```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
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
RRRRRRRRRRR      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  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

```
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     1          $BEGIN  RMOSETDID,000,RM$RMS0,<SET DID FROM DIRECTORY NAME>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
```

```
0000     28 ;++
0000     29 ;
0000     30 ; Facility: rms32
0000     31 ;
0000     32 ; Abstract:
0000     33 ;       this module includes various routines to
0000     34 ;       obtain the did of a given directory spec.
0000     35 ;
0000     36 ;
0000     37 ; Environment:
0000     38 ;               star processor running starlet exec.
0000     39 ;
0000     40 ; Author: l f laverdure,         creation date: 11-march-77
0000     41 ;
0000     42 ; Modified By:
0000     43 ;
0000     44 ;       V03-018 SRB0142         Steve Beckhardt         8-Aug-1984
0000     45 ;               Added some comments in rearm directory cache routine.
0000     46 ;
0000     47 ;       V03-017 CDS0001         Christian D. Saether     9-May-1984
0000     48 ;               Use general addressing mode to specify blockast.
0000     49 ;
0000     50 ;       V03-016 SRB0122         Steve Beckhardt         29-Apr-1994
0000     51 ;               Fixed bug in rearm cache code where LKSB wasn't
0000     52 ;               removed from stack on certain error paths.
0000     53 ;
0000     54 ;       V03-015 DGB0023         Donald G. Blair         08-Mar-1984
0000     55 ;               Remove global symbol MFD_FID.  Make PREFIX_0 a local
0000     56 ;               routine.
0000     57 ;
0000     58 ;       V03-014 SRB0111         Steve Beckhardt         9-Feb-1984
0000     59 ;               Added RMS directory caching support for cluster operation.
0000     60 ;
0000     61 ;       V03-013 RAS0223         Ron Schaefer            16-Dec-1983
0000     62 ;               Change $SCBDEF and SCB$xxx to $FSCBDEF and FSCB$xxx.
0000     63 ;
0000     64 ;       V03-012 SHZ0001         Stephen H. Zalewski     13-Sep-1983
0000     65 ;               No longer use RM$GETDEVNAM to get device id as this routine
0000     66 ;               is now obsolete.  Pull the device id from the FWA.
0000     67 ;
0000     68 ;               Add routine RM$GETCCB to this module.  It was in module
0000     69 ;               RMOGETDVI, but that module was deleted because is is obsolete.
0000     70 ;
0000     71 ;       V03-011 KBT0588         Keith B. Thompson       18-Aug-1983
0000     72 ;               Try one more time to get grpmbr directories in rooted
0000     73 ;               directories to work!
0000     74 ;
0000     75 ;       V03-010 KBT0561         Keith B. Thompson       21-Jul-1983
0000     76 ;               Ignore open by name block if search list pass
0000     77 ;
0000     78 ;       V03-009 KBT0544         Keith B. Thompson       15-Jun-1983
0000     79 ;               Check for grpmbr directory in the descriptor at BLDNAM
0000     80 ;
0000     81 ;       V03-008 KBT0526         Keith B. Thompson       24-May-1983
0000     82 ;               Fix a bobo, don't skip the mfd if there are no rooted
0000     83 ;               directories
0000     84 ;
```

```
0000     85 ;       V03-007 KBT0517         Keith B. Thompson       23-May-1983
0000     86 ;               RM$CHKNAMBLK no longer exist
0000     87 ;
0000     88 ;       V03-006 KBT0511         Keith B. Thompson       13-May-1983
0000     89 ;               Change search algorithm to use FWA$G_CDIRn to get
0000     90 ;               concealed directories
0000     91 ;
0000     92 ;       V03-005 KBT0455         Keith B. Thompson       7-Jan-1983
0000     93 ;               Directory cache is now two pages long. Also put in
0000     94 ;               ASSUME to check that enough nodes can be allocated
0000     95 ;               in it.
0000     96 ;
0000     97 ;       V03-004 JWH0151         Jeffrey W. Horn         7-Dec-1982
0000     98 ;               Reference Directory Cache page via a SHELL pointer
0000     99 ;               rather than an offset from the top of the RMS impure
0000    100 ;               area.
0000    101 ;
0000    102 ;       V03-003 KBT0216         Keith B. Thompson       23-Aug-1982
0000    103 ;               Reorganize psects
0000    104 ;
0000    105 ;       V03-002 RAS0087         Ron Schaefer    23-Apr-1982
0000    106 ;               Correct directory cache rebuild for rooted directory
0000    107 ;               $SEARCH sequences.
0000    108 ;
0000    109 ;       V03-001 RAS0086         Ron Schaefer    8-Apr-1982
0000    110 ;               Zero-out FIB$W_VERLIMIT after directory access to
0000    111 ;               correctly propagate version limits.
0000    112 ;
0000    113 ;       V02-008 RAS0068         Ron Schaefer    16-Feb-1982
0000    114 ;               Correct spurious error code caused be wildcard directories
0000    115 ;               appearing in calls to SETDID.
0000    116 ;
0000    117 ;       V02-007 RAS0040         Ron Schaefer    18-Oct-1981
0000    118 ;               Implement rooted directories for concealed devices.
0000    119 ;
0000    120 ;--
0000    121
```

```
                        0000    123          .SBTTL  DECLARATIONS
                        0000    124
                        0000    125 ;
                        0000    126 ; Macros:
                        0000    127 ;
                        0000    128
                        0000    129          $FIDDEF
                        0000    130          $IODEF
                        0000    131          $RMSDEF
                        0000    132          $SSDEF
                        0000    133          $CCBDEF
                        0000    134          $DEVDEF
                        0000    135          $DRCDEF
                        0000    136          $FABDEF
                        0000    137          $FIBDEF
                        0000    138          $FWADEF
                        0000    139          $IFBDEF
                        0000    140          $IMPDEF
                        0000    141          $IPLDEF
                        0000    142          $LCKDEF
                        0000    143          $NAMDEF
                        0000    144          $QIODEF
                        0000    145          $FSCBDEF
                        0000    146          $SSDEF
                        0000    147          $UCBDEF
                        0000    148          $VCBDEF
                        0000    149
                        0000    150 ;
                        0000    151 ; Equated Symbols:
                        0000    152 ;
                        0000    153
                00000020 0000   154          FOP     = FAB$L_FOP*8   ; bit offset to fop
                        0000    155
                        0000    156 ;
                        0000    157 ; Own Storage:
                        0000    158 ;
                        0000    159
                        0000    160 DIR_SUFFIX:
      31 3B 52 49 44 2E 0000    161          .ASCII  /.DIR;1/                ; constant suffix for directory files
                        0006    162
```

