV    (SP)+,R2
85 012424 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 012426 010146  INSERT: MOV      R1,-(SP)
15         ;
16         ;  Locate end-of-segment entry.
17         ;
18 012430 012700 000000G  MOV      #FS$E0S,R0      ;SEARCH FOR END-OF-SEGMENT MARKER
19 012434 030061 000000G  BIT      R0,FD$STA(R1)  ;ARE WE POINTING TO IT NOW?
20 012440 001002  BNE     2$              ;BR IF YES
21 012442 004737 013424'  CALL    NXTDIR         ;FIND END OF SEGMENT ENTRY
22         ;
23         ;  Shove down all entries beyond insert point.
24         ;
25 012446 005721 2$:      TST      (R1)+      ;POINT BEYOND END-OF-SEGMENT WORD
26 012450 010100  MOV     R1,R0
27 012452 063700 002006'  ADD    DIRSIZ,R0      ;MAKE ROOM FOR NEW ENTRY
28 012456 014140 1$:      MOV     -(R1),-(R0)  ;SHOVE DOWN ALL ENTRIES
29 012460 020116  CMP    R1,(SP)        ;TILL WE REACH INSERT POINT
30 012462 101375  BHI    1$
31         ;
32         ;  Initialize the new entry.
33         ;
34 012464 013700 002006'  MOV    DIRSIZ,R0      ;SIZE OF ENTRY (BYTES)
35 012470 006200  ASR    R0              ;GET # WORDS
36 012472 005021 3$:      CLR    (R1)+      ;ZERO THE ENTRY
37 012474 077002  SOB    R0,3$
38 012476 012601  MOV    (SP)+,R1       ;GET ADDRESS OF NEW ENTRY
39 012500 012761 000000G 000000G  MOV    #FS$EMP,FD$STA(R1);SAY ENTRY IS EMPTY
40         ;
41         ;  Finished
42         ;
43 012506 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 012510 010246 SPLIT:  MOV     R2, -(SP)
18 012512 010346      MOV     R3, -(SP)
19 012514 010446      MOV     R4, -(SP)
20 012516 010546      MOV     R5, -(SP)
21 012520 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 012522 023737 002010' 002004'      CMP     DIRNSG, DIRHIS  ; IS THERE A FREE SEGMENT?
26 012530 101524      BLOS    10$      ; BR IF NOT
27 012532 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 012536 012701 000000C      MOV     #USRBUF+DH##SZ, R1; POINT TO FIRST ENTRY IN SEGMENT
35 012542 005000      CLR     RO      ; COUNT # ENTRIES IN RO
36 012544 032761 000000G 000000G 1$:  BIT     #FS#EOS, FD$STA(R1); IS THIS THE END OF SEGMENT MARKER?
37 012552 001004      BNE     2$      ; BR IF YES
38 012554 005200      INC     RO      ; COUNT ANOTHER ENTRY
39 012556 063701 002006'      ADD     DIRSIZ, R1      ; POINT TO NEXT ENTRY
40 012562 000770      BR     1$
41                                    ; Leave approximately half the entries in the current segment.
42 012564 006200 2$:  ASR     RO      ; GET 1/2 # ENTRIES
43 012566 013702 000000C      MOV     USRBUF+DH$BLK, R2; GET BASE BLOCK # OF 1ST FILE IN SEGMENT
44 012572 012701 000000C      MOV     #USRBUF+DH##SZ, R1; POINT TO 1ST ENTRY
45 012576 066102 000000G 3$:  ADD     FD$LEN(R1), R2  ; KEEP TRACK OF BLOCK #'S OF FILES
46 012602 063701 002006'      ADD     DIRSIZ, R1      ; POINT TO NEXT DIRECTORY ENTRY
47 012606 077005      SOB     RO, 3$      ; SKIP OVER 1/2 OF THE ENTRIES
48                                    ; Split on 1st permanent or tentative entry beyond mid-point.
49 012610 032761 000000C 000000G 4$:  BIT     #<FS$PRM+FS$TEN>, FD$STA(R1); IS THIS A PERM OR TENTATIVE ENTRY?
50 012616 001005      BNE     7$      ; BR IF YES
51 012620 066102 000000G      ADD     FD$LEN(R1), R2  ; KEEP TRACK OF STARTING BLOCK NUMBERS
52 012624 063701 002006'      ADD     DIRSIZ, R1      ; POINT TO NEXT ENTRY
53 012630 000767      BR     4$
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.

```

SPLIT -- Split a directory segment

