```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

D 3

FILEID**RMOSETDID

RMC
V04

```
RRRRRRRR    MM      MM    000000      SSSSSSSS  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD      IIIIII    DDDDDDDD
RRRRRRRR    MM      MM    000000      SSSSSSSS  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD      IIIIII    DDDDDDDD
RR      RR  MMMM  MMMM  00      00  SS          EE              TT      DD      DD      II      DD      DD
RR      RR  MMMM  MMMM  00      00  SS          EE              TT      DD      DD      II      DD      DD
RR      RR  MM  MM  MM  00    0000  SS          EE              TT      DD      DD      II      DD      DD
RR      RR  MM  MM  MM  00    0000  SS          EE              TT      DD      DD      II      DD      DD
RRRRRRRR    MM      MM  00  00  00    SSSSSS    EEEEEEEE        TT      DD      DD      II      DD      DD
RRRRRRRR    MM      MM  00  00  00    SSSSSS    EEEEEEEE        TT      DD      DD      II      DD      DD
RR  RR      MM      MM  0000    00          SS  EE              TT      DD      DD      II      DD      DD
RR  RR      MM      MM  0000    00          SS  EE              TT      DD      DD      II      DD      DD
RR    RR    MM      MM  00      00          SS  EE              TT      DD      DD      II      DD      DD      ....
RR    RR    MM      MM  00      00          SS  EE              TT      DD      DD      II      DD      DD      ....
RR      RR  MM      MM    000000    SSSSSSSS    EEEEEEEEEE      TT      DDDDDDDD      IIIIII    DDDDDDDD        ....
RR      RR  MM      MM    000000    SSSSSSSS    EEEEEEEEEE      TT      DDDDDDDD      IIIIII    DDDDDDDD        ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000            $BEGIN  RMOSETDID,000,RM$RMS0,<SET DID FROM DIRECTORY NAME>
0000
0000     ;
0000     ;****************************************************************************
0000     ;*                                                                          *
0000     ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     ;*  ALL RIGHTS RESERVED.                                                    *
0000     ;*                                                                          *
0000     ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000     ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000     ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000     ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000     ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000     ;*  TRANSFERRED.                                                            *
0000     ;*                                                                          *
0000     ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000     ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000     ;*  CORPORATION.                                                            *
0000     ;*                                                                          *
0000     ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000     ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000     ;*                                                                          *
0000     ;*                                                                          *
0000     ;****************************************************************************
0000     ;
```

```
0000  ;++
00C0  ;
0000  ; Facility: rms32
0000  ;
0000  ; Abstract:
0000  ;       this module includes various routines to
0000  ;       obtain the did of a given directory sper.
0000  ;
0000  ;
0000  ; Environment:
0000  ;               star processor running starlet exec.
0000  ;
0000  ; Author: l f laveraure,         creation date: 11-march-77
0000  ;
0000  ; Modified By:
0000  ;
0000  ;       V03-018 SRB0142         Steve Beckhardt         8-Aug-1984
0000  ;               Added some comments in rearm directory cache routine.
0000  ;
0000  ;       V03-017 CDS0001         Christian D. Saether     9-May-1984
0000  ;               Use general addressing mode to specify blockast.
0000  ;
0000  ;       V03-016 SRB0122         Steve Beckhardt         29-Apr-1994
0000  ;               Fixed bug in rearm cache code where LKSB wasn't
0000  ;               removed from stack on certain error paths.
0000  ;
0000  ;       V03-015 DGB0023         Donald G. Blair         08-Mar-1984
0000  ;               Remove global symbol MFD_FID.  Make PREFIX_0 a local
0000  ;               routine.
0000  ;
0000  ;       V03-014 SRB0111         Steve Beckhardt         9-Feb-1984
0000  ;               Added RMS directory caching support for cluster operation.
0000  ;
0000  ;       V03-013 RAS0223         Ron Schaefer            16-Dec-1983
0000  ;               Change $SCBDEF and SCB$xxx to $FSCBDEF and FSCB$xxx.
0000  ;
0000  ;       V03-012 SHZ0001         Stephen H. Zalewski     13-Sep-1983
0000  ;               No longer use RM$GETDEVNAM to get device id as this routine
0000  ;               is now obsolete.  Pull the device id from the FWA.
0000  ;
0000  ;               Add routine RM$GETCCB to this module.  It was in module
0000  ;               RMOGETDVI, but that module was deleted because is is obsolete.
0000  ;
0000  ;       V03-011 KBT0588         Keith B. Thompson       18-Aug-1983
0000  ;               Try one more time to get grpmbr directories in rooted
0000  ;               directories to work!
0000  ;
0000  ;       V03-010 KBT0561         Keith B. Thompson       21-Jul-1983
0000  ;               Ignore open by name block if search list pass
0000  ;
00U0  ;       V03-009 KBT0544         Keith B. Thompson       15-Jun-1983
0000  ;               Check for grpmbr directory in the descriptor at BLDNAM
0000  ;
0000  ;       V03-008 KBT0526         Keith B. Thompson       24-May-1983
0000  ;               Fix a bobo, don't skip the mfd if there are no rooted
0000  ;               directories
0000  ;
```

```
0000      ;       V03-007 KBT0517         Keith B. Thompson       23-May-1983
0000      ;               RMS$CHKNAMBLK no longer exist
0000      ;
0000      ;       V03-006 KBT0511         Keith B. Thompson       13-May-1983
0000      ;               Change search algorithm to use FWA$G_CDIRn to get
0000      ;               concealed directories
0000      ;
0000      ;       V03-005 KBT0455         Keith B. Thompson       7-Jan-1983
0000      ;               Directory cache is now two pages long. Also put in
0000      ;               ASSUME to check that enough nodes can be allocated
0000      ;               in it.
0000      ;
0000      ;       V03-004 JWH0151         Jeffrey W. Horn          7-Dec-1982
0000      ;               Reference Directory Cache page via a SHELL pointer
0000      ;               rather than an offset from the top of the RMS impure
0000      ;               area.
0000      ;
0000      ;       V03-003 KBT0216         Keith B. Thompson       23-Aug-1982
0000      ;               Reorganize psects
0000      ;
0000      ;       V03-002 RAS0087         Ron Schaefer    23-Apr-1982
0000      ;               Correct directory cache rebuild for rooted directory
0000      ;               $SEARCH sequences.
0000      ;
0000      ;       V03-001 RAS0086         Ron Schaefer     8-Apr-1982
0000      ;               Zero-out FIB$W_VERLIMIT after directory access to
0000      ;               correctly propagate version limits.
0000      ;
0000      ;       V02-008 RAS0068         Ron Schaefer    16-Feb-1982
0000      ;               Correct spurious error code caused be wildcard directories
0000      ;               appearing in calls to SETDID.
0000      ;
0000      ;       V02-007 RAS0040         Ron Schaefer    18-Oct-1981
0000      ;               Implement rooted directories for concealed devices.
0000      ;
0000      ;--
0000
```

```
                        0000            .SBTTL  DECLARATIONS
                        0000
                        0000    ;
                        0000    ; Macros:
                        0000    ;
                        0000
                        0000            $FIDDEF
                        0000            $IODEF
                        0000            $RMSDEF
                        0000            $SSDEF
                        0000            $CCBDEF
                        0000            $DEVDEF
                        0000            $DRCDEF
                        0000            $FABDEF
                        0000            $FIBDEF
                        0000            $FWADEF
                        0000            $IFBDEF
                        0000            $IMPDEF
                        0000            $IPLDEF
                        0000            $LCKDEF
                        0000            $NAMDEF
                        0000            $QIODEF
                        0000            $FSCBDEF
                        0000            $SSDEF
                        0000            $UCBDEF
                        0000            $VCBDEF
                        0000
                        0000    ;
                        0000    ; Equated Symbols:
                        0000    ;
                        0000
              00000020  0000            FOP     = FAB$L_FOP*8   ; bit offset to fop
                        0000
                        0000    ;
                        0000    ; Own Storage:
                        0000    ;
                        0000
                        0000    DIR_SUFFIX:
     31 3B 52 49 44 2E  0000            .ASCII  /.DIR;1/                ; constant suffix for directory files
                        0006
```