```
0006   164          .SBTTL  RM$SETDID, Routine to set Directory File ID
0006   165
0006   166 ;++
0006   167 ;
0006   168 ; RM$SETDID  - Set directory ID
0006   169 ;
0006   170 ; The rm$setdid routine's function is to initialize the
0006   171 ; directory id field of the fib by setting it to the file id
0006   172 ; of the (lowest level) directory file.  it accomplishes this
0006   173 ; by performing the following operations:
0006   174 ;
0006   175 ;       1.  assumes the fib buffer descriptor is initialized.
0006   176 ;       2.  utilities the file id or directory id value from the
0006   177 ;           user's nam block if specified and if non-zero.
0006   178 ;           if found, returns to caller with fib fid or did filled in.
0006   179 ;       3.  otherwise, constructs the directory filename
0006   180 ;           based on the directory spec format
0006   181 ;               - if [grp,mbr] prefixes from 0 to 2 zeroes
0006   182 ;                 to each of the grp and mbr octal values
0006   183 ;                 to give a 6-character file name, e.g.,
0006   184 ;                 [1,20] gives 001020
0006   185 ;               - if [directory-name] format uses the name as given
0006   186 ;       4.  searches the directory cache for the specified device and directory
0006   187 ;           entries.
0006   188 ;       5.  if any entry not found, a new entry is made by looking up the directory.
0006   189 ;           in order to do the lookup, the code appends the fixed type and version
0006   190 ;           of '.dir;1' to the filename and issues a qio to lookup the file id
0006   191 ;           in the master file directory or lower level directory.
0006   192 ;       6.  the returned file id is copied to the directory id field of the fib
0006   193 ;       7.  the file id field of the fib is zeroed.
0006   194 ;
0006   195 ; Calling sequence:
0006   196 ;
0006   197 ;       bsbw    rm$setdid
0006   198 ;
0006   199 ; Input Parameters:
0006   200 ;
0006   201 ;       r11     impure area address
0006   202 ;       r10     fwa address
0006   203 ;       r9      ifab address
0006   204 ;       r8      fab address
0006   205 ;
0006   206 ; Implicit Inputs:
0006   207 ;
0006   208 ;       nam$w_did       -  directory id to use else zero
0006   209 ;       ifb$l_chnl      -  channel # for qio
0006   210 ;       ifb$l_prim_dev  -  device characteristics
0006   211 ;       fwa$q_cdirT...  - concealed directory spec element descriptors
0006   212 ;       fwa$q_dir1...   - directory spec element descriptors
0006   213 ;       fwa$q_dir+4     -  address of scratch buffer
0006   214 ;       fwa$t_fibbuf    -  must be zero
0006   215 ;       the directory cache
0006   216 ;
0006   217 ; Output Parameters:
0006   218 ;
0006   219 ;       r0      status code
0006   220 ;       r1-r7,ap destroyed
```