```

58 ;
59 ; Calculate address of dir entry we are interested in after split.
60 012632 020501 7$: CMP R5,R1 ;WILL ENTRY GO IN OLD OR NEW SEGMENT?
61 012634 103404 BLO 5$ ;BR IF GOING IN OLD SEGMENT
62 012636 160105 SUB R1,R5 ;CALCULATE ITS ADDRESS IN NEW SEGMENT
63 012640 062705 000000C ADD #USRBUF+DH$$SZ,R5
64 012644 005004 CLR R4 ;REMEMBER ENTRY GOES IN NEW SEGMENT
65 ;
66 ; Truncate current segment and set link to new segment.
67 ;
68 012646 011146 5$: MOV (R1),-(SP) ;SAVE FD$STAT FOR ENTRY FOLLOWING SPLIT
69 012650 012711 000000G MOV #FS$E0S,(R1) ;SET END-OF-SEGMENT MARKER
70 012654 013746 000000C MOV USRBUF+DH$NXT,-(SP);SAVE LINK TO NEXT SEGMENT
71 012660 013703 002004' MOV DIRHIS,R3 ;GET # OF HIGHEST SEGMENT CURRENTLY IN USE
72 012664 005203 INC R3 ;GET # OF 1ST FREE SEGMENT
73 012666 010337 000000C MOV R3,USRBUF+DH$NXT;SET AS NEW LINK
74 012672 004737 014110' CALL WRTSEG ;WRITE OUT TRUNCATED OLD SEGMENT
75 ;
76 ; Now set up new segment.
77 ;
78 ; Set new header.
79 012676 012637 000000C MOV (SP)+,USRBUF+DH$NXT;SET FLINK
80 012702 010337 002002' MOV R3,CURSEG ;SET OUR SEGMENT #
81 012706 010237 000000C MOV R2,USRBUF+DH$BLK;STARTING BLOCK # FOR FILES IN SEGMENT
82 ; Slide up directory entries from end of segment.
83 012712 012611 MOV (SP)+,(R1) ;FIX FD$STA OF 1ST ENTRY
84 012714 012700 000000C MOV #USRBUF+DH$$SZ,R0;POINT TO TOP OF AREA FOR ENTRIES
85 012720 012702 002002' MOV #USRBUF+1024.,R2;POINT PAST END OF BUFFER
86 012724 160102 SUB R1,R2 ;GET # BYTES TO MOVE
87 012726 006202 ASR R2 ;GET # WORDS TO MOVE
88 012730 012120 6$: MOV (R1)+,(R0)+ ;MOVE UP ENTRIES
89 012732 077202 SOB R2,6$
90 ; Write out the new segment.
91 012734 004737 014110' CALL WRTSEG
92 ;
93 ; Update information in header of 1st (master) directory segment.
94 ;
95 012740 004737 013656' CALL RDSEG1 ;READ IN DIR SEG 1
96 012744 010337 000000C MOV R3,USRBUF+DH$HIS;SET # OF HIGHEST SEG IN USE
97 012750 010337 002004' MOV R3,DIRHIS
98 012754 004737 014110' CALL WRTSEG ;WRITE OUT SEG # 1
99 ;
100 ; Now read back in the segment that has the directory entry we are
101 ; interested in.
102 ;
103 012760 005704 TST R4 ;IS ENTRY IN OLD OR NEW DIRECTORY SEGMENT?
104 012762 001401 BEQ 9$ ;BR IF IN NEW SEGMENT
105 012764 010403 MOV R4,R3 ;GET NUMBER OF OLD SEGMENT
106 012766 010300 9$: MOV R3,R0 ;SET AS ARGUMENT TO RDSEG
107 012770 004737 013772' CALL RDSEG ;READ IN SEGMENT WITH ENTRY WE WANT
108 ;
109 ; Finished
110 ;
111 012774 010501 MOV R5,R1 ;GET ADDRESS OF ENTRY OF INTEREST
112 012776 000241 CLC ;SIGNAL GOOD RETURN
113 013000 000401 BR 11$
114 ;

```

```
115          ; Directory overflow
116          ;
117 013002 000261      10$: SEC          ; SIGNAL ERROR ON RETURN
118 013004 012605      11$: MOV      (SP)+,R5
119 013006 012604      MOV      (SP)+,R4
120 013010 012603      MOV      (SP)+,R3
121 013012 012602      MOV      (SP)+,R2
122 013014 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 013016 010146  CONSOL: MOV      R1, -(SP)
13 013020 010246      MOV      R2, -(SP)
14 013022 010346      MOV      R3, -(SP)
15 013024 010446      MOV      R4, -(SP)
16         ;
17         ; Get exclusive access to job context block buffer
18         ;
19 013026          OCALL  GETCXT      ;Get exclusive access to context block buffer
20 013034 105037 002050' CLRB    CNSLXD      ;Say we haven't changed anything yet
21         ;
22         ; Pass 1: Convert unopen tentative entries into empty entries.
23         ;
24 013040 012701 000000C MOV     #USRBUF+DH$$SZ,R1;POINT TO 1ST ENTRY
25 013044 013704 002006' MOV     DIRSIZ,R4      ;CARRY DIRECTORY ENTRY SIZE IN R4
26 013050 160401      SUB     R4,R1      ;POINT IN FRONT OF 1ST ENTRY FOR NXTDIR
27         ; Find the next tentative file entry.
28 013052 012700 000000G 3$: MOV     #FS$TEN,R0      ;LOOK FOR A TENTATIVE FILE ENTRY
29 013056 004737 013424' CALL    NXTDIR      ;DO SEARCH
30 013062 103440      BCS     5$      ;BR IF NO MORE TENTATIVE FILE ENTRIES
31         ; We found a tentative file entry.
32         ; R1 points to the entry.
33         ; See if the entry is open now.
34 013064 116102 000000G MOVVB  FD$JOB(R1),R2    ;GET # OF JOB USING ENTRY
35 013070 001427      BEQ     1$      ;BR IF NOT TSX JOB
36 013072 032702 000001 BIT     #1,R2      ;TSX JOB NUMBER CAN NEVER BE ODD
37 013076 001024      BNE     1$      ;BR IF NOT TSX JOB USING CHANNEL
38 013100 020227 000000G CMP     R2,#LSTSL    ;MAKE SURE JOB NUMBER IS NOT TOO BIG
39 013104 101021      BHI     1$      ;BR IF NOT TSX JOB
40 013106 032762 000000G 000000G BIT     #$DILUP,LSW(R2) ;IS THAT JOB LOGGED ON?
41 013114 001415      BEQ     1$      ;BR IF NOT
42 013116 116103 000000G MOVVB  FD$CHN(R1),R3    ;GET CHANNEL # ASSOCIATED WITH FILE
43 013122          OCALL  GETCHA      ;GET POINTER TO CHANNEL BLOCK
44 013130 032760 000000G 000000G BIT     #CS$OPN,C.CSW(R0);IS THAT CHANNEL OPEN?
45 013136 001404      BEQ     1$      ;BR IF NOT
46 013140 032760 000000G 000000G BIT     #CS$ENT,C.CSW(R0);IS CHANNEL OPEN FOR CREATING A NEW FILE?
47 013146 001005      BNE     2$      ;BR IF YES
48         ; Convert this tentative file entry to an empty file entry.
49 013150 012761 000000G 000000G 1$: MOV     #FS$EMP,FD$STA(R1);SAY THIS IS AN EMPTY FILE ENTRY
50 013156 105237 002050' INCB   CNSLXD      ;REMEMBER WE HAVE CHANGED THE SEGMENT
51         ; Check next entry.
52 013162 000733      2$: 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 013164 012701 000000C 5$: MOV #USRBUF+DH##SZ,R1;POINT TO 1ST ENTRY
6 013170 160401 SUB R4,R1 ;POINT IN FRONT OF 1ST ENTRY
7 013172 012700 000000G 11$: MOV #FS$EMP,RO ;SEARCH FOR NEXT EMPTY FILE ENTRY
8 013176 004737 013424' CALL NXTDIR
9 013202 103462 BCS 12$ ;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 013204 010102 7$: MOV R1,R2 ;GET ADDRESS OF CURRENT ENTRY
13 013206 060402 ADD R4,R2 ;POINT TO NEXT ENTRY
14 013210 032762 000000G 000000G BIT #FS$EMP,FD$STA(R2); IS NEXT ENTRY EMPTY TOO?
15 013216 001421 BEQ 8$ ;BR IF NOT
16 ; Combine the two entries.
17 013220 066261 000000G 000000G ADD FD$LEN(R2),FD$LEN(R1);COMBINE ALLOCATED SIZES
18 013226 105237 002050' INCB CNSLXD ;REMEMBER WE HAVE CHANGED THE SEGMENT
19 ; Remove the second empty entry.
20 013232 010146 MOV R1,-(SP) ;REMEMBER ADDRESS OF 1ST EMPTY ENTRY
21 013234 012700 000000G MOV #FS$EOS,RO ;SEARCH FOR END OF SEGMENT MARKER
22 013240 004737 013424' CALL NXTDIR
23 ; R1 now points to end of segment marker.
24 ; Move up entries over the one being removed (pointed to by R2).
25 013244 010200 MOV R2,RO ;GET ADDRESS OF ENTRY TO BE REMOVED
26 013246 060400 ADD R4,RO ;POINT TO FOLLOWING ENTRY
27 013250 012022 6$: MOV (RO)+,(R2)+ ;MOVE UP REST OF ENTRIES OVER ONE BEING REMOVED
28 013252 020001 CMP RO,R1 ;MOVED END-OF-SEGMENT MARKER YET?
29 013254 101775 BLOS 6$ ;BR IF NOT
30 013256 012601 MOV (SP)+,R1 ;GET BACK ADDRESS OF 1ST EMPTY ENTRY
31 013260 000751 BR 7$ ;SEE IF THERE ARE MORE EMPTIES TO BE MERGED
32 ;
33 ; This empty is not followed by any other empty entries.
34 ; See if it is of zero length and follows a permanent file entry.
35 ;
36 013262 005761 000000G 8$: TST FD$LEN(R1) ;IS THIS EMPTY OF ZERO LENGTH?
37 013266 001027 BNE 9$ ;BR IF NOT
38 013270 010102 MOV R1,R2 ;GET ADDRESS OF ENTRY
39 013272 160402 SUB R4,R2 ;GET ADDRESS OF PREVIOUS ENTRY
40 013274 020227 000000C CMP R2,#USRBUF+DH##SZ;ARE WE 1ST ENTRY?
41 013300 103422 BLO 9$ ;BR IF YES
42 013302 032762 000000G 000000G BIT #FS$PRM,FD$STA(R2); IS PREVIOUS ENTRY A PERMANENT FILE?
43 013310 001416 BEQ 9$ ;BR IF NOT
44 ; Remove a zero length empty entry following a permanent file entry.
45 013312 010246 MOV R2,-(SP) ;SAVE ADDRESS OF PERMANENT FILE ENTRY
46 013314 010102 MOV R1,R2 ;GET ADDRESS OF EMPTY ENTRY
47 013316 012700 000000G MOV #FS$EOS,RO ;SEARCH FOR END OF SEGMENT ENTRY
48 013322 004737 013424' CALL NXTDIR
49 013326 010200 MOV R2,RO ;GET ADDRESS OF EMPTY ENTRY
50 013330 060400 ADD R4,RO ;POINT TO FOLLOWING ENTRY
51 013332 012022 10$: MOV (RO)+,(R2)+ ;MOVE UP FOLLOWING ENTRIES OVER EMPTY ENTRY
52 013334 020001 CMP RO,R1 ;HAVE WE MOVED END-OF-SEGMENT MARKER YET?
53 013336 101775 BLOS 10$ ;BR IF NOT
54 013340 105237 002050' INCB CNSLXD ;REMEMBER WE HAVE CHANGED THIS SEGMENT
55 013344 012601 MOV (SP)+,R1 ;GET BACK ADDRESS OF PERMANENT FILE ENTRY
56 ; Look for next empty entry.
57 013346 000711 9$: BR 11$

```