```
0006            .SBTTL  RM$SETDID, Routine to set Directory File ID
0006
0006    ;++
0006    ;
0006    ; RM$SETDID  - Set directory ID
0006    ;
0006    ; The rm$setdid routine's function is to initialize the
0006    ; directory id field of the fib by setting it to the file id
0006    ; of the (lowest level) directory file.  it accomplishes this
0006    ; by performing the following operations:
0006    ;
0006    ;       1.  assumes the fib buffer descriptor is initialized.
0006    ;       2.  utilities the file id or directory id value from the
0006    ;           user's nam block if specified and if non-zero.
0006    ;           if found, returns to caller with fib fid or did filled in.
0006    ;       3.  otherwise, constructs the directory filename
0006    ;           based on the directory spec format
0006    ;               - if [grp,mbr] prefixes from 0 to 2 zeroes
0006    ;                 to each of the grp and mbr octal values
0006    ;                 to give a 6-character file name, e.g.,
0006    ;                 [1,20] gives 001020
0006    ;               - if [directory-name] format uses the name as given
0006    ;       4.  searches the directory cache for the specified device and directory
0006    ;           entries.
0006    ;       5.  if any entry not found, a new entry is made by looking up the directory.
0006    ;           in order to do the lookup, the code appends the fixed type and version
0006    ;           of '.dir;1' to the filename and issues a qio to lookup the file id
0006    ;           in the master file directory or lower level directory.
0006    ;       6.  the returned file id is copied to the directory id field of the fib
0006    ;       7.  the file id field of the fib is zeroed.
0006    ;
0006    ; Calling sequence:
0006    ;
0006    ;       bsbw    rm$setdid
0006    ;
0006    ; Input Parameters:
0006    ;
0006    ;       r11     impure area address
0006    ;       r10     fwa address
0006    ;       r9      ifab address
0006    ;       r8      fab address
0006    ;
0006    ; Implicit Inputs:
0006    ;
0006    ;       nam$w_did       -  directory id to use else zero
0006    ;       ifb$l_chnl      -  channel # for qio
0006    ;       ifb$l_prim_dev  -  device characteristics
0006    ;       fwa$q_cdir1...  - concealed directory spec element descriptors
0006    ;       fwa$q_dir1...   - directory spec element descriptors
0006    ;       fwa$q_dir+4     -  address of scratch buffer
0006    ;       fwa$t_fibbuf    -  must be zero
0006    ;       the directory cache
0006    ;
0006    ; Output Parameters:
0006    ;
0006    ;       r0      status code
0006    ;       r1-r7,ap destroyed
```

```
                    0006      ;
                    0006      ; Implicit Outputs:
                    0006      ;
                    0006      ;       fwa$q_fib            -  descriptor initialized
                    0006      ;       fwa$t_fibbuf+fib$w_did  -  directory file id initialized
                    0006      ;       fwa$t_fibbuf+fib$w_fid  -  set from nam$w_fid
                    0006      ;       ifb$l_ios            -  set to i/o status
                    0006      ;       fab$l_stv            -  set to system error code on error
                    0006      ;       the directory cache is updated.
                    0006      ;
                    0006      ; Completion Codes:
                    0006      ;
                    0006      ;       standard rms, in particular, suc, dnf and idr.
                    0006      ;
                    0006      ; Side Effects:
                    0006      ;
                    0006      ;       may have switched to running at ast level.
                    0006      ;       all user structures except fab must be re-probed.
                    0006      ;
                    0006      ;--
                    0006
                    0006      RM$SETDID::
                    0006              $TSTPT  SETDID
                    000C
                    000C      ;
                    000C      ; check if we really need to go through this code
                    000C      ;
                    000C
       69   03   E1 000C              BBC     #DEV$V_DIR,IFB$L_PRIM_DEV(R9),- ; branch if no directory
            32      000F                      SUCCESS                         ;
  57   28   A8   D0 0010              MOVL    FAB$L_NAM(R8),R7                ; get nam block
            41   13 0014              BEQL    CHKMT                           ; branch if none
         FFE7'   30 0016              BSBW    RM$CHKNAM                       ; verify nam
       29   50   E9 0019              BLBC    R0,RETURN                       ; if not ok exit
                    001C
                    001C      ;
                    001C      ; try to get file id from nam block
                    001C      ;
                    001C
  37 6A     02   E0 001C              BBS     #FWA$V_SL_PASS,(R10),CHKMT      ; ignore if in search list oper
  33 68     38   E1 0020              BBC     #FAB$V_NAM+FOP,(R8),CHKMT       ; branch if not doing nam blk open
       24   A7   D0 0024              MOVL    NAM$W_FID(R7),-                 ; get file-id
     01F8   CA      0027                      FIB$W_FID+FWA$T_FIBBUF(R10)     ;
            08   13 002A              BEQL    10$                             ; branch if none
       28   A7   B0 002C              MOVW    NAM$W_FID_RVN(R7),-             ; copy relative vol number too
     01FC   CA      002F                      FIB$W_FID_RVN+FWA$T_FIBBUF(R10) ;
            0E   11 0032              BRB     SUCCESS                         ; all done
                    0034
                    0034      ;
                    0034      ;  try to get directory id from nam block
                    0034      ;
                    0034
       2A   A7   B0 0034      10$:    MOVW    NAM$W_DID(R7),-                 ; pick up directory id from nam blk
     01FE   CA      0037                      FIB$W_DID_NUM+FWA$T_FIBBUF(R10) ;
            1B   13 003A              BEQL    CHKMT                           ; branch if not specified
                    003C
                    003C              ASSUME  FIB$W_DID_RVN   EQ      FIB$W_DID_SEQ+2
```

```
                          003C    278
       2C A7       D0     003C    279              MOVL    NAM$W_DID_SEQ(R7),-                 ; move the rest of the did
     0200 CA              003F    280                      FIB$W_DID_SEQ+FWA$T_FIBBUF(R10)     ;
                          0042    281  SUCCESS:RMSSUC                                          ; set success
                   05     0045    282  RETURN: RSB                                             ;  and return
                          0046    283
                          0046    284  ;
                          0046    285  ;  set mfd did for magtape and exit with success
                          0046    286  ;
                          0046    287
                          0046    288  SET_MT_MFD:
                          0046    289              RMSSUC
                          0049    290
                          0049    291  ;
                          0049    292  ;  subroutine to set the mfd directory id into the fib
                          0049    293  ;
                          0049    294
                          0049    295              ASSUME  FIB$W_DID_SEQ   EQ      FIB$W_DID_NUM+2
                          0049    296
00040004 8F        D0     0049    297  SETMFD: MOVL    #<FID$C_MFD@16>+FID$C_MFD,-         ; set file id of mfd
     01FE CA              004F    298                      FIB$W_DID_NUM+FWA$T_FIBBUF(R10)     ;
     0202 CA       B4     0052    299              CLRW    FIB$W_DID_RVN+FWA$T_FIBBUF(R10)     ; from the rooted directory DID
                   05     0056    300              RSB
```