```
                     0006   221 ;
                     0006   222 ; Implicit Outputs:
                     0006   223 ;
                     0006   224 ;       fwa$q_fib        -  descriptor initialized
                     0006   225 ;       fwa$t_fibbuf+fib$w_did  -  directory file id initialized
                     0006   226 ;       fwa$t_fibbuf+fib$w_fid  -  set from nam$w_fid
                     0006   227 ;       ifb$l_ios        -  set to i/o status
                     0006   228 ;       fab$l_stv        -  set to system error code on error
                     0006   229 ;       the directory cache is updated.
                     0006   230 ;
                     0006   231 ; Completion Codes:
                     0006   232 ;
                     0006   233 ;       standard rms, in particular, suc, dnf and idr.
                     0006   234 ;
                     0006   235 ; Side Effects:
                     0006   236 ;
                     0006   237 ;       may have switched to running at ast level.
                     0006   238 ;       all user structures except fab must be re-probed.
                     0006   239 ;
                     0006   240 ;--
                     0006   241
                     0006   242 RM$SETDID::
                     0006   243         $TSTPT  SETDID
                     000C   244
                     000C   245 ;
                     000C   246 ; check if we really need to go through this code
                     000C   247 ;
                     000C   248
   69    03    E1    000C   249         BBC     #DEV$V_DIR,IFB$L_PRIM_DEV(R9),- ; branch if no directory
         32          000F   250                 SUCCESS                         ;
57    28 A8    D0    0010   251         MOVL    FAB$L_NAM(R8),R7                ; get nam block
      41    13    0014   252         BEQL    CHKMT                           ; branch if none
    FFE7'   30    0016   253         BSBW    RM$CHKNAM                       ; verify nam
    29 50     E9    0019   254         BLBC    R0,RETURN                       ; if not ok exit
                     001C   255
                     001C   256 ;
                     001C   257 ; try to get file id from nam block
                     001C   258 ;
                     001C   259
37 6A    02    E0    001C   260         BBS     #FWA$V_SL_PASS,(R10),CHKMT      ; ignore if in search list oper
33 68    38    E1    0020   261         BBC     #FAB$V_NAM+FOP,(R8),CHKMT       ; branch if not doing nam blk open
   24 A7    D0    0024   262         MOVL    NAM$W_FID(R7),-                 ; get file-id
01F8 CA          0027   263                 FIB$W_FID+FWA$T_FIBBUF(R10)     ;
      08    13    002A   264         BEQL    10$                             ; branch if none
   28 A7    B0    002C   265         MOVW    NAM$W_FID_RVN(R7),-             ; copy relative vol number too
01FC CA          002F   266                 FIB$W_FID_RVN+FWA$T_FIBBUF(R10) ;
      0E    11    0032   267         BRB     SUCCESS                         ; all done
                     0034   268
                     0034   269 ;
                     0034   270 ;  try to get directory id from nam block
                     0034   271 ;
                     0034   272
   2A A7    B0    0034   273 10$:    MOVW    NAM$W_DID(R7),-                 ; pick up directory id from nam blk
01FE CA          0037   274                 FIB$W_DID_NUM+FWA$T_FIBBUF(R10) ;
      1B    13    003A   275         BEQL    CHKMT                           ; branch if not specified
                     003C   276
                     003C   277         ASSUME  FIB$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   302
                     0057   303 ;++
                     0057   304 ;
                     0057   305 ; directory id wasn't in nam block.  get it from directory cache.
                     0057   306 ;
                     0057   307 ;  alternate entry if nam block not to be used for input (from rms$rename)
                     0057   308 ;
                     0057   309 ;--
                     0057   310
                     0057   311 RM$SETDID_ALT::
              05  E0 0057   312 CHKMT:  BBS     #DEV$V_SQD,-
              69     0059   313                 IFB$L_PRIM_DEV(R9),-
              EB     005A   314                 SET_MT_MFD              ; branch if magtape
                     005B   315
                     005B   316 ;++
                     005B   317 ;
                     005B   318 ;  locate the device id in the directory cache
                     005B   319 ;
                     005B   320 ;--
                     005B   321
        53   40 AA  D0 005B   322         MOVL    FWA$Q_DIR+4(R10),R3     ; set addr of scratch buffer
              53  DD 005F   323         PUSHL   R3                      ; Push address of buffer onto stack.
     83   0198 CA  90 0061   324         MOVB    FWA$Q_SHRFIL_LCK(R10),(R3)+ ; Make first byte count of string
          0198 CA  28 0066   325         MOVC3   FWA$Q_SHRFIL_LCK(R10),- ; Move device id string into buffer
          019C DA     006A   326                 @FWA$Q_SHRFIL_LCK+4(R10),-  ;(this is the unreadable form)
                63    006D   327                 (R3)
              54 8ED0 006E   328         POPL    R4                      ; Pop address of buffer into R4.
            0281  30 0071   329         BSBW    RM$GETCCB               ; Get CCB address in R1.
        56    61  D0 0074   330         MOVL    CCB$L_UCB(R1),R6        ; Get UCB address.
    57  00AC  C6  3E 0077   331         MOVAW   UCB$W_DIRSEQ(R6),R7     ; Save UCB dirseq address here.
55   00000000'9F  DE 007C   332         MOVAL   @#PIO$GL_DIRCACHE,R5    ; addr of device list head
                     0083   333
            01C2  30 0083   334         BSBW    FIND_ENTRY              ; go find this entry in cache
              10  13 0086   335         BEQL    10$                     ; branch if none found
     67   3A AC  B1 0088   336         CMPW    DRC$W_DIRSEQ(AP),(R7)   ; cache entry still valid?
              24  13 008C   337         BEQL    20$                     ; branch if yes
        55    5C  D0 008E   338         MOVL    AP,R5                   ; get device node to correct reg
     38 AA    5C  D0 0091   339         MOVL    AP,FWA$L_DEVNODADR(R10) ; save the device node address
            015E  31 0095   340         BRW     PRUNE                   ; and go prune back branch
                     0098   341
                     0098   342 ;
                     0098   343 ;  no entry for this device in the directory cache.  -  make one.
                     0098   344 ;
                     0098   345
            01E6  30 0098   346 10$:    BSBW    GET_FREE                ; go pick a free node
        65   6C  0E 009B   347         INSQUE  (AP),(R5)               ; insert node at list head
        50   64  9B 009E   348         MOVZBW  (R4),R0                 ; get length of device string
10 AC   64   50  28 00A1   349         MOVC3   R0,(R4),DRC$T_NAME(AP)  ; move the device string
                     00A6   350         ASSUME  UCB$V_AST_ARMED  EQ  15
     3A AC   67  B0 00A6   351 15$:    MOVW    (R7),DRC$W_DIRSEQ(AP)   ; save the dir seq. count
              06  19 00AA   352         BLSS    20$                     ; branch if cache blocking AST is armed
            0250  30 00AC   353         BSBW    RM$ARM_DIRCACHE         ; Arm it
          F4 50   E8 00AF   354         BLBS    R0,15$                  ; Repeat saving DIRSEQ if successful
     38 AA    5C  D0 00B2   355 20$:    MOVL    AP,FWA$L_DEVNODADR(R10) ; save the device node address
```