```
58 ;  
59 ; Finished  
60 ;  
61 013350 12*: OCALL FREEXT ;Release context block buffer  
62 013356 012604 MOV (SP)+,R4  
63 013360 012603 MOV (SP)+,R3  
64 013362 012602 MOV (SP)+,R2  
65 013364 012601 MOV (SP)+,R1  
66 013366 000207 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$xxx entry type flags.
10         ; USRBUF = Current directory segment.
11         ;
12         ; Outputs:
13         ; R1 = Address of next directory entry of the specified type.
14         ; USRBUF = Directory segment that contains the entry.
15         ; C-flag set if no entry of the specified type can be found.
16         ;
17 013370 GETDIR:
18         ;
19         ; See if we can find entry of specified type in the current directory segment.
20         ;
21 013370 004737 013424' 1$:      CALL    NXTDIR      ;SEARCH FOR ENTRY OF SPECIFIED TYPE
22 013374 103012          BCC      9$          ;BR IF WE FOUND ONE
23         ;
24         ; Hit end of current directory segment.  See if there are more segments.
25         ;
26 013376 010046          MOV      RO,-(SP)
27 013400 013700 000000C  MOV      USRBUF+DH$NXT,RO;GET # OF NEXT SEGMENT IN LIST
28 013404 001404          BEQ      2$          ;BR IF REACHED END
29 013406 004737 013772' CALL      RDSEG      ;READ IN NEXT SEGMENT
30 013412 012600          MOV      (SP)+,RO    ;GET BACK TYPE FLAGS
31 013414 000765          BR       1$          ;SEARCH THIS SEGMENT
32         ;
33         ; No entry of the specified type was found.
34         ;
35 013416 012600 2$:      MOV      (SP)+,RO    ;CLEAN OFF STACK
36 013420 000261          SEC          ;SIGNAL FAILURE ON RETURN
37         ;
38         ; finished
39         ;
40 013422 000207 9$:      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          ;  R0 = FS$xxx directory entry type flags.
8          ;  R1 = Address of last directory entry.
9          ;
10         ;  Outputs:
11         ;  R1 = Address of directory entry found.
12         ;  C-flag set if no entry of specified type could be found.
13         ;
14 013424 063701 002006'  NXTDIR: ADD    DIRSIZ,R1    ;POINT TO NEXT DIRECTORY ENTRY
15 013430 030061 000000G    BIT    R0,FD$STA(R1) ;IS THIS ENTRY OF THE RIGHT TYPE?
16 013434 001006          BNE    1$          ;BR IF YES
17 013436 032761 000000G 000000G    BIT    #FS$E0S,FD$STA(R1); IS THIS THE END-OF-SEGMENT MARKER?
18 013444 001767          BEQ    NXTDIR      ;KEEP SEARCHING IF NOT
19          ; Hit end of segment.
20 013446 000261          SEC                ;SIGNAL FAILURE ON RETURN
21 013450 000401          BR     9$
22          ; Found entry.
23 013452 000241 1$:      CLC                ;SIGNAL SUCCESS ON RETURN
24 013454 000207 9$:      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 013456 032737 000001 000002G DIDDLE: BIT    #1,EMTBLK+2    ; DOES USER WANT TO DIDDLE WITH THE ENTRY?
19 013464 001455                BEQ      9$                ; BR IF NOT
20 013466 010246                MOV      R2,-(SP)
21 013470 010346                MOV      R3,-(SP)
22 013472 010446                MOV      R4,-(SP)
23 013474 010546                MOV      R5,-(SP)
24 013476 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 013500 012703 000300                MOV      #USRUCA,R3    ; GET ADDRESS OF AREA IN USER'S AREA
30 013504 012700 000000C                MOV      #FD$$SZ/2,R0  ; GET # WORDS TO MOVE
31 013510 012146                1$:   MOV      (R1)+,-(SP)    ; PICK UP WORD FROM DIRECTORY ENTRY
32 013512 106623                MTPD    (R3)+          ; MOVE TO AREA IN USER'S SPACE
33 013514 077003                SOB      R0,1$        ; MOVE ALL OF DIRECTORY ENTRY
34         ;
35         ; Use special EMT to get control back from user's routine.
36         ;
37 013516 013703 000000G                MOV      EMTCAD,R3     ; Get pointer to top of return addr stack
38 013522 012743 013566'                MOV      #5$,-(R3)    ; Push our return address
39 013526 010337 000000G                MOV      R3,EMTCAD    ; Save updated stack pointer
40 013532 106506                MFPD    SP            ; GET USER'S STACK POINTER
41 013534 012603                MOV      (SP)+,R3
42 013536 012746 000000G                MOV      #CPLEMT,-(SP) ; PUSH ADDR OF EMT INSTRUCTION ON USER'S STACK
43 013542 106643                MTPD    -(R3)
44 013544 010346                MOV      R3,-(SP)    ; NOW STORE UPDATED USER STACK POINTER
45 013546 106606                MTPD    SP
46         ;
47         ; Now enter user's routine in user mode.
48         ; (Use RTI to do this)
49         ; On entry to user's routine, R1 points to the directory entry.
50         ;
51 013550 012701 000300G                MOV      #USRUCA+FD$DAT,R1; MAKE R1 POINT TO DATE WORD OF DIRECTORY ENTRY
52 013554 013746 000000G                MOV      EMTPS,-(SP)  ; PS
53 013560 013746 000010G                MOV      EMTBLK+10,-(SP) ; PC = Address of user's routine
54 013564 000002                RTI                    ; ENTER USER'S ROUTINE
55         ;
56         ; Return here from user's routine (when EMT is done).
57         ;

```

DIDDLE -- Allow user to access directory entry