```
                          0057
                          0057  ;++
                          0057  ;
                          0057  ; directory id wasn't in nam block.  get it from directory cache.
                          0057  ;
                          0057  ;  alternate entry if nam block not to be used for input (from rms$rename)
                          0057  ;
                          0057  ;--
                          0057
                          0057  RM$SETDID_ALT::
                05   E0   0057  CHKMT:  BBS     #DEV$V_SQD,-
                69        0059                  IFB$L_PRIM_DEV(R9),-
                EB        005A                  SET_MT_MFD              ; branch if magtape
                          005B
                          005B  ;++
                          005B  ;
                          005B  ;  locate the device id in the directory cache
                          005B  ;
                          005B  ;--
                          005B
      53    40 AA    D0   005B          MOVL    FWA$Q_DIR+4(R10),R3     ; set addr of scratch buffer
                53   DD   005F          PUSHL   R3                      ; Push address of buffer onto stack.
      83  0198 CA    90   0061          MOVB    FWA$Q_SHRFIL_LCK(R10),(R3)+ ; Make first byte count of string
          0198 CA    28   0066          MOVC3   FWA$Q_SHRFIL_LCK(R10),- ; Move device id string into buffer
          019C DA         006A                  @FWA$Q_SHRFIL_LCK+4(R10),-  ;(this is the unreadable form)
                63        006D                  (R3)
                54 8ED0   006E          POPL    R4                      ; Pop address of buffer into R4.
              0281   30   0071          BSBW    RM$GETCCB               ; Get CCB address in R1.
          56    61   D0   0074          MOVL    CCB$L_UCB(R1),R6        ; Get UCB address.
      57  00AC C6    3E   0077          MOVAW   UCB$W_DIRSEQ(R6),R7     ; Save UCB dirseq address here.
55    00000000'9F    DE   007C          MOVAL   @#PIO$GL_DIRCACHE,R5    ; addr of device list head
                          0083
              01C2   30   0083          BSBW    FIND_ENTRY              ; go find this entry in cache
                10   13   0086          BEQL    10$                     ; branch if none found
      67    3A AC    B1   0088          CMPW    DRC$W_DIRSEQ(AP),(R7)   ; cache entry still valid?
                24   13   008C          BEQL    20$                     ; branch if yes
          55    5C   D0   008E          MOVL    AP,R5                   ; get device node to correct reg
      38 AA     5C   D0   0091          MOVL    AP,FWA$L_DEVNODADR(R10) ; save the device node address
              015E   31   0095          BRW     PRUNE                   ; and go prune back branch
                          0098
                          0098  ;
                          0098  ;  no entry for this device in the directory cache.  -  make one.
                          0098  ;
                          0098
              01E6   30   0098  10$:    BSBW    GET_FREE                ; go pick a free node
          65    6C   0E   009B          INSQUE  (AP),(R5)               ; insert node at list head
          50    64   9B   009E          MOVZBW  (R4),R0                 ; get length of device string
   10 AC    64   50  28   00A1          MOVC3   R0,(R4),DRC$T_NAME(AP)  ; move the device string
                          00A6          ASSUME  UCB$V_AST_ARMED  EQ  15
      3A AC     67   B0   00A6  15$:    MOVW    (R7),DRC$Q_DIRSEQ(AP)   ; save the dir seq. count
                06   19   00AA          BLSS    20$                     ; branch if cache blocking AST is armed
              0250   30   00AC          BSBW    RM$ARM_DIRCACHE         ; Arm it
             F4 50   E8   00AF          BLBS    R0,15$                  ; Repeat saving DIRSEQ if successful
      38 AA     5C   D0   00B2  20$:    MOVL    AP,FWA$L_DEVNODADR(R10) ; save the device node address
```

```
                        00B6
                        00B6      ;++
                        00B6      ;
                        00B6      ;  follow the directory cache entries for this directory spec.
                        00B6      ;  if any missing, do a lookup to supply the entry and restart scan from the top.
                        00B6      ;
                        00B6      ;--
                        00B6
                        00B6      CLR_LOOKUP:
            34 AA   D4  00B6               CLRL    FWA$L_LOOKUP(R10)       ; say no lookup done
                        00B9
                        00B9      ;
                        00B9      ; If a root directory is defined, then locate the root directory string
                        00B9      ; before starting the (sub)directory lookups.  This is necessary since the
                        00B9      ; the UFD in this case is actually an SFD of the root directory.
                        00B9      ;
                        00B9
                        00B9      FIRST_DIR:
      5C    38 AA   D0  00B9               MOVL    FWA$L_DEVNODADR(R10),AP          ; reset device node address
     56   0130 CA   7E  00BD               MOVAQ   FWA$Q_DIR1(R10),R6               ; get addr of 1st dir. discriptor
      05 6A    3A   E1  00C2               BBC     #FWA$V_ROOT_DIR,(R10),BLDNAM     ; no root present
     56   00F0 CA   7E  00C6               MOVAQ   FWA$Q_CDIR1(R10),R6              ; if a concealed directory
                        00CB                                                        ; start from there
                        00CB
                        00CB      ;
                        00CB      ; Construct directory name
                        00CB      ;
                        00CB
  53    40 AA   01  C3  00CB      BLDNAM:  SUBL3   #1,FWA$Q_DIR+4(R10),R3           ; get name scratch buffer
     1D 66      16  E0  00D0               BBS     #FSCB$V_GRPMBR,(R6),10$          ; branch if [grp,mbr] format
     0A 6A      3A  E1  00D4               BBC     #FWA$V_ROOT_DIR,(R10),5$         ; skip MFD test if not rooted
     06 66      19  E1  00D8               BBC     #FSCB$V_MFD,(R6),5$              ; branch if not MFD string
        56      08  C0  00DC               ADDL2   #8,R6                            ; skip this directory
         00B1       31  00DF               BRW     NEXT_DIR
                        00E2
                        00E2      ;
                        00E2      ; Directory name is in [name1.name2...] format construct the current
                        00E2      ; level directory name
                        00E2      ;
                        00E2
                53  DD  00E2      5$:      PUSHL   R3                               ; save buff start addr
         50     86  D0  00E4               MOVL    (R6)+,R0                         ; get name length
     83  50     01  81  00E7               ADDB3   #1,R0,(R3)+                      ; store length count in string
     63  96     50  28  00EB               MOVC3   R0,a(R6)+,(R3)                   ; move to temporary buffer
                0B  11  00EF               BRB     20$                              ; go look up the file id
                        00F1
                        00F1      ;
                        00F1      ; directory name is in [grp,mbr] format.
                        00F1      ; build the directory name from the two values, prefixing
                        00F1      ; with leading zeroes if neccessary to get a 6-character name
                        00F1      ;
                        00F1
                53  DD  00F1      10$:     PUSHL   R3                               ; save buff start addr
         83     07  90  00F3               MOVB    #7,(R3)+                         ; count of string to match
              01E4  30  00F6               BSBW    PREFIX_0                         ; move group part
              01E1  30  00F9               BSBW    PREFIX_0                         ; move member part
                        00FC
```