```
                                00B6   357
                                00B6   358 ;++
                                00B6   359 ;
                                00B6   360 ;  follow the directory cache entries for this directory spec.
                                00B6   361 ;  if any missing, do a lookup to supply the entry and restart scan from the top.
                                00B6   362 ;
                                00B6   363 ;--
                                00B6   364
                                00B6   365 CLR_LOOKUP:
                      34 AA     D4 00B6   366         CLRL    FWA$L_LOOKUP(R10)       ; say no lookup done
                                00B9   367
                                00B9   368 ;
                                00B9   369 ; If a root directory is defined, then locate the root directory string
                                00B9   370 ; before starting the (sub)directory lookups.  This is necessary since the
                                00B9   371 ; the UFD in this case is actually an SFD of the root directory.
                                00B9   372 ;
                                00B9   373
                                00B9   374 FIRST_DIR:
                   5C   38 AA   D0 00B9   375         MOVL    FWA$L_DEVNODADR(R10),AP         ; reset device node address
                56    0130 CA   7E 00BD   376         MOVAQ   FWA$Q_DIR1(R10),R6              ; get addr of 1st dir. discriptor
                   05 6A    3A   E1 00C2   377         BBC     #FWA$V_ROOT_DIR,(R10),BLDNAM    ; no root present
                56    00F0 CA   7E 00C6   378         MOVAQ   FWA$Q_CDIR1(R10),R6             ; if a concealed directory
                                00CB   379                                                 ; start from there
                                00CB   380
                                00CB   381 ;
                                00CB   382 ; Construct directory name
                                00CB   383 ;
                                00CB   384
53              40 AA    01    C3 00CB   385 BLDNAM: SUBL3   #1,FWA$Q_DIR+4(R10),R3           ; get name scratch buffer
                1D 66    16    E0 00D0   386         BBS     #FSCB$V_GRPMBR,(R6),10$          ; branch if [grp,mbr] format
                0A 6A    3A    E1 00D4   387         BBC     #FWA$V_ROOT_DIR,(R10),5$         ; skip MFD test if not rooted
                06 66    19    E1 00D8   388         BBC     #FSCB$V_MFD,(R6),5$              ; branch if not MFD string
                   56    08    C0 00DC   389         ADDL2   #8,R6                            ; skip this directory
                      00B1    31 00DF   390         BRW     NEXT_DIR
                                00E2   391
                                00E2   392 ;
                                00E2   393 ; Directory name is in [name1.name2...] format construct the current
                                00E2   394 ; level directory name
                                00E2   395 ;
                                00E2   396
                   53    DD 00E2   397 5$:     PUSHL   R3                               ; save buff start addr
                50    86    D0 00E4   398         MOVL    (R6)+,R0                         ; get name length
             83    50    01    81 00E7   399         ADDB3   #1,R0,(R3)+                      ; store length count in string
             63    96    50    28 00EB   400         MOVC3   R0,@(R6)+,(R3)                   ; move to temporary buffer
                   0B    11    11 00EF   401         BRB     20$                              ; go look up the file id
                                00F1   402
                                00F1   403 ;
                                00F1   404 ; directory name is in [grp,mbr] format.
                                00F1   405 ; build the directory name from the two values, prefixing
                                00F1   406 ; with leading zeroes if neccessary to get a 6-character name
                                00F1   407 ;
                                00F1   408
                   53    DD 00F1   409 10$:    PUSHL   R3                               ; save buff start addr
                83    07    90 00F3   410         MOVB    #7,(R3)+                         ; count of string to match
                   01E4    30 00F6   411         BSBW    PREFIX_0                         ; move group part
                   01E1    30 00F9   412         BSBW    PREFIX_0                         ; move member part
                                00FC   413
```

```
                                00FC    414 ;
                                00FC    415 ;  look up file in cache
                                00FC    416 ;
                                00FC    417
                     54 8ED0    00FC    418 20$:    POPL    R4                          ; restore counted string addr
                55   08 AC  9E  00FF    419         MOVAB   DRC$L_LVLFLNK(AP),R5        ; addr of list hdr for nxt level
                      0142  30  0103    420         BSBW    FIND_ENTRY                  ; go find this directory entry
                        16  12  0106    421         BNEQ    NXT_DIR                     ; next_dir if found
                                0108    422
                                0108    423 ;++
                                0108    424 ;
                                0108    425 ; No entry for this (sub)directory in the cache.  We must lookup the file and
                                0108    426 ; make and entry.  Because a more priviliged mode could invalidate the cache
                                0108    427 ; while we stall, (verrrry unlikely, but possible), we must find our way back
                                0108    428 ; down to this level before actually adding the new entry.
                                0108    429 ;
                                0108    430 ;--
                                0108    431
                5C   34 AA   D0  0108    432 NOT_FND:MOVL    FWA$L_LOOKUP(R10),AP        ; get addr node for last lookup
                        15   13  010C    433         BEQL    LOOKUP                      ; branch if none
                                010E    434
                                010E    435 ;
                                010E    436 ;  this is the 2nd time thru.  lookup has already been done.  add the looked-up
                                010E    437 ;  entry to the cache as long as it's still the one we want.
                                010E    438 ;
                                010E    439
                     34 AA   D4 010E    440         CLRL    FWA$L_LOOKUP(R10)           ; indicate no lookup node
                50      64   9B  0111    441         MOVZBW  (R4),R0                     ; get length of string
          10 AC 64      50   29  0114    442         CMPC3   R0,(R4),DRC$T_NAME(AP)      ; is this the right entry?
                        05   12  0119    443         BNEQ    FREE_UP                     ; branch if not
                65      6C   0E  011B    444         INSQUE  (AP),(R5)                   ; insert new node after header
                        73   11  011E    445 NXT_DIR:BRB     NEXT_DIR                    ; and continue
                                0120    446
                      01AC   30  0120    447 FREE_UP:BSBW    ADD_TO_FREE                 ; return the node
                                0123    448
                                0123    449 ;
                                0123    450 ;  must look up the file.  use the current cache node to set the did.
                                0123    451 ;
                                0123    452
                55      08   C2  0123    453 LOOKUP: SUBL2   #DRC$L_LVLFLNK,R5           ; back to start of current node
             38 AA      55   D1  0126    454         CMPL    R5,FWA$L_DEVNODADR(R10)     ; is this the device node?
                        05   12  012A    455         BNEQ    10$                         ; branch if not
                    FF1A     30  012C    456         BSBW    SETMFD                      ; go set mfd did
                        0C   11  012F    457         BRB     20$                         ; continue
                   38 A5     D0  0131    458 10$:    MOVL    DRC$W_DID(R5),-
                01FE CA                 0134    459                 FIB$W_DID+FWA$T_FIBBUF(R10) ; set the did from cur. node
                   3C A5     B0  0137    460         MOVW    DRC$W_DID+4(R5),-
                0202 CA                 013A    461                 FIB$W_DID+4+FWA$T_FIBBUF(R10) ;  (ditto)
                                013D    462
                                013D    463 ;
                                013D    464 ; append '.dir;1' to the directory name, determine
                                013D    465 ; the total string length, and perform qio to get the file-id
                                013D    466 ;
                                013D    467
                53      64   9A  013D    468 20$:    MOVZBL  (R4),R3                     ; set size of dir name string
          01 A4 53      2A   3A  0140    469         LOCC    #^A'*',R3,1(R4)             ; '*' in file name?
                        46   12  0145    470         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
```