```

58          ; Move directory entry back to directory buffer.
59          ;
60 013566 011602          5$:      MOV      (SP),R2          ; GET ADDRESS WHERE WE SHOULD PUT DIR ENTRY
61 013570 012703 000300      MOV      #USRUC, R3        ; POINT TO AREA IN USER'S AREA WITH DIR ENTRY
62 013574 012700 000000C      MOV      #FD$$SZ/2, R0      ; GET # WORDS TO MOVE
63 013600 106523          2$:      MFPD     (R3)+          ; GET A WORD FROM USER'S DIRECTORY ENTRY
64 013602 012622          MOV      (SP)+, (R2)+      ; MOVE BACK TO OUR INTERNAL AREA
65 013604 077003          SOB      R0, 2$          ; MOVE ALL OF ENTRY
66          ;
67          ; Finished
68          ;
69 013606 012601          MOV      (SP)+, R1
70 013610 012605          MOV      (SP)+, R5
71 013612 012604          MOV      (SP)+, R4
72 013614 012603          MOV      (SP)+, R3
73 013616 012602          MOV      (SP)+, R2
74 013620 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 013622 010246 SBCALC: MOV      R2,-(SP)
13 013624 013700      MOV      USRBUF+DH$BLK,RO;GET BLOCK # OF 1ST FILE IN THIS DIR SEGMENT
14 013630 012702      MOV      #USRBUF+DH$$SZ,R2;POINT TO 1ST DIR ENTRY IN THIS SEGMENT
15 013634 020201 1$:      CMP      R2,R1          ;HAVE WE REACHED OUR ENTRY?
16 013636 001405      BEQ      2$          ;BR IF YES
17 013640 066200      ADD      FD$LEN(R2),RO    ;ADD SPACE ALLOCATED TO THAT FILE ENTRY
18 013644 063702      ADD      DIRSIZ,R2      ;POINT TO NEXT ENTRY
19 013650 000771      BR       1$
20 013652 012602 2$:      MOV      (SP)+,R2
21 013654 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          ;   CHNUM = 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 013656  RDSEG1:
17         ;
18         ; Read in directory segment # 1
19         ;
20 013656  012700  000001      MOV     #1,R0          ;SAY WE WANT SEG # 1
21 013662  004737  013772'    CALL    RDSEG        ;READ THE SEGMENT IN
22         ;
23         ; Set up parameters about the directory
24         ;
25 013666  012701  000002'    MOV     #USRBUF,R1    ;POINT TO BUFFER WITH DIRECTORY SEGMENT
26 013672  016137  000000G 002004'  MOV     DH$HIS(R1),DIRHIS;HIGHEST SEGMENT # USED
27 013700  016137  000000G 002010'  MOV     DH$NSG(R1),DIRNSG;NUMBER OF DIRECTORY SEGMENTS AVAILABLE
28 013706  001425          BEQ     5$           ;BR IF INVALID DIRECTORY
29 013710  023727  002010' 000037  CMP     DIRNSG,#31.   ;CHECK TO SEE IF VALID
30 013716  101021          BHI     5$           ;BR IF INVALID
31 013720  016100  000000G    MOV     DH$NEB(R1),R0 ;# EXTRA BYTES PER DIRECTORY ENTRY
32 013724  020027  001000    CMP     R0,#512.     ;CHECK IF REASONABLE
33 013730  101014          BHI     5$           ;BR IF INVALID
34 013732  062700  000000G    ADD     #FD$OPT,R0   ;# STANDARD BYTES PER DIRECTORY ENTRY
35 013736  010037  002006'    MOV     R0,DIRSIZ   ;TOTAL # BYTES PER DIRECTORY ENTRY
36 013742  026127  000000G 000000G  CMP     DH$BLK(R1),#DH$$BS;MAKE SURE BASE BLOCK # IS REASONABLE
37 013750  103404          BLO     5$           ;BR IF INVALID
38         ;
39         ; Point R1 in front of 1st directory entry so NXTDIR will get 1st real entry.
40         ;
41 013752  062701  000000G    ADD     #DH$$SZ,R1  ;POINT TO 1ST REAL DIRECTORY ENTRY
42 013756  160001          SUB     R0,R1        ;POINT BACK 1 ENTRY
43         ;
44         ; Finished
45         ;
46 013760  000207          RETURN
47         ;
48         ; Invalid directory
49         ;
50 013762  012700  177755      5$:    MOV     #UERR6,R0   ;FATAL ERROR
51 013766  000137  015356'    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 013772 020037 002002' RDSEG:  CMP      RO,CURSEG      ; IS DESIRED SEGMENT ALREADY IN BUFFER?
15 013776 001437          BEQ      1$              ; BR IF YES
16 014000 010001          MOV      RO,R1          ; GET SEG #
17         ; Convert segment # to absolute block #.
18 014002 010137 002002' 2$:   MOV      R1,CURSEG      ; SAVE CURRENT SEGMENT #
19 014006 005301          DEC      R1              ; 1ST DIRECTORY SEGMENT IS # 1
20 014010 006301          ASL      R1              ; CVT SEGMENT # TO BLOCK #
21 014012 062701 000000G  ADD      #DH$$BS,R1      ; BASE BLOCK # OF 1ST SEGMENT
22         ; Read in the segment.
23 014016          .READW  #USREMT,CHNNUM,#USRBUF,#512.,R1
24 014060 103006          BCC      1$              ; BR IF NO ERROR
25 014062 004737 015426' CALL     ERRNAM          ; SET FILE SPEC FOR TSKMON ERROR MESSAGE
26 014066 012700 177775  MOV      #-3,RO          ; DIRECTORY READ ERROR
27 014072 000137 015356' JMP      USRCLS
28         ; Set R1 to point to dummy directory entry in front of first real one.
29 014076 012701 000000C 1$:   MOV      #USRBUF+DH$$SZ,R1 ; POINT TO FIRST REAL ENTRY IN SEGMENT
30 014102 163701 002006' SUB      DIRSIZ,R1        ; POINT IN FRONT OF FIRST ONE
31 014106 000207          RETURN

```

WRTSEG -- Write directory segment