```
                            00FC   ;
                            00FC   ;  look up file in cache
                            00FC   ;
                            00FC
            54 8ED0         00FC   20$:     POPL    R4                              ; restore counted string addr
    55   08 AC   9E         00FF            MOVAB   DRC$L_LVLFLNK(AP),R5            ; addr of list hdr for nxt level
         0142    30         0103            BSBW    FIND_ENTRY                      ; go find this directory entry
           16    12         0106            BNEQ    NXT_DIR                         ; next_dir if found
                            0108
                            0108   ;++
                            0108   ;
                            0108   ; No entry for this (sub)directory in the cache.  We must lookup the file and
                            0108   ; make and entry.  Because a more priviliged mode could invalidate the cache
                            0108   ; while we stall, (verrrry unlikely, but possible), we must find our way back
                            0108   ; down to this level before actually adding the new entry.
                            0108   ;
                            0108   ;--
                            0108
    5C   34 AA   D0         0108   NOT_FND:MOVL     FWA$L_LOOKUP(R10),AP            ; get addr node for last lookup
           15    13         010C            BEQL    LOOKUP                          ; branch if none
                            010E
                            010E   ;
                            010E   ;  this is the 2nd time thru.  lookup has already been done.  add the looked-up
                            010E   ;  entry to the cache as long as it's still the one we want.
                            010E   ;
                            010E
         34 AA   D4         010E            CLRL    FWA$L_LOOKUP(R10)               ; indicate no lookup node
      50   64    9B         0111            MOVZBW  (R4),R0                         ; get length of string
10 AC 64   50    29         0114            CMPC3   R0,(R4),DRC$T_NAME(AP)          ; is this the right entry?
           05    12         0119            BNEQ    FREE_UP                         ; branch if not
      65   6C    0E         011B            INSQUE  (AP),(R5)                       ; insert new node after header
           73    11         011E   NXT_DIR:BRB      NEXT_DIR                        ; and continue
                            0120
         01AC    30         0120   FREE_UP:BSBW     ADD_TO_FREE                     ; return the node
                            0123
                            0123   ;
                            0123   ;  must look up the file.  use the current cache node to set the did.
                            0123   ;
                            0123
      55   08    C2         0123   LOOKUP: SUBL2    #DRC$L_LVLFLNK,R5               ; back to start of current node
   38 AA   55    D1         0126            CMPL    R5,FWA$L_DEVNODADR(R10)         ; is this the device node?
           05    12         012A            BNEQ    10$                             ; branch if not
         FF1A    30         012C            BSBW    SETMFD                          ; go set mfd did
           0C    11         012F            BRB     20$                             ; continue
         38 A5   D0         0131   10$:     MOVL    DRC$W_DID(R5),-
       01FE CA              0134                    FIB$W_DID+FWA$T_FIBBUF(R10)     ; set the did from cur. node
         3C A5   B0         0137            MOVW    DRC$W_DID+4(R5),-
       0202 CA              013A                    FIB$W_DID+4+FWA$T_FIBBUF(R10)   ;  (ditto)
                            013D
                            013D   ;
                            013D   ; append '.dir;1' to the directory name, determine
                            013D   ; the total string length, and perform qio to get the file-id
                            013D   ;
                            013D
      53   64    9A         013D   20$:     MOVZBL  (R4),R3                         ; set size of dir name string
01 A4 53   2A    3A         0140            LOCC    #^A'*',R3,1(R4)                 ; '*' in file name?
           46    12         0145            BNEQ    ERRDIR                          ; do not allow wildcards here
      01 A4    53    25   3A  0147   471          LOCC    #^A'%',R3,1(R4)                 ; '%' in file name?
                     3F   12  014C   472          BNEQ    ERRDIR                          ; do not allow wildcards here
 6443    FEAD CF     06   28  014E   473          MOVC3   #6,DIR_SUFFIX,(R4)[R3]          ; append fixed suffix
  3C AA    53    40 AA   C3  0155   474          SUBL3   FWA$Q_DIR+4(R10),R3,-           ; compute name length
                              015B   475                  FWA$Q_DIR(R10)
              0204 CA     D4  015B   476          CLRL    FWA$T_FIBBUF+FIB$L_WCC(R10)     ; wcc must be zero
                     7E   7C  015F   477          CLRQ    -(SP)                           ; p5, p6 zero
                     7E   7C  0161   478          CLRQ    -(SP)                           ; p3, p4 zero
                  3C AA   7F  0163   479          PUSHAQ  FWA$Q_DIR(R10)                  ; p2 = directory name descriptor
              50     32   9A  0166   480          MOVZBL  S^#IO$_ACCESS,R0                ; qio function code
                  FE94'   30  0169   481          BSBW    RM$FCPFNC                       ; issue the fcp function
                  68 50   E9  016C   482          BLBC    R0,ERRDNF                       ; get out on error
                              016F   483
                              016F   484 ;
                              016F   485 ; directory look up succeeded.
                              016F   486 ; move the directory file id to the new directory cache node
                              016F   487 ;
                              016F   488
                  010F    30  016F   489          BSBW    GET_FREE                        ; go pick a free node
          34 AA      5C   D0  0172   490          MOVL    AP,FWA$L_LOOKUP(R10)            ; save addr of lookup node
        01F8 CA      06   28  0176   491          MOVC3   #6,FIB$W_FID+FWA$T_FIBBUF(R10),-; save the directory fid
                  38 AC       017B   492                  DRC$W_DID(AP)
      53    40 AA    01   C3  017D   493          SUBL3   #1,FWA$Q_DIR+4(R10),R3          ; get save string addr
              50     63   9B  0182   494          MOVZBW  (R3),R0                         ; get string len
      10 AC    63    50   28  0185   495          MOVC3   R0,(R3),DRC$T_NAME(AP)          ; save string in dir node
                  FF2C    31  018A   496          BRW     FIRST_DIR                       ; branch to top to come down
                              018D   497                                                  ; tree again and find this node
                              018D   498
                              018D   499 ;
                              018D   500 ;  return error if wildcards got this far
                              018D   501 ;
                              018D   502 ERRDIR: RMSERR  DIR                             ; return error in directory
                          05  0192   503          RSB
```

```
RMO
Sym
$$.
$$A
$$R
$$R
$$R
$$R
$$T
ADD
ARM
BLD
CCB
CHK
CHK
CLR
DFV
DEV
DEV
DIR
DRC
DRC
DRC
DRC
DRC
DRC
DRC
ERR
ERR
ERR
EXI
FAB
FAB
FAB
FAB
FIB
FIB
FIB
FIB
FIB
FIB
FIB
FIB
FID
FIN
FIR
FOP
FRE
FSC
FSC
FTL
FWA
FWA
FWA
FWA
FWA
FWA
FWA
FWA
```

```
                        0193
                        0193  ;++
                        0193  ;
                        0193  ; Found this directory entry o.k.  -  see if more to find
                        0193  ;
                        0193  ; The directory descriptors are organized as follows:
                        0193  ;
                        0193  ;         FWA$Q_CDIR1     - Concealed directory descriptors
                        0193  ;                 .
                        0193  ;                 .
                        0193  ;                 .
                        0193  ;         FWA$Q_CDIR8
                        0193  ;         FWA$Q_DIR1      - Followed by the normal directory descriptors
                        0193  ;                 .
                        0193  ;                 .
                        0193  ;                 .
                        0193  ;         FWA$Q_DIR8
                        0193  ;
                        0193  ; If a zero entry is found we must check to see which group we are in.  If
                        0193  ; it is the concealed list we start with the normal directory descriptors.
                        0193  ; If there are 8 concealed directories this loop will fall right through
                        0193  ; and search the normal ones.
                        0193  ;
                        0193  ;--
                        0193
                        0193
                        0193  ;
                        0193  ; Pick up the next sub-directory name if any more to go
                        0193  ;
                        0193
                        0193  NEXT_DIR:
            66    B5    0193          TSTW    (R6)                            ; zero directory length?
            15    12    0195          BNEQ    20$                             ; branch if not
                        0197
                        0197  ;
                        0197  ; If the descriptor is zero see if we have passed the concealed directory
                        0197  ; descriptors.  If so we are done, else start on the normal directories
                        0197  ;
                        0197
       0128 CA    7F    0197          PUSHAQ  FWA$Q_CDIR8(R10)                ; get lowest level concealed
                        019B                                                  ;  directory descriptor addr
       8E   56    D1    019B          CMPL    R6,(SP)+                        ; past it already?
            15    1A    019E          BGTRU   EXIT                            ; branch if yes (all done)
  56   0130 CA    7E    01A0          MOVAQ   FWA$Q_DIR1(R10),R6              ; start one normal d'r list
            66    B5    01A5          TSTW    (R6)                            ; zero directory length?
            0C    13    01A7          BEQL    EXIT                            ; exit if so (no normal dirs)
          FF1F    31    01A9  10$:    BRW     BLDNAM
                        01AC
       0168 CA    7F    01AC  20$:    PUSHAQ  FWA$Q_DIR8(R10)                 ; get lowest level sub
                        01B0                                                  ;  directory descriptor addr
       8E   56    D1    01B0          CMPL    R6,(SP)+                        ; past it already?
            F4    1B    01B3          BLEQU   10$                             ; branch not
                        01B5
                        01B5  ;++
                        01B5  ;
                        01B5  ; Have found all needed nodes.  Check if directory sequence count still valid.
                        01B5  ;
```