```
                     0193   505
                     0193   506 ;++
                     0193   507 ;
                     0193   508 ; Found this directory entry o.k.  -  see if more to find
                     0193   509 ;
                     0193   510 ; The directory descriptors are organized as follows:
                     0193   511 ;
                     0193   512 ;       FWA$Q_CDIR1     - Concealed directory descriptors
                     0193   513 ;             .
                     0193   514 ;             .
                     0193   515 ;             .
                     0193   516 ;       FWA$Q_CDIR8
                     0193   517 ;       FWA$Q_DIR1      - Followed by the normal directory descriptors
                     0193   518 ;             .
                     0193   519 ;             .
                     0193   520 ;             .
                     0193   521 ;       FWA$Q_DIR8
                     0193   522 ;
                     0193   523 ; If a zero entry is found we must check to see which group we are in.  If
                     0193   524 ; it is the concealed list we start with the normal directory descriptors.
                     0193   525 ; If there are 8 concealed directories this loop will fall right through
                     0193   526 ; and search the normal ones.
                     0193   527 ;
                     0193   528 ;--
                     0193   529
                     0193   530
                     0193   531 ;
                     0193   532 ; Pick up the next sub-directory name if any more to go
                     0193   533 ;
                     0193   534
                     0193   535 NEXT_DIR:
           66   B5   0193   536         TSTW    (R6)                            ; zero directory length?
           15   12   0195   537         BNEQ    20$                             ; branch if not
                     0197   538
                     0197   539 ;
                     0197   540 ; If the descriptor is zero see if we have passed the concealed directory
                     0197   541 ; descriptors.  If so we are done, else start on the normal directories
                     0197   542 ;
                     0197   543
     0128 CA   7F    0197   544         PUSHAQ  FWA$Q_CDIR8(R10)                ; get lowest level concealed
                     019B   545                                                 ;  directory descriptor addr
     8E   56   D1    019B   546         CMPL    R6,(SP)+                        ; past it already?
           15   1A    019E   547         BGTRU   EXIT                            ; branch if yes (all done)
56   0130 CA   7E    01A0   548         MOVAQ   FWA$Q_DIR1(R10),R6              ; start one normal dir list
           66   B5    01A5   549         TSTW    (R6)                            ; zero directory length?
           0C   13    01A7   550         BEQL    EXIT                            ; exit if so (no normal dirs)
        FF1F   31    01A9   551 10$:     BRW     BLDNAM
                     01AC   552
     0168 CA   7F    01AC   553 20$:     PUSHAQ  FWA$Q_DIR8(R10)                 ; get lowest level sub
                     01B0   554                                                 ;  directory descriptor addr
     8E   56   D1    01B0   555         CMPL    R6,(SP)+                        ; past it already?
           F4   1B    01B3   556         BLEQU   10$                             ; branch not
                     01B5   557
                     01B5   558 ;++
                     01B5   559 ;
                     01B5   560 ; Have found all needed nodes.  Check if directory sequence count still valid.
                     01B5   561 ;
```

```
                     01B5   562 ;--
                     01B5   563
    56   5C   D0    01B5   564 EXIT:   MOVL    AP,R6                           ; save addr of dir node
5C    34   AA   D0    01B8   565        MOVL    FWA$L_LOOKUP(R10),AP            ; unused lookup node?
          03   13    01BC   566        BEQL    10$                             ; branch if not
        010E   30    01BE   567        BSBW    ADD_TO_FREE                     ; return it to the free list
          7C   10    01C1   568 10$:    BSBB    CHKDIRSEQ                       ; cache still valid?
          31   12    01C3   569        BNEQ    PRUNE                           ; branch if not
                     01C5   570
                     01C5   571 ;++
                     01C5   572 ;
                     01C5   573 ;  All set.  Just set the did in the fib and clear the fid and version limit.
                     01C5   574 ;
                     01C5   575 ;--
                     01C5   576
38   A6   06   28    01C5   577        MOVC3   #6,DRC$W_DID(R6),-              ; set the directory id
     01FE   CA        01C9   578                FIB$W_DID+FWA$T_FIBBUF(R10)
     01F8   CA   D4   01CC   579        CLRL    FIB$W_FID+FWA$T_FIBBUF(R10)     ; zero the file id
     0220   CA   B4   01D0   580        CLRW    FIB$W_VERLIMIT+FWA$T_FIBBUF(R10); zero the version limit
          50   D6   01D4   581        INCL    R0                              ; show success (r0 = 0 from movc3)
               05   01D6   582        RSB                                     ; back to caller of rm$setdid
                     01D7   583
```

```
                     01D7   585
                     01D7   586 ;++
                     01D7   587 ;
                     01D7   588 ; Handle directory not found error.
                     01D7   589 ;
                     01D7   590 ;--
                     01D7   591
                     01D7   592 ERRDNF:
          66   10    01D7   593          BSBB    CHKDIRSEQ                   ; error due to invalid cache?
          02   13    01D9   594          BEQL    5$                          ; branch if not
          19   11    01DB   595          BRB     PRUNE                       ; possibly - go try again
0910 8F   50   B1    01DD   596 5$:      CMPW    R0,#SS$_NOSUCHFILE          ; was error file not found?
          0A   12    01E2   597          BNEQ    10$                         ; branch if not
  0C A8   50   D0    01E4   598          MOVL    R0,FAB$L_STV(R8)            ; save system code
                     01E8   599          RMSERR  DNF                         ; replace with directory not found
               05    01ED   600          RSB                                 ; and return
                     01EE   601 10$:     RMSERR  DNF,R1                      ; default error to directory not found
        FE0A'  31    01F3   602          BRW     RM$MAPERR                   ; map error to rms & return
                     01F6   603
```