```

1
2
3
4
5
6
7
8
9 014110 010146
10 014112 013701 002002'
11 014116 005301
12 014120 006301
13 014122 062701 000000G
14 014126
15 014170 103006
16 014172 004737 015426'
17 014176 012700 177775
18 014202 000137 015356'
19 014206 012601
20 014210 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$xxx).
10         ; R3 = Address of CSW for channel.
11         ; FILSPC = File spec.
12         ;
13 014212  SPLDIR:
14         ;
15         ; Free the USR module.
16         ;
17 014212 004737 010146'  CALL  FREUSR          ;FREE THE USR DATA BASE FOR OTHERS TO USE
18         ;
19         ; Obtain exclusive access to special device data base.
20         ;
21 014216 004737 014610'  CALL  GETSPD          ;GAIN EXCLUSIVE ACCESS TO SPD DATA BASE
22         ;
23         ; Copy the untranslated device/file specification from the user's area
24         ; to a system buffer.
25         ;
26 014222 013701 000002G  MOV   EMTBLK+2,R1    ;Point to file spec in user's area
27 014226 042701 000001  BIC   #1,R1         ;Make sure address is even
28 014232 010246  MOV   R2,-(SP)    ;Save operation function code
29 014234 012702 000000G  MOV   #SPFLDV,R2    ;Point to buffer where name is to be stored
30 014240 004737 007720'  CALL  CPYSPC        ;Copy file spec to SPFLDV
31 014244 012602  MOV   (SP)+,R2     ;Restore operation function code
32 014246 005037 000000G  CLR   SPSIZE        ;HANDLER RETURNS FILE SIZE HERE
33 014252 005037 000000G  CLR   SPUSR         ;HANDLER RETURNS ERROR CODE HERE
34 014256 042763 000000C 000000G  BIC   #<CS$EOF!CS$ERR>,C.CSW(R3) ;IGNORE PRIOR ERROR
35         ;
36         ; Build an I/O queue entry to do the operation.
37         ;
38 014264 004737 000000G  CALL  GETQ          ;GET A FREE I/O QUEUE ENTRY
39 014270 010361 000000G  MOV   R3,Q.UCSW(R1) ;SET ADDRESS OF CHANNEL CSW
40 014274 013761 000000G 000000G  MOV   @#KPAR6,Q.PA6(R1);Save mapping for context block
41 014302 010105  MOV   R1,R5         ;Get address of Q element
42 014304 062705 000000G  ADD   #Q.ICSW,R5    ;Point to internal CSW cell
43 014310 010561 000000G  MOV   R5,Q.CSW(R1) ;Set up pointer to internal CSW cell
44 014314 010300  MOV   R3,RO        ;Get pointer to original channel block
45 014316 012025  MOV   (RO)+,(R5)+  ;Copy original channel block to internal one
46 014320 012025  MOV   (RO)+,(R5)+
47 014322 012025  MOV   (RO)+,(R5)+
48 014324 012025  MOV   (RO)+,(R5)+
49 014326 011015  MOV   (RO),(R5)
50 014330 016300 000000G  MOV   C.CSW(R3),RO ;Get the channel status word
51 014334 042700 177701  BIC   #^C76,RO     ;Isolate the device index number
52 014340 110061 000000G  MOVB  RO,Q.DEVX(R1) ;store the device index number
53 014344 013700 000006G  MOV   EMTBLK+6,RO  ;GET FILE SEQUENCE # FOR .ENTER
54 014350 020227 000000G  CMP   R2,#DF$ENT   ;ARE WE DOING A .ENTER?
55 014354 001402  BEQ   6$           ;BR IF YES
56 014356 013700 000004G  MOV   EMTBLK+4,RO  ;GET SEQUENCE # FOR .LOOKUP, .DELETE, ETC.
57 014362 010061 000000G  6$:  MOV   RO,Q.BLKN(R1) ;SET SEQUENCE # IN Q.BLKN FIELD

```

SPLDIR -- Directory operations for special devices