```
                           01B5  ;--
                           01B5
      56   5C      D0      01B5  EXIT:    MOVL    AP,R6                            ; save addr of dir node
  5C   34  AA      D0      01B8           MOVL    FWA$L_LOOKUP(R10),AP             ; unused lookup node?
           03      13      01BC           BEQL    10$                              ; branch if not
         010E      30      01BE           BSBW    ADD_TO_FREE                      ; return it to the free list
           7C      10      01C1  10$:     BSBB    CHKDIRSEQ                        ; cache still valid?
           31      12      01C3           BNEQ    PRUNE                            ; branch if not
                           01C5
                           01C5  ;++
                           01C5  ;
                           01C5  ;  All set.  Just set the did in the fib and clear the fid and version limit.
                           01C5  ;
                           01C5  ;--
                           01C5
  38 A6    06      28      01C5           MOVC3   #6,DRC$W_DID(R6),-               ; set the directory id
     01FE  CA              01C9                   FIB$W_DID+FWA$T_FIBBUF(R10)
     01F8  CA      D4      01CC           CLRL    FIB$W_FID+FWA$T_FIBBUF(R10)      ; zero the file id
     0220  CA      B4      01D0           CLRW    FIB$W_VERLIMIT+FWA$T_FIBBUF(R10); zero the version limit
           50      D6      01D4           INCL    R0                               ; show success (r0 = 0 from movc3)
                   05      01D6           RSB                                      ; back to caller of rm$setdid
                           01D7
```

```
                    01D7
                    01D7  ;++
                    01D7  ;
                    01D7  ; Handle directory not found error.
                    01D7  ;
                    01D7  ;--
                    01D7
                    01D7  ERRDNF:
          66   10   01D7          BSBB    CHKDIRSEQ               ; error due to invalid cache?
          02   13   01D9          BEQL    5$                      ; branch if not
          19   11   01DB          BRB     PRUNE                   ; possibly - go try again
0910 8F   50   B1   01DD  5$:     CMPW    R0,#SS$_NOSUCHFILE      ; was error file not found?
          0A   12   01E2          BNEQ    10$                     ; branch if not
  0C A8   50   D0   01E4          MOVL    R0,FAB$L_STV(R8)        ; save system code
                    01E8          RMSERR  DNF                     ; replace with directory not found
               05   01ED          RSB                             ; and return
                    01EE  10$:    RMSERR  DNF,R1                  ; default error to directory not found
       FE0A'   31   01F3          BRW     RM$MAPERR               ; map error to rms & return
                    01F6
```

```
                      01F6
                      01F6      ;++
                      01F6      ;
                      01F6      ;  have run into an invalid cache condition, i.e., something was done
                      01F6      ;  by the acp (e.g., mount) to invalidate the cache contents.
                      01F6      ;  remove all entries below the device, reset dirseq, and try again.
                      01F6      ;
                      01F6      ;--
                      01F6
                      01F6      PRUNE:
      55    08   C0   01F6              ADDL2   #DRC$L_LVLFLNK,R5          ; get address of ufd header
      54    55   D0   01F9              MOVL    R5,R4                      ; set stop address
      55    65   D1   01FC              CMPL    (R5),R5                    ; anything to prune?
            26   13   01FF              BEQL    30$                        ; branch if not
            55   DD   0201      10$:    PUSHL   R5                         ; save header addr
      5C    65   D0   0203              MOVL    (R5),AP                    ; get next level down
  55   08 AC     DE   0206      15$:    MOVAL   DRC$L_LVLFLNK(AP),R5       ; get addr of level link
      55    65   D1   020A              CMPL    (R5),R5                    ; another level?
            F2   12   020D              BNEQ    10$                        ; branch if yes
                      020F
                      020F      ;
                      020F      ;  at lowest level  -  remove this node and move to side node
                      020F      ;
                      020F
            6C   DD   020F      20$:    PUSHL   (AP)                       ; save next node addr
      5C    6C   0F   0211              REMQUE  (AP),AP                    ; remove node
          00B8   30   0214              BSBW    ADD_TO_FREE                ; add it to the free list
      5C    8E   D0   0217              MOVL    (SP)+,AP                   ; get next node addr
      6E    5C   D1   021A              CMPL    AP,(SP)                    ; back to previous level?
            E7   12   021D              BNEQ    15$                        ; branch if not
      5C    08   C2   021F              SUBL2   #DRC$L_LVLFLNK,AP          ; get node start address
      54    8E   D1   0222              CMPL    (SP)+,R4                   ; back to dev node?
            E8   12   0225              BNEQ    20$                        ; branch if not
                      0227
                      0227      ;
                      0227      ;  store new dirseq value and rebuild tree for this device
                      0227      ;
                      0227
                      0227              ASSUME  UCB$V_AST_ARMED  EQ  15
  32 A4     67   B0   0227      30$:    MOVW    (R7),DRC$Q_DIRSEQ-DRC$L_LVLFLNK(R4)
            06   19   022B              BLSS    40$                        ; branch if cache blocking AST is armed
          00CF   30   022D              BSBW    RM$ARM_DIRCACHE            ; Arm it
       F4 50     E8   0230              BLBS    R0,30$                     ; Repeat saving DIRSEQ if successful
          FE80   31   0233      40$:    BRW     CLR_LOOKUP
```

```
                                  0236
                                  0236   ;++
                                  0236   ;
                                  0236   ;  handle bad directory rename error.
                                  0236   ;
                                  0236   ;--
                                  0236
                         50 8ED0  0236   ERRIDR: POPL    R0                          ; discard local ret addr
                                  0239           RMSERR  IDR                         ; set bad directory rename
                              05  023E           RSB                                 ; and return
                                  023F
                                  023F   ;++
                                  023F   ;
                                  023F   ;  chkdirseq subroutine to verify cache validity
                                  023F   ;
                                  023F   ;  inputs:
                                  023F   ;               r10     fwa address
                                  023F   ;               r7      ucb$w_dirseq address
                                  023F   ;               fwa$l_devnodadr
                                  023F   ;
                                  023F   ;  outputs:
                                  023F   ;               r5      fwa$l_devnodadr
                                  023F   ;               z-bit   set if cache valid, else clear
                                  023F   ;--
                                  023F
                                  023F   CHKDIRSEQ:
               55    38 AA    D0  023F           MOVL    FWA$L_DEVNODADR(R10),R5 ; get device node address
               67    3A A5    B1  0243           CMPW    DRC$W_DIRSEQ(R5),(R7)   ; still valid?
                              05  0247           RSB
                                  0248
```