```
                     01F6   605
                     01F6   606 ;++
                     01F6   607 ;
                     01F6   608 ;  have run into an invalid cache condition, i.e., something was done
                     01F6   609 ;  by the acp (e.g., mount) to invalidate the cache contents.
                     01F6   610 ;  remove all entries below the device, reset dirseq, and try again.
                     01F6   611 ;
                     01F6   612 ;--
                     01F6   613
                     01F6   614 PRUNE:
     55    08    C0  01F6   615          ADDL2   #DRC$L_LVLFLNK,R5       ; get address of ufd header
     54    55    D0  01F9   616          MOVL    R5,R4                   ; set stop address
     55    65    D1  01FC   617          CMPL    (R5),R5                 ; anything to prune?
           26    13  01FF   618          BEQL    30$                     ; branch if not
           55    DD  0201   619 10$:     PUSHL   R5                      ; save header addr
     5C    65    D0  0203   620          MOVL    (R5),AP                 ; get next level down
55   08 AC    DE  0206   621 15$:     MOVAL   DRC$L_LVLFLNK(AP),R5    ; get addr of level link
     55    65    D1  020A   622          CMPL    (R5),R5                 ; another level?
           F2    12  020D   623          BNEQ    10$                     ; branch if yes
                     020F   624
                     020F   625 ;
                     020F   626 ;  at lowest level  -  remove this node and move to side node
                     020F   627 ;
                     020F   628
           6C    DD  020F   629 20$:     PUSHL   (AP)                    ; save next node addr
     5C    6C    0F  0211   630          REMQUE  (AP),AP                 ; remove node
        00B8    30  0214   631          BSBW    ADD_TO_FREE             ; add it to the free list
     5C    8E    D0  0217   632          MOVL    (SP)+,AP                ; get next node addr
     6E    5C    D1  021A   633          CMPL    AP,(SP)                 ; back to previous level?
           E7    12  021D   634          BNEQ    15$                     ; branch if not
     5C    08    C2  021F   635          SUBL2   #DRC$L_LVLFLNK,AP       ; get node start address
     54    8E    D1  0222   636          CMPL    (SP)+,R4                ; back to dev node?
           E8    12  0225   637          BNEQ    20$                     ; branch if not
                     0227   638
                     0227   639 ;
                     0227   640 ;  store new dirseq value and rebuild tree for this device
                     0227   641 ;
                     0227   642
                     0227   643          ASSUME  UCB$V_AST_ARMED  EQ  15
32 A4      67    B0  0227   644 30$:     MOVW    (R7),DRC$W_DIRSEQ-DRC$L_LVLFLNK(R4)
           06    19  022B   645          BLSS    40$                     ; branch if cache blocking AST is armed
        00CF    30  022D   646          BSBW    RM$ARM_DIRCACHE         ; Arm it
      F4 50    E8  0230   647          BLBS    R0,30$                  ; Repeat saving DIRSEQ if successful
        FE80    31  0233   648 40$:     BRW     CLR_LOOKUP
```

```
                     0236   650
                     0236   651 ;++
                     0236   652 ;
                     0236   653 ;  handle bad directory rename error.
                     0236   654 ;
                     0236   655 ;--
                     0236   656
          50 8ED0    0236   657 ERRIDR: POPL    R0                          ; discard local ret addr
                     0239   658         RMSERR  IDR                         ; set bad directory rename
               05    023E   659         RSB                                 ; and return
                     023F   660
                     023F   661 ;++
                     023F   662 ;
                     023F   663 ;  chkdirseq subroutine to verify cache validity
                     023F   664 ;
                     023F   665 ;  inputs:
                     023F   666 ;                  r10     fwa address
                     023F   667 ;                  r7      ucb$w_dirseq address
                     023F   668 ;                  fwa$l_devnodadr
                     023F   669 ;
                     023F   670 ;  outputs:
                     023F   671 ;                  r5      fwa$l_devnodadr
                     023F   672 ;                  z-bit   set if cache valid, else clear
                     023F   673 ;--
                     023F   674
                     023F   675 CHKDIRSEQ:
55   38 AA    D0     023F   676         MOVL    FWA$L_DEVNODADR(R10),R5 ; get device node address
67   3A A5    B1     0243   677         CMPW    DRC$W_DIRSEQ(R5),(R7)   ; still valid?
               05    0247   678         RSB
                     0248   679
```

```
                     0248   681
                     0248   682 ;++
                     0248   683 ;
                     0248   684 ;  find_entry subroutine to find an entry in the directory cache
                     0248   685 ;
                     0248   686 ;  inputs:
                     0248   687 ;                  r4      address of counted string to match
                     0248   688 ;                  r5      address of list head for level to scan
                     0248   689 ;
                     0248   690 ;  outputs:
                     0248   691 ;                  z-bit   set if no match found, else clear
                     0248   692 ;                  ap      address of matching entry
                     0248   693 ;                  r0-r3   destroyed
                     0248   694 ;
                     0248   695 ;  note: if match found, matching entry is requeued to immediately follow list head.
                     0248   696 ;
                     0248   697 ;--
                     0248   698
                     0248   699 FIND_ENTRY:
          5C   55 D0 0248   700          MOVL    R5,AP                           ; set up to find 1st node
          5C   6C D0 024B   701 10$:     MOVL    DRC$L_NXTFLNK(AP),AP            ; get next node
          55   5C D1 024E   702          CMPL    AP,R5                           ; back at head?
               2D 13 0251   703          BEQL    20$                             ; branch if yes (no match)
          50   64 9B 0253   704          MOVZBW  (R4),R0                         ; get len of string to match
    10 AC 64   50 29 0256   705          CMPC3   R0,(R4),DRC$T_NAME(AP)          ; do they match?
               EE 12 025B   706          BNEQ    10$                             ; branch if not
                     025D   707
                     025D   708 ;
                     025D   709 ;  matching entry found  -  requeue entry to head of the list
                     025D   710 ;
                     025D   711
          5C   6C OF 025D   712          REMQUE  (AP),AP
          65   6C 0E 0260   713          INSQUE  (AP),(R5)
                     0263   714
                     0263   715 ;
                     0263   716 ; check if the saved fid is the same as this directory id.
                     0263   717 ; true iff both fid and did are valid file identification fields.
                     0263   718 ;
                     0263   719
          0240 CA A9 0263   720          BISW3   FWA$T_RNM_FID(R10),-    ; is this a valid fid?
    50    0244 CA    0267   721                  FWA$T_RNM_FID+4(R10),R0 ; not if 1st and 3rd words are 0
               11 13 026B   722          BEQL    15$
            38 AC A9 026D   723          BISW3   DRC$W_DID(AP),-
            3C AC    0270   724                  DRC$W_DID+4(AP),-
               50    0272   725                  R0                              ; likewise for did
               09 13 0273   726          BEQL    15$
    0240 CA    06 29 0275   727          CMPC3   #6,FWA$T_RNM_FID(R10),- ; is directory same as file?
            38 AC    027A   728                  DRC$W_DID(AP)
               B8 13 027C   729          BEQL    ERRIDR                          ; bad operation if so
               5C D5 027E   730 15$:     TSTL    AP                              ; clear z-bit
                  05 0280   731 20$:     RSB
```