```

58 014366 013761 000004G 000000G      MOV      EMTBLK+4, Q.WCNT(R1); SET FILE ENTER SIZE IN Q.WCNT FIELD
59 014374 110261 000000G              MOVVB    R2, Q.FUNC(R1) ; SET OPERATION FUN CODE (LOOKUP, ENTER, ETC.)
60 014400 116361 000000G 000000G      MOVVB    C.DEVG(R3), Q.UNIT(R1); SET UNIT NUMBER
61 014406 113700 000000G              MOVVB    CORUSR, R0 ; Get current job index number
62 014412 006200              ASR      R0 ; Convert to job number
63 014414 110061 000000G              MOVVB    R0, Q.JOB(R1) ; Set job number in queue element
64 014420 072027 0000003              ASH     #3, R0 ; Position for Q.JNUM field
65 014424 042700 177407              BIC     #^C<370>, R0 ; Clear all but job number field
66 014430 150061 000000G              BISB    R0, Q.JNUM(R1) ; Set job # in queue element
67 014434 012704 000000G              MOV     #SPFLNM, R4 ; GET ADDRESS OF FILE SPEC BUFFER
68 014440 005005              CLR     R5 ; SET FOR SHIFT
69 014442 073427 177772              ASHC   #-6, R4 ; SHIFT LOW-ORDER 6 BITS INTO R5
70 014446 000241              CLC                    ;
71 014450 006005              ROR     R5 ; RIGHT JUSTIFY LOW-ORDER 6 BITS IN R5
72 014452 072527 177767              ASH     #-9, R5 ;
73 014456 062705 000000G              ADD     #VPAR6, R5 ; BIAS ADDRESS TO BE IN PAR6 RANGE
74 014462 010561 000000G              MOV     R5, Q.BUFF(R1) ; SET THIS AS VIRTUAL BUFFER ADDRESS
75 014466 010461 000000G              MOV     R4, Q.PAR(R1) ; SET PAR6 BASE OFFSET
76 014472 005061 000000G              CLR     Q.COMP(R1) ; NO COMPLETION ROUTINE
77 014476 013761 000000G 000000G      MOV     CHNNUM, Q.CHAN(R1); SET USER'S CHANNEL NUMBER
78 ;
79 ; Initiate the I/O operation.
80 ;
81 014504 004737 000000G              CALL    QIO ; QUEUE THE I/O OPERATION
82 ;
83 ; Wait for I/O to finish.
84 ;
85 014510              .WAIT   CHNNUM ; WAIT FOR OPERATION TO FINISH
86 014520 103003              BCC     3$ ; BR IF NO I/O ERROR
87 ;
88 014522 012737 177775 000000G      MOV     #-3, SPUSR ; SET I/O ERROR CODE
89 ;
90 ; Return handler reported file size to user in R0.
91 ;
92 014530 013737 000000G 000000G 3$:   MOV     SPSIZE, URO ; RETURN HDLR REPORTED FILE SIZE IN USER'S R0
93 014536 013705 000000G              MOV     SPUSR, R5 ; GET HANDLER REPORTED ERROR CODE
94 ;
95 ; Release special device data base area.
96 ;
97 014542 004737 014652'              CALL    FRESPD ; FREE SPD AREA
98 ;
99 ; See if we should purge the channel.
100 ;
101 014546 020227 000000G              CMP     R2, #DF$LOK ; DOING A LOOKUP?
102 014552 001405              BEQ     4$ ; BR IF YES
103 014554 020227 000000G              CMP     R2, #DF$ENT ; DOINT AN ENTER?
104 014560 001402              BEQ     4$ ; BR IF YES
105 014562 005063 000000G              CLR     C.CSW(R3) ; PURGE THE CHANNEL
106 ;
107 ; See if handler reported an error
108 ;
109 014566 010500 4$:   MOV     R5, R0 ; DID HANDLER RETURN AN ERROR CODE?
110 014570 001002              BNE     5$ ; BR IF YES
111 014572 000137 000000G              JMP     EMTXIT ; NORMAL EMT EXIT
112 014576 5$:
113 ; ; CLR     C.CSW(R3) ; PURGE CHANNEL IF ERROR OCCURED
114 ; ; This change (from CLR to BIC) is implemented to allow "some" information

```

SPLDIR -- Directory operations for special devices