```
                    0248
                    0248   ;++
                    0248   ;
                    0248   ;  find_entry subroutine to find an entry in the directory cache
                    0248   ;
                    0248   ;  inputs:
                    0248   ;                r4      address of counted string to match
                    0248   ;                r5      address of list head for level to scan
                    0248   ;
                    0248   ;  outputs:
                    0248   ;                z-bit   set if no match found, else clear
                    0248   ;                ap      address of matching entry
                    0248   ;                r0-r3   destroyed
                    0248   ;
                    0248   ;  note: if match found, matching entry is requeued to immediately follow list head.
                    0248   ;
                    0248   ;--
                    0248
                    0248   FIND_ENTRY:
         5C   55  D0 0248          MOVL    R5,AP                      ; set up to find 1st node
         5C   6C  D0 024B  10$:    MOVL    DRC$L_NXTFLNK(AP),AP       ; get next node
         55   5C  D1 024E          CMPL    AP,R5                      ; back at head?
              2D  13 0251          BEQL    20$                        ; branch if yes (no match)
         50   64  9B 0253          MOVZBW  (R4),R0                    ; get len of string to match
   10 AC 64   50  29 0256          CMPC3   R0,(R4),DRC$T_NAME(AP)     ; do they match?
              EE  12 025B          BNEQ    10$                        ; branch if not
                     025D
                     025D  ;
                     025D  ;  matching entry found  -  requeue entry to head of the list
                     025D  ;
                     025D
         5C   6C  0F 025D          REMQUE  (AP),AP
         65   6C  0E 0260          INSQUE  (AP),(R5)
                     0263
                     0263  ;
                     0263  ; check if the saved fid is the same as this directory id.
                     0263  ; true iff both fid and did are valid file identification fields.
                     0263  ;
                     0263
       0240 CA    A9 0263          BISW3   FWA$T_RNM_FID(R10),-       ; is this a valid fid?
    50 0244 CA       0267                  FWA$T_RNM_FID+4(R10),R0  ; not if 1st and 3rd words are 0
              11  13 026B          BEQL    15$
         38 AC    A9 026D          BISW3   DRC$W_DID(AP),-
         3C AC       0270                  DRC$W_DID+4(AP),-
            50       0272                  R0                         ; likewise for did
            09    13 0273          BEQL    15$
    0240 CA 06    29 0275          CMPC3   #6,FWA$T_RNM_FID(R10),-  ; is directory same as file?
         38 AC       027A                  DRC$W_DID(AP)
            B8    13 027C          BEQL    ERRIDR                     ; bad operation if so
            5C    D5 027E  15$:    TSTL    AP                         ; clear z-bit
                  05 0280  20$:    RSB
```

```
                              0281
                              0281   ;++
                              0281   ;
                              0281   ;  get_free subroutine to find a free node.
                              0281   ;  picks node from free list, if any, else picks least recently used dir entry
                              0281   ;  on least recently used device.
                              0281   ;
                              0281   ;  inputs:
                              0281   ;                  none
                              0281   ;
                              0281   ;  outputs:
                              0281   ;                  ap      addr of node
                              0281   ;                  r0      destroyed
                              0281   ;--
                              0281
                              0281   GET_FREE:
 50    00000000'9F     DE     0281           MOVAL   @#PIO$GL_DIRCFRLH,R0        ; get free list addr
            5C   60    D0     0288           MOVL    (R0),AP                     ; pick first node
                 04    13     028B           BEQL    10$                         ; branch if none
            60   6C    D0     028D           MOVL    (AP),(R0)                   ; bring up next free node
                       05     0290           RSB
                              0291
                              0291   ;
                              0291   ;  nothing on free list.  check that it has been initialized.
                              0291   ;
                              0291
         5C    04'A0   D0     0291   10$:    MOVL    B^PIO$GL_DIRCACHE+4-PIO$GL_DIRCFRLH(R0),AP
                              0295
                              0295   ;
                              0295   ; get lru device node
                              0295   ;
                              0295
            6C   5C    D1     0295           CMPL    AP,(AP)                     ; empty list?
                 28    12     0298           BNEQ    30$                         ; branch if not
                              029A
                              029A   ;
                              029A   ;  initialize free directory nodes list
                              029A   ;
                              029A
            5C   50    D0     029A           MOVL    R0,AP                       ; set up to init free node list head
      50    0000'C0    DE     029D           MOVAL   PIO$A_DIRCACHE-PIO$GL_DIRCFRLH(R0),R0
                              02A2
                              02A2   ;
                              02A2   ; get addr of directory cache page
                              02A2   ;
                              02A2   ; NOTE: There must be enough room in the directory cache to have a node
                              02A2   ;       for each possible subdirectoy plus the root and the device node.
                              02A2   ;
                              02A2
                              02A2           ASSUME  <<2*512>/DRC$C_BLN>     GE      FWA$C_MAXSUBDIR+1+1
                              02A2
                 10    DD     02A2           PUSHL   #<<2*512>/DRC$C_BLN>        ; set # of nodes in cache (2 pages)
            6C   50    D0     02A4   15$:    MOVL    R0,(AP)                     ; set flink of previous node
            5C   50    D0     02A7           MOVL    R0,AP                       ; save addr this flink for next node
                              02AA
                              02AA           ASSUME  DRC$L_LVLFLNK   EQ      8
                              02AA
```

```
                      80    7C  02AA    790             CLRQ    (R0)+                         ; move to level list head
                60    50    D0  02AC    791             MOVL    R0,(R0)                       ; init list to empty (flink)
             04 A0    50    D0  02AF    792             MOVL    R0,4(R0)                      ;       ''             (blink)
                50    36    C0  02B3    793             ADDL2   #DRC$C_BLN-DRC$L_LVLFLNK,R0; move to next node
                   EB 6E    F5  02B6    794             SOBGTR  (SP),15$                      ; loop if more
                      50  8ED0  02B9    795             POPL    R0                            ; clean stack
                      C3    11  02BC    796             BRB     GET_FREE                      ; and try again
                                02BE    797
                                02BE    798 ;
                                02BE    799 ;  pick relatively little used node
                                02BE    800 ;
                                02BE    801
             5C    0C AC    D0  02BE    802 20$:        MOVL    DRC$L_LVLBLNK(AP),AP          ; get lru (sub)directory
             50    08 AC    DE  02C2    803 30$:        MOVAL   DRC$L_LVLFLNK(AP),R0          ; get next level list head
                50    60    D1  02C6    804             CMPL    (R0),R0                       ; list empty?
                      F3    12  02C9    805             BNEQ    20$                           ; branch if not
                5C    6C    0F  02CB    806             REMQUE  (AP),AP                       ; pick the node
                            05  02CE    807             RSB
                                02CF    808
                                02CF    809 ;++
                                02CF    810 ;
                                02CF    811 ;  add_to_free subroutine to return a node to the free list.
                                02CF    812 ;
                                02CF    813 ;  inputs:
                                02CF    814 ;                   ap      node address
                                02CF    815 ;
                                02CF    816 ;  outputs:
                                02CF    817 ;                   r0      destroyed
                                02CF    818 ;--
                                02CF    819
                                02CF    820 ADD_TO_FREE:
     50     00000000'9F     DE  02CF    821             MOVAL   @#PIO$GL_DIRCFRLH,R0          ; get free list head addr
                6C    60    D0  02D6    822             MOVL    (R0),(AP)                     ; flink to new node
                60    5C    D0  02D9    823             MOVL    AP,(R0)                       ; new node addr to list head
                            05  02DC    824             RSB
```

```
                  02DD          .SBTTL  PREFIX_0, Convert Group-Member Format to Normal Directory
                  02DD
                  02DD  ;++
                  02DD  ;
                  02DD  ; subroutine prefix_0 to move either the group or
                  02DD  ; member part of a directory spec prefixing it
                  02DD  ; with one or two zeros to give 3 characters total
                  02DD  ;
                  02DD  ; inputs:
                  02DD  ;
                  02DD  ;       r6      addr of descriptor for group of member part
                  02DD  ;       r3      addr of output buffer
                  02DD  ;
                  02DD  ; outputs:
                  02DD  ;
                  02DD  ;       r6      r6+8
                  02DD  ;       r3      r3+3
                  02DD  ;       r0,r1,r2,r4,r5  destroyed
                  02DD  ;
                  02DD  ;--
                  02DD
                  02DD  PREFIX_0:
      50  86  D0  02DD          MOVL    (R6)+,R0                ; get length
      03  50  B1  02E0          CMPW    R0,#3                   ; all 3 chars present?
          0B  13  02E3          BEQL    20$                     ; branch if yes
      02  50  B1  02E5          CMPW    R0,#2                   ; 2 of the 3?
          03  13  02E8          BEQL    10$                     ; branch if yes
      83  30  90  02EA          MOVB    #^A/0/,(R3)+            ; move a zero
      83  30  90  02ED  10$:    MOVB    #^A/0/,(R3)+            ; move a zero
  63  96  50  28  02F0  20$:    MOVC3   R0,@(R6)+,(R3)          ; move the grp or mbr number
              05  02F4          RSB
                  02F5
```