```
                     0281   733
                     0281   734 ;++
                     0281   735 ;
                     0281   736 ;  get_free subroutine to find a free node.
                     0281   737 ;  picks node from free list, if any, else picks least recently used dir entry
                     0281   738 ;  on least recently used device.
                     0281   739 ;
                     0281   740 ;  inputs:
                     0281   741 ;                  none
                     0281   742 ;
                     0281   743 ;  outputs:
                     0281   744 ;                  ap      addr of node
                     0281   745 ;                  r0      destroyed
                     0281   746 ;--
                     0281   747
                     0281   748 GET_FREE:
50   00000000'9F DE  0281   749          MOVAL   @#PIO$GL_DIRCFRLH,R0    ; get free list addr
     5C     60   D0  0288   750          MOVL    (R0),AP                 ; pick first node
            04   13  028B   751          BEQL    10$                     ; branch if none
     60     6C   D0  028D   752          MOVL    (AP),(R0)               ; bring up next free node
            05   0290   753          RSB
                     0291   754
                     0291   755 ;
                     0291   756 ;  nothing on free list.  check that it has been initialized.
                     0291   757 ;
                     0291   758
     5C   04'A0   D0 0291   759 10$:    MOVL    B^PIO$GL_DIRCACHE+4-PIO$GL_DIRCFRLH(R0),AP
                     0295   760
                     0295   761 ;
                     0295   762 ; get lru device node
                     0295   763 ;
                     0295   764
     6C   5C   D1    0295   765          CMPL    AP,(AP)                 ; empty list?
          28   12    0298   766          BNEQ    30$                     ; branch if not
                     029A   767
                     029A   768 ;
                     029A   769 ;  initialize free directory nodes list
                     029A   770 ;
                     029A   771
     5C   50   D0    029A   772          MOVL    R0,AP                   ; set up to init free node list head
50   0000'C0   DE    029D   773          MOVAL   PIO$A_DIRCACHE-PIO$GL_DIRCFRLH(R0),R0
                     02A2   774
                     02A2   775 ;
                     02A2   776 ; get addr of directory cache page
                     02A2   777 ;
                     02A2   778 ; NOTE: There must be enough room in the directory cache to have a node
                     02A2   779 ;       for each possible subdirectoy plus the root and the device node.
                     02A2   780 ;
                     02A2   781
                     02A2   782          ASSUME  <<2*512>/DRC$C_BLN>     GE      FWA$C_MAXSUBDIR+1+1
                     02A2   783
          10   DD    02A2   784          PUSHL   #<<2*512>/DRC$C_BLN>    ; set # of nodes in cache (2 pages)
     6C   50   D0    02A4   785 15$:     MOVL    R0,(AP)                 ; set flink of previous node
     5C   50   D0    02A7   786          MOVL    R0,AP                   ; save addr this flink for next node
                     02AA   787
                     02AA   788          ASSUME  DRC$L_LVLFLNK   EQ      8
                     02AA   789
```

```
                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    826          .SBTTL  PREFIX_0, Convert Group-Member Format to Normal Directory
                     02DD    827
                     02DD    828 ;++
                     02DD    829 ;
                     02DD    830 ; subroutine prefix_0 to move either the group or
                     02DD    831 ; member part of a directory spec prefixing it
                     02DD    832 ; with one or two zeros to give 3 characters total
                     02DD    833 ;
                     02DD    834 ; inputs:
                     02DD    835 ;
                     02DD    836 ;       r6      addr of descriptor for group of member part
                     02DD    837 ;       r3      addr of output buffer
                     02DD    838 ;
                     02DD    839 ; outputs:
                     02DD    840 ;
                     02DD    841 ;       r6      r6+8
                     02DD    842 ;       r3      r3+3
                     02DD    843 ;       r0,r1,r2,r4,r5  destroyed
                     02DD    844 ;
                     02DD    845 ;--
                     02DD    846
                     02DD    847 PREFIX_0:
          50    86   D0  02DD    848          MOVL    (R6)+,R0                ; get length
          03    50   B1  02E0    849          CMPW    R0,#3                   ; all 3 chars present?
                0B   13  02E3    850          BEQL    20$                     ; branch if yes
          02    50   B1  02E5    851          CMPW    R0,#2                   ; 2 of the 3?
                03   13  02E8    852          BEQL    10$                     ; branch if yes
          83    30   90  02EA    853          MOVB    #^A/0/,(R3)+            ; move a zero
          83    30   90  02ED    854 10$:     MOVB    #^A/0/,(R3)+            ; move a zero
    63    96    50   28  02F0    855 20$:     MOVC3   R0,@(R6)+,(R3)          ; move the grp or mbr number
                05   02F4    856          RSB
                     02F5    857
```

```
                     02F5    859          .SBTTL  RM$GETCCB, GET CCB ADDRESS
                     02F5    860 ;++
                     02F5    861 ;
                     02F5    862 ; RM$GETCCB --  subroutine to return the CCB address.
                     02F5    863 ;
                     02F5    864 ;  INPUTS:
                     02F5    865 ;
                     02F5    866 ;       R9      IFAB address with channel in IFB$W_CHNL
                     02F5    867 ;
                     02F5    868 ;  OUTPUTS:
                     02F5    869 ;
                     02F5    870 ;       R1      address of CCB
                     02F5    871 ;       R0,R2,R3 destroyed
                     02F5    872 ;
                     02F5    873 ;--
                     02F5    874
                     02F5    875 RM$GETCCB::
 50    20 A9    3C    02F5    876          MOVZWL  IFB$W_CHNL(R9),R0       ; set channel #
00000000'9F    17    02F9    877          JMP     @#IOC$VERIFYCHAN        ; get the ccb address (in r1)
                     02FF    878
                     02FF    879 ;
                     02FF    880 ; and return
                     02FF    881 ;
```

```
                     02FF    883 ;++
                     02FF    884 ;
                     02FF    885 ;  rm$arm_dircache routine.  Converts the volume lock to rearm
                     02FF    886 ;  the blocking AST which in turn invalidates the cache.
                     02FF    887 ;
                     02FF    888 ;  inputs:
                     02FF    889 ;                  r7      address of UCB$W_DIRSEQ cell in UCB
                     02FF    890 ;
                     02FF    891 ;  outputs:
                     02FF    892 ;
                     02FF    893 ;                  r0      low bit clear = failure (blocking ast could
                     02FF    894 ;                          not be rearmed).  Just save current seq. #.
                     02FF    895 ;
                     02FF    896 ;                          low bit set = success.  Repeat saving of the
                     02FF    897 ;                          seq. #.
                     02FF    898 ;
                     02FF    899 ;--
                     02FF    900
                     02FF    901 RM$ARM_DIRCACHE::
                     02FF    902 ;
                     02FF    903 ; Duplicate some checks we will make in kernel mode only so that
                     02FF    904 ; if there is no lock (ODS-1, for example) we save the $CMKRNL.
                     02FF    905 ;
          50   D4   02FF    906         CLRL    R0                      ; assume failure
51   FF54 C7   3E   0301    907         MOVAW   -UCB$W_DIRSEQ(R7),R1    ; get address of UCB
          13   E1   0306    908         BBC     #DEV$V_MNT,-            ; Return failure if device
     21 38 A1         0308    909                 UCB$L_DEVCHAR(R1),50$   ; is not mounted
50        34 A1   D0   030B    910         MOVL    UCB$L_VCB(R1),R0        ; Or if VCB isn't attached
          1B   13   030F    911         BEQL    50$
        7C A0   D5   0311    912         TSTL    VCB$L_VOLLKID(R0)       ; Or if there is no volume lock
          16   13   0314    913         BEQL    50$                     ; (ODS-1, for example)
          51   DD   0316    914         PUSHL   R1                      ; push address of UCB
          01   DD   0318    915         PUSHL   #1                      ; push argument count
     51   5E   D0   031A    916         MOVL    SP,R1                   ; r1 points to argument list
                     031D    917         $CMKRNL_S   B^ARM_CACHE,(R1)    ; call kernel mode routine
     5E   08   C0   0329    918         ADDL    #8,SP                   ; clean argument list off stack
               05   032C    919 50$:    RSB
                     032D    920
                     032D    921 ARM_CACHE:
               003C  032D    922         .WORD   ^M<R2,R3,R4,R5>
  55   04 AC   D0   032F    923         MOVL    4(AP),R5                ; Get UCB address
                     0333    924 10$:    SETIPL  IPL_DES                 ; raise IPL and lock pages
          13   E1   033A    925 20$:    BBC     #DEV$V_MNT,-            ; Return failure if device
     6B 38 A5         033C    926                 UCB$L_DEVCHAR(R5),50$   ; is not mounted
  53   34 A5   D0   033F    927         MOVL    UCB$L_VCB(R5),R3        ; Or if VCB isn't attached
          65   13   0343    928         BEQL    50$
  54   7C A3   D0   0345    929         MOVL    VCB$L_VOLLKID(R3),R4    ; Or if there is no volume lock
          5F   13   0349    930         BEQL    50$                     ; (ODS-1, for example)
52   00AC C5   3C   034B    931         MOVZWL  UCB$W_DIRSEQ(R5),R2     ; Save old seq. #
                     0350    932         SETIPL  #0                      ; lower IPL
                     0353    933
                     0353    934 ;
                     0353    935 ;  Set up for $ENQ service to convert lock to rearm blocking AST.  We must
                     0353    936 ;  handle possible SS$_IVLOCKID errors due to lock manager's handling
                     0353    937 ;  of conversions mastered on other systems.  If two users are in this
                     0353    938 ;  path simultaneously then the first will do the conversion and the
                     0353    939 ;  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
```

```
$$.PSECT_EP                     =  00000000
$$ARGS                          =  0000000C
$$RMSTEST                       =  0000001A
$$RMS_PBUGCHK                   =  00000010
$$RMS_TBUGCHK                   =  00000008
$$RMS_UMODE                     =  00000004
$$T1                            =  00000000
ADD_TO_FREE                        000002CF R      01
ARM_CACHE                          0000032D R      01
BLDNAM                             000000CB R      01
CCB$L_UCB                       =  00000000
CHKDIRSEQ                          0000023F R      01
CHKMT                              00000057 R      01
CLR_LOOKUP                         000000B6 R      01
DEV$V_DIR                       =  00000003
DEV$V_MNT                       =  00000013
DEV$V_SQD                       =  00000005
DIR_SUFFIX                         00000000 R      01
DRC$C_BLN                       =  0000003E
DRC$L_LVLBLNK                   =  0000000C
DRC$L_LVLFLNK                   =  00000008
DRC$L_NXTFLNK                   =  00000000
DRC$T_NAME                      =  00000010
DRC$W_DID                       =  00000038
DRC$W_DIRSEQ                    =  0000003A
ERRDIR                             0000018D R      01
ERRDNF                             000001D7 R      01
ERRIDR                             00000236 R      01
EXIT                               000001B5 R      01
FAB$L_FOP                       =  00000004
FAB$L_NAM                       =  00000028
FAB$L_STV                       =  0000000C
FAB$V_NAM                       =  00000018
FIB$L_WCC                       =  00000010
FIB$W_DID                       =  0000000A
FIB$W_DID_NUM                   =  0000000A
FIB$W_DID_RVN                   =  0000000E
FIB$W_DID_SEQ                   =  0000000C
FIB$W_FID                       =  00000004
FIB$W_FID_RVN                   =  00000008
FIB$W_VERLIMIT