```
115          ; to be returned by .CSTAT on failed lookups. RT-11 actually maintains
116          ; a valid C.CSW word on this type of error so their .CSTAT always works.
117 014576 042763 000000G 000000G          BIC      #CS$OPN,C.CSW(R3) ;SPECIAL PURGE FOR NON-RT DIR DEVICES
118 014604 000137 000000G          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 014610  113700  000000G  GETSPD:  MOV  SPDJOB,RO  ; IS SPD DATA BASE FREE NOW?
8 014614  001412          BEQ  1$          ; BR IF YES
9 014616  120037  000000G          CMPB RO,CORUSR  ; HAVE WE ALREADY GOT IT?
10 014622  001412          BEQ  2$          ; BR IF YES
11         ;
12         ; Someone else owns SPD data base.
13         ; Suspend our job until it becomes free.
14         ;
15 014624  012700  000000G          MOV  #S$QSPD,RO  ; QUEUE FOR SPD DATA BASE
16 014630  004737  000000G          CALL  QNSPNX    ; PUT US AT TAIL OF QUEUE FOR IT
17 014634  004737  000000G          CALL  CHKABT   ; SEE IF WE WERE ABORTED WHILE ASLEEP
18 014640  000763          BR   GETSPD    ; NOW TRY TO GET SPD
19         ;
20         ; We can get SPD data base now.
21         ;
22 014642  113737  000000G 000000G 1$:  MOV  CORUSR,SPDJOB ; CLAIM SPD DATA BASE FOR US
23 014650  000207          2$:  RETURN
24         ;
25         .SBTTL  FRESPD -- Free special device data base
26         ;-----
27         ; Free the special device data base and restart any user who is waiting
28         ; for it.
29         ;
30 014652  123737  000000G 000000G FRESPD: CMPB  CORUSR,SPDJOB ; DO WE CURRENTLY OWN SPD?
31 014660  001006          BNE  1$          ; BR IF NOT
32 014662  105037  000000G          CLRB SPDJOB    ; FREE IT
33         ;
34         ; Restart any job that is waiting for SPD data base.
35         ;
36 014666  012700  000000G          MOV  #S$QSPD,RO  ; GET SPD WAIT QUEUE STATE CODE
37 014672  004737  000000G          CALL  UREGD    ; RESTART ANY WAITING JOB
38         ;
39         ; Finished
40         ;
41 014676  000207          1$:  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 014700 010246 CSHADD: MOV      R2,-(SP)
10 014702 010346      MOV      R3,-(SP)
11 014704 010446      MOV      R4,-(SP)
12 014706 010546      MOV      R5,-(SP)
13          ;
14          ; See if we are caching file entries for this device
15          ;
16 014710 004737 015210' CALL     CSHTST      ;ARE WE TO CACHE ENTRIES FOR THIS DEVICE?
17 014714 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 014716 012702 000000C      MOV      #CSHHD-FC$LNK,R2;DUMMY POINTER TO PHANTOM FIRST ENTRY
23 014722 010203 1$:      MOV      R2,R3      ;REMEMBER ADDRESS OF PREVIOUS ENTRY
24 014724 016202 000000G      MOV      FC$LNK(R2),R2 ;CHAIN FORWARD TO NEXT ENTRY IN LIST
25 014730 005762 000000G      TST      FC$CDX(R2)  ;IS THIS ENTRY UNUSED?
26 014734 001403      BEQ      2$        ;BR IF YES
27 014736 005762 000000G      TST      FC$LNK(R2)  ;IS THIS THE LAST ENTRY IN LIST?
28 014742 001367      BNE      1$        ;BR IF NOT
29          ;
30          ; Relink cache entry at head of chain (it is most recently used).
31          ;
32 014744 016263 000000G 000000G 2$:      MOV      FC$LNK(R2),FC$LNK(R3);RELINK CHAIN AROUND US
33 014752 013762 000000G 000000G      MOV      CSHHD,FC$LNK(R2);NOW LINK US AT FRONT OF LIST
34 014760 010237 000000G      MOV      R2,CSHHD
35          ;
36          ; Copy info from file directory entry to cache entry.
37          ;
38 014764 010203      MOV      R2,R3      ;GET ADDRESS OF CACHE ENTRY
39 014766 010104      MOV      R1,R4      ;GET ADDRESS OF DIRECTORY ENTRY
40 014770 012700 000000G      MOV      #FD$$SZ,R0   ;GET # BYTES TO MOVE
41 014774 006200      ASR      R0          ;GET # WORDS TO MOVE
42 014776 012423 3$:      MOV      (R4)+,(R3)+ ;COPY DIRECTORY INFO TO CACHE ENTRY
43 015000 077002      SOB      R0,3$
44          ;
45          ; Store pointer to cached-device table entry into file cache entry
46          ;
47 015002 010562 000000G      MOV      R5,FC$CDX(R2) ;REMEMBER WHICH DEVICE FILE IS ASSOC WITH
48          ;
49          ; Store starting block number of file
50          ;
51 015006 004737 013622'      CALL     SBCALC      ;CALCULATE STARTING BLOCK # OF FILE
52 015012 010062 000000G      MOV      R0,FC$SBL(R2) ;SAVE STARTING BLOCK #
53          ;
54          ; Finished
55          ;
56 015016 012605 9$:      MOV      (SP)+,R5
57 015020 012604      MOV      (SP)+,R4

```

58 015022 012603
59 015024 012602
60 015026 000207

MOV (SP)+,R3
MOV (SP)+,R2
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 015030 010146 CSHDEL: MOV      R1,-(SP)
10         ;
11         ; Search for file entry in cache table.
12         ;
13 015032 062701 000000G      ADD      #FD$NAM,R1      ;POINT TO FILE NAME IN DIRECTORY ENTRY
14 015036 004737 015054'      CALL     CSHCHK        ;SEE IF ENTRY IS IN DIRECTORY CACHE
15 015042 103402              BCS      4$                ;BR IF FILE ENTRY IS NOT IN CACHE
16         ;
17         ; Found entry in cache table.
18         ; Mark it as deleted.
19         ;
20 015044 005061 000000G      CLR      FC$CDX(R1)      ;MARK ENTRY AS DELETED
21         ;
22         ; Finished
23         ;
24 015050 012601              4$:   MOV      (SP)+,R1
25 015052 000207              RETURN

```

CSHCHK -- Search for file entry in directory cache

```

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         ; RO = Starting block # of file.
16         ;
17 015054 010246 CSHCHK: MOV      R2,-(SP)
18 015056 010346      MOV      R3,-(SP)
19 015060 010546      MOV      R5,-(SP)
20 015062 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 015064 004737 015210' CALL     CSHTST          ;SEE IF THIS DEVICE IS BEING CACHED
25 015070 103442      BCS      4$          ;BR IF DEVICE IS NOT BEING CACHED
26         ;
27         ; Search for file entry in cache table.
28         ;
29 015072 012701 000000C      MOV      #CSHHD-FC$LNK,R1;GET POINTER TO DUMMY STARTING POINT
30 015076 010102      1$:     MOV      R1,R2          ;REMEMBER ADDRESS OF PREVIOUS ENTRY
31 015100 016101 000000G      MOV      FC$LNK(R1),R1  ;GET ADDRESS OF NEXT ENTRY IN LIST
32 015104 020561 000000G      CMP      R5,FC$CDX(R1)  ;IS FILE ON CACHED DEVICE?
33 015110 001012      BNE      2$          ;BR IF NOT
34 015112 010300      MOV      R3,RO          ;POINT TO FILE NAME
35 015114 022061 000000G      CMP      (RO)+,FD$NAM(R1);COMPARE FILE NAMES
36 015120 001006      BNE      2$
37 015122 022061 000002G      CMP      (RO)+,FD$NAM+2(R1)
38 015126 001003      BNE      2$
39 015130 021061 000004G      CMP      (RO),FD$NAM+4(R1)
40 015134 001404      BEQ      3$          ;BR IF FOUND ENTRY FOR FILE
41 015136 005761 000000G      2$:     TST      FC$LNK(R1)  ;IS THIS LAST ENTRY IN LIST?
42 015142 001355      BNE      1$          ;BR IF NOT
43 015144 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 015146 016162 000000G 000000G 3$:     MOV      FC$LNK(R1),FC$LNK(R2);RELINK LIST AROUND US
48 015154 013761 000000G 000000G      MOV      CSHHD,FC$LNK(R1);LINK US IN AT HEAD OF LIST
49 015162 010137 000000G      MOV      R1,CSHHD
50         ;
51         ; Return with R1 = address of cache entry, RO = Starting block #.
52         ;
53 015166 016100 000000G      MOV      FC$SBL(R1),RO  ;GET STARTING BLOCK # FOR FILE
54 015172 000241      CLC          ;SIGNAL SUCCESSFUL RETURN
55 015174 000401      BR       5$
56         ;
57         ; Can't find file entry in cache.

```



```
58 ;  
59 015176 000261 4$: SEC ; SIGNAL UNSUCCESSFUL RETURN  
60 ;  
61 ; Return  
62 ;  
63 015200 012605 5$: MOV (SP)+, R5  
64 015202 012603 MOV (SP)+, R3  
65 015204 012602 MOV (SP)+, R2  
66 015206 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 015210  010346  CSHTST: MOV     R3, -(SP)
14 015212  010446          MOV     R4, -(SP)
15         ;
16         ; If this file is on a logical disk, set up information about the LD.
17         ;
18 015214  004737  015266' CALL    GETDVU          ;GET INFO ABOUT PHYSICAL DEVICE AND UNIT
19         ;
20         ; Search for device information in table of cached devices
21         ;
22 015220  013705  000000G MOV     CSHDEV, R5      ;POINT TO TABLE OF MOUNTED DEVICES
23 015224  020365  000000G 2$:    CMP     R3, CD$DVU(R5) ;DO DEVICE # AND UNIT #'S MATCH?
24 015230  001003          BNE     5$              ;BR IF NOT
25 015232  020465  000000G CMP     R4, CD$BAS(R5) ;DOES BASE BLOCK # OF DISK MATCH?
26 015236  001407          BEQ     3$              ;BR IF YES -- THIS DEVICE IS CACHED
27 015240  062705  000000G 5$:    ADD     #CD$$SZ, R5 ;POINT TO NEXT TABLE ENTRY
28 015244  020537  000000G CMP     R5, CSHDEVN    ;REACHED END OF TABLE?
29 015250  103765          BLO     2$              ;BR IF NOT
30 015252  000261          SEC                      ;SIGNAL THAT DEVICE IS NOT TO BE CACHED
31 015254  000401          BR      4$
32 015256  000241  3$:    CLC                      ;SIGNAL THAT DEVICE IS TO BE CACHED
33         ;
34         ; Finished
35         ;
36 015260  012604  4$:    MOV     (SP)+, R4
37 015262  012603          MOV     (SP)+, R3
38 015264  000207          RETURN

```

GETDVU -- Get device # & unit # info

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 015266 013703 002014'
15 015272 005004
16 015274 120337 000000G
17 015300 001010
18 015302 000303
19 015304 042703 177770
20 015310 006303
21 015312 016304 000000G
22 015316 016303 000000G
23
24
25
26 015322 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 015324 006203  CDJFLG: ASR      R3          ; Convert job index to index by 1
15 015326 005303          DEC      R3          ; Make base job number 0
16 015330 005002          CLR      R2          ; Clear for divide
17 015332 071227 000010  DIV      #8.,R2       ; Divide by 8 jobs per byte
18 015336 060502          ADD      R5,R2       ; Get address of byte within
19 015340 062702 000000G  ADD      #CD$JOB,R2     ; cached-device table entry
20 015344 012700 000001  MOV      #1,R0         ; Get a mount flag
21 015350 072003          ASH      R3,R0         ; Position flag according to job number
22 015352 010003          MOV      R0,R3         ; Return flag in R3
23 015354 000207          RETURN

```

USRXIT -- Exit from USR

```

1
2
3
4
5
6
7
8
9
10 015356 013703 000000G
11 015362 001402
12 015364 005063 000000G
13 015370 010046
14 015372 004737 010146'
15 015376 012600
16 015400 000137 000000G
17
18
19
20 015404 004737 010146'
21 015410 000137 000000G
22
23
24
25
26
27
28 015414 010005
29 015416 013704 000000G
30 015422 000137 000000G

```

```

.SBTTL USRXIT -- Exit from USR
-----
; USRXIT is the exit point for EMT processing within the USR.
; USRCLS sets an I/O error code, purges the channel then aborts the emt.
; USRERR sets an I/O error code but does not purge the channel.
;
; Inputs:
; RO = error code [USRERR, and USRCLS only]
;
USRCLS: MOV     CHNADR,R3      ;GET ADDRESS OF CURRENT CHANNEL
        BEQ     USRERR        ;BR IF NONE
        CLR     C.CSW(R3)     ;PURGE THE CHANNEL
USRERR: MOV     RO,-(SP)      ;SAVE ERROR CODE
        CALL    FREUSR        ;FREE USR DATA BASE IF WE HAVE IT
        MOV     (SP)+,RO      ;GET BACK ERROR CODE
        JMP     SETERR        ;RETURN ERROR CODE FOR EMT
;
; Normal exit from USR
;
USRXIT: CALL    FREUSR        ;FREE USR MODULE
        JMP     EMTXIT        ;EXIT FROM EMT
;
; Fatal abort due to internal consistency check.
;
; Inputs:
; RO = Abort code.
;
UABORT: MOV     RO,R5         ;GET ERROR CODE FOR ABORT
        MOV     EMTADR,R4     ;GET ADDRESS OF ABORTED EMT
        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 015426  ERRNAM:
14         ;
15         ; Don't set file spec for error message if a .SERR has been done
16         ;
17 015426 105737 000000G      TSTB   SERFLG      ;Has a .SERR been done?
18 015432 001014           BNE     9$           ;Br if yes
19         ;
20         ; Move file spec to holding area
21         ;
22 015434 013737 000000G 000000G      MOV    FILSPC,ERRSPC      ;Device name
23 015442 013737 000002G 000002G      MOV    FILSPC+2,ERRSPC+2  ;File name part 1
24 015450 013737 000004G 000004G      MOV    FILSPC+4,ERRSPC+4  ;File name part 2
25 015456 013737 000006G 000006G      MOV    FILSPC+6,ERRSPC+6  ;Extension
26         ;
27         ; Finished
28         ;
29 015464 000207          9$:   RETURN
30         ;
31         ; Top of TSUSR module
32         ;
33 015466 000001          USRTOP:
34         ;
35         .END

```

Errors detected: 0

*** Assembler statistics

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

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

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

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