```
                                 02F5          .SBTTL  RM$GETCCB, GET CCB ADDRESS
                                 02F5    ;++
                                 02F5    ;
                                 02F5    ; RM$GETCCB --  subroutine to return the CCB address.
                                 02F5    ;
                                 02F5    ;  INPUTS:
                                 02F5    ;
                                 02F5    ;       R9      IFAB address with channel in IFB$W_CHNL
                                 02F5    ;
                                 02F5    ;  OUTPUTS:
                                 02F5    ;
                                 02F5    ;       R1      address of CCB
                                 02F5    ;       R0,R2,R3 destroyed
                                 02F5    ;
                                 02F5    ;--
                                 02F5
                                 02F5    RM$GETCCB::
   50    20 A9          3C       02F5          MOVZWL  IFB$W_CHNL(R9),R0       ; set channel #
     00000000'9F        17       02F9          JMP     @#IOC$VERIFYCHAN        ; get the ccb address (in r1)
                                 02FF
                                 02FF    ;
                                 02FF    ; and return
                                 02FF    ;
```

```
                        02FF   ;++
                        02FF   ;
                        02FF   ;  rm$arm_dircache routine.  Converts the volume lock to rearm
                        02FF   ;  the blocking AST which in turn invalidates the cache.
                        02FF   ;
                        02FF   ;  inputs:
                        02FF   ;                 r7      address of UCB$W_DIRSEQ cell in UCB
                        02FF   ;
                        02FF   ;  outputs:
                        02FF   ;
                        02FF   ;                 r0      low bit clear = failure (blocking ast could
                        02FF   ;                         not be rearmed).  Just save current seq. #.
                        02FF   ;
                        02FF   ;                         low bit set = success.  Repeat saving of the
                        02FF   ;                         seq. #.
                        02FF   ;
                        02FF   ;--
                        02FF
                        02FF   RM$ARM_DIRCACHE::
                        02FF   ;
                        02FF   ; Duplicate some checks we will make in kernel mode only so that
                        02FF   ; if there is no lock (ODS-1, for example) we save the $CMKRNL.
                        02FF   ;
          50      D4    02FF           CLRL    R0                           ; assume failure
51   FF54 C7      3E    0301           MOVAW   -UCB$W_DIRSEQ(R7),R1         ; get address of UCB
          13      E1    0306           BBC     #DEV$V_MNT,-                 ; Return failure if device
     21 38 A1           0308                   UCB$L_DEVCHAR(R1),50$        ; is not mounted
50     34 A1      D0    030B           MOVL    UCB$L_VCB(R1),R0             ; Or if VCB isn't attached
          1B      13    030F           BEQL    50$
       7C A0      D5    0311           TSTL    VCB$L_VOLLKID(R0)            ; Or if there is no volume lock
          16      13    0314           BEQL    50$                          ; (ODS-1, for example)
          51      DD    0316           PUSHL   R1                           ; push address of UCB
          01      DD    0318           PUSHL   #1                           ; push argument count
     51   5E      D0    031A           MOVL    SP,R1                        ; r1 points to argument list
                        031D           $CMKRNL_S   B^ARM_CACHE,(R1)         ; call kernel mode routine
     5E   08      C0    0329           ADDL    #8,SP                        ; clean argument list off stack
          05            032C   50$:    RSB
                        032D
                        032D   ARM_CACHE:
        003C            032D           .WORD   ^M<R2,R3,R4,R5>
55     04 AC      D0    032F           MOVL    4(AP),R5                     ; Get UCB address
                        0333   10$:    SETIPL  IPL_DES                      ; raise IPL and lock pages
          13      E1    033A   20$:    BBC     #DEV$V_MNT,-                 ; Return failure if device
     6B 38 A5           033C                   UCB$L_DEVCHAR(R5),50$        ; is not mounted
53     34 A5      D0    033F           MOVL    UCB$L_VCB(R5),R3             ; Or if VCB isn't attached
          65      13    0343           BEQL    50$
54     7C A3      D0    0345           MOVL    VCB$L_VOLLKID(R3),R4         ; Or if there is no volume lock
          5F      13    0349           BEQL    50$                          ; (ODS-1, for example)
52   00AC C5      3C    034B           MOVZWL  UCB$W_DIRSEQ(R5),R2          ; Save old seq. #
                        0350           SETIPL  #0                           ; lower IPL
                        0353
                        0353   ;
                        0353   ;  Set up for $ENQ service to convert lock to rearm blocking AST.  We must
                        0353   ;  handle possible SS$_IVLOCKID errors due to lock manager's handling
                        0353   ;  of conversions mastered on other systems.  If two users are in this
                        0353   ;  path simultaneously then the first will do the conversion and the
                        0353   ;  second may get SS$_IVLOCKID if the conversion is in progress and
```

```
                         0353   940 ;   mastered on another system.  The solution is to retry.  We can also
                         0353   941 ;   get SS$_IVLOCKID if the volume is dismounted and the lock is dequeued
                         0353   942 ;   while we are in here.
                         0353   943 ;
                         0353   944
               54    DD  0353   945         PUSHL   R4                          ; Push lockid to create lock status
               00    DD  0355   946         PUSHL   #0                          ; block on the stack
          54   5E    D0  0357   947         MOVL    SP,R4                       ; R4 points to LKSB
                         035A   948
                         035A   949         $ENQ_S          EFN = #IMP$C_ASYQIOEFN,-
                         035A   950                         LKMODE = #LCK$K_CRMODE,-
                         035A   951                         LKSB = (R4),-
                         035A   952                         FLAGS = #<LCK$M_CONVERT!LCK$M_CVTSYS!LCK$M_SYNCSTS>,-
                         035A   953                         BLKAST = G^RM$DIRCACHE_BLKAST,-
                         035A   954                         ASTPRM = R5
          5E   08    C0  037D   955         ADDL    #8,SP                       ; Clean LKSB off stack
     2124 8F   50    B1  0380   956         CMPW    R0,#SS$_IVLOCKID            ; Can occur due to race with volume
               AC    13  0385   957         BEQL    10$                         ; dismount or due to two users
                         0387   958                                             ; in this path simultaneously.
     0689 8F   50    B1  0387   959         CMPW    R0,#SS$_SYNCH               ; Should be performed synchronously
               20    12  038C   960         BNEQ    90$                         ; Error!
                         038E   961
                         038E   962 ;
                         038E   963 ;   Check that seq. # hasn't changed before setting armed flag.
                         038E   964 ;   This must be done at IPL$_SYNCH to avoid the race condition
                         038E   965 ;   of the blocking ast being delivered between the check and the
                         038E   966 ;   setting of the armed flag.
                         038E   967 ;
                         038E   968
                         038E   969         SETIPL  IPL_DEST                    ; Raise IPL and lock pages
     00AC C5   52    B1  0395   970         CMPW    R2,UCB$W_DIRSEQ(R5)         ; Verify seq. # hasn't changed
               9E    12  039A   971         BNEQ    20$                         ; It has - repeat
          8000 8F    A8  039C   972         BISW    #UCB$M_AST_ARMED,-          ; Set the armed flag
          00AC C5        03A0   973                 UCB$W_DIRSEQ(R5)
          50   01    D0  03A3   974         MOVL    #1,R0                       ; Return success
                         03A6   975 40$:    SETIPL  #0                          ; Lower IPL
                     04  03A9   976         RET
                         03AA   977
               50    D4  03AA   978 50$:    CLRL    R0                          ; Return failure
               F8    11  03AC   979         BRB     40$
                         03AE   980
                         03AE   981
                         03AE   982 90$:    RMSPBUG FTL$_ENQDEQFAIL
                         03B5   983
                         03B5   984 IPL_DEST:
               00000008  03B5   985         .LONG   IPL$_SYNCH
                         03B9   986         ASSUME  .-ARM_CACHE  LE  513    ; Make sure we fit on two pages
                         03B9   987
                         03B9   988
                         03B9   989
                         03B9   990         .END
```

```
$$.PSECT_EP                   = 00000000                  FWA$Q_SHRFIL_LCK              = 00000198
$$ARGS                        = 0000000C                  FWA$T_FIBBUF                  = 000001F4
$$RMSTEST                     = 0000001A                  FWA$T_RNM_FID                 = 00000240
$$RMS_PBUGCHK                 = 00000010                  FWA$V_ROOT_DIR                = 0000003A
$$RMS_TBUGCHK                 = 00000008                  FWA$V_SL_PASS                 = 00000002
$$RMS_UMODE                   = 00000004                  GET_FREE                        00000281 R       01
$$T1                          = 00000000                  IFB$L_PRIM_DEV                = 00000000
ADD_TO_FREE                     000002CF R       01       IFB$W_CHNL                    = 00000020
ARM_CACHE                       0000032D R       01       IMP$C_ASYQIOEFN               = 0000001F
BLDNAM                          000000CB R       01       IO$_ACCESS                    = 00000032
CCB$L_UCB                     = 00000000                  IOC$VERIFYCHAN                  ********    X    01
CHKDIRSEQ                       0000023F R       01       IPL$_SYNCH                    = 00000008
CHKMT                           00000057 R       01       IPL_DEST                        000003B5 R       01
CLR_LOOKUP                      000000B6 R       01       LCK$K_CRMODE                  = 00000001
DEV$V_DIR                     = 00000003                  LCK$M_CONVERT                 = 00000002
DEV$V_MNT                     = 00000013                  LCK$M_CVTSYS                  = 00000040
DEV$V_SQD                     = 00000005                  LCK$M_SYNCSTS                 = 00000008
DIR_SUFFIX                      00000000 R       01       LOOKUP                          00000123 R       01
DRC$C_BLN                     = 0000003E                  NAM$W_DID                     = 0000002A
DRC$L_LVLBLNK                 = 0000000C                  NAM$W_DID_SEQ                 = 0000002C
DRC$L_LVLFLNK                 = 00000008                  NAM$W_FID                     = 00000024
DRC$L_NXTFLNK                 = 00000000                  NAM$W_FID_RVN                 = 00000028
DRC$T_NAME                    = 00000010                  NEXT_DIR                        00000193 R       01
DRC$W_DID                     = 00000038                  NOT_FND                         00000108 R       01
DRC$W_DIRSEQ                  = 0000003A                  NXT_DIR                         0000011E R       01
ERRDIR                          0000018D R       01       PIO$A_DIRCACHE                  ********    X    01
ERRDNF                          000001D7 R       01       PIO$A_TRACE                     ********    X    01
ERRIDR                          00000236 R       01       PIO$GL_DIRCACHE                 ********    X    01
EXIT                            000001B5 R       01       PIO$GL_DIRCFRLH                 ********    X    01
FAB$L_FOP                     = 00000004                  PR$_IPL                         ********    X    01
FAB$L_NAM                     = 00000028                  PREFIX_0                        000002DD R       01
FAB$L_STV                     = 0000000C                  PRUNE                           000001F6 R       01
FAB$V_NAM                     = 00000018                  QIO$_ASTADR                   = 00000014
FIB$L_WCC                     = 00000010                  QIO$_ASTPRM                   = 00000018
FIB$W_DID                     = 0000000A                  QIO$_CHAN                     = 00000008
FIB$W_DID_NUM                 = 0000000A                  QIO$_EFN                      = 00000004
FIB$W_DID_RVN                 = 0000000E                  QIO$_FUNC                     = 0000000C
FIB$W_DID_SEQ                 = 0000000C                  QIO$_IOSB                     = 00000010
FIB$W_FID                     = 00000004                  QIO$_NARGS                    = 0000000C
FIB$W_FID_RVN                 = 00000008                  QIO$_P1                       = 0000001C
FIB$W_VERLIMIT                = 0000002C                  QIO$_P2                       = 00000020
FID$C_MFD                     = 00000004                  QIO$_P3                       = 00000024
FIND_ENTRY                      00000248 R       01       QIO$_P4                       = 00000028
FIRST_DIR                       000000B9 R       01       QIO$_P5                       = 0000002C
FOP                           = 00000020                  QIO$_P6                       = 00000030
FREE_UP                         00000120 R       01       RETURN                          00000045 R       01
FSCB$V_GRPMBR                 = 00000016                  RMS$ARM_DIRCACHE                000002FF RG      01
FSCB$V_MFD                    = 00000019                  RMS$BUG                         ********    X    01
FTL$_ENQDEQFAIL               = FFFFFFF2                  RMS$CHKNAM                      ********    X    01
FWA$C_MAXSUBDIR               = 00000007                  RMS$DIRCACHE_BLKAST             ********    X    01
FWA$L_DEVNODADR               = 00000038                  RMS$FCPFNC                      ********    X    01
FWA$L_LOOKUP                  = 00000034                  RMS$GETCCB                      000002F5 RG      01
FWA$Q_CDIR1                   = 000000F0                  RMS$MAPERR                      ********    X    01
FWA$Q_CDIR8                   = 00000128                  RMS$SETDID                      00000006 RG      01
FWA$Q_DIR                     = 0000003C                  RMS$SETDID_ALT                  00000057 RG      01
FWA$Q_DIR1                    = 00000130                  RMS$_DIR                      = 000184CC
FWA$Q_DIR8                    = 00000168                  RMS$_DNF                      = 0001C04A
```

```
RMS$_IDR                        = 000182F2
SETMFD                            00000049 R      01
SET_MT_MFD                        00000046 R      01
SS$_IVLOCKID                    = 00002124
SS$_NOSUCHFILE                  = 00000910
SS$_SYNCH                       = 00000689
SUCCESS                           00000042 R      01
SYS$CMKRNL                        ********  GX    01
SYS$ENQ                           ********  GX    01
TPT$L_SETDID                      ********   X    01
UCB$L_DEVCHAR                   = 00000038
UCB$L_VCB                       = 00000034
UCB$M_AST_ARMED                 = 00008000
UCB$V_AST_ARMED                 = 0000000F
UCB$W_DIRSEQ                    = 000000AC
VCB$L_VOLLKID                   = 0000007C

                                                  +-----------------+
                                                  ! Psect synopsis  !
                                                  +-----------------+

PSECT name                        Allocation          PSECT No.  Attributes
----------                        ----------          ---------  ----------
.  ABS  .                         00000000  (    0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RMS$RMS0                          00000389  (  953.)  01 (  1.)    PIC    USR    CON    REL    GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                             00000000  (    0.)  02 (  2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                              +------------------------+
                                              ! Performance indicators !
                                              +------------------------+

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                     35    00:00:00.04    00:00:01.00
Command processing                126    00:00:00.75    00:00:04.05
Pass 1                            627    00:00:26.79    00:01:10.08
Symbol table sort                   0    00:00:04.42    00:00:06.94
Pass 2                            176    00:00:05.09    00:00:10.62
Symbol table output                16    00:00:00.17    00:00:00.88
Psect synopsis output               2    00:00:00.02    00:00:00.04
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              984    00:00:37.29    00:01:33.61

The working set limit was 2100 pages.
152575 bytes (298 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2949 non-local and 34 local symbols.
990 source lines were read in Pass 1, producing 15 object records in Pass 2.
44 pages of virtual memory were used to define 43 macros.
```

```
                                             +--------------------------+
                                             ! Macro library statistics !
                                             +--------------------------+

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                         15
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                          6
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      18
TOTALS (all libraries)                                 39

3174 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RMOSETDID/OBJ=OBJ$:RMOSETDID MSRC$:RMOSETDID/UPDATE=(ENH$:RMOSETDID)+EXECML$/LIB+LIB$:RMS/LIB
```

0320 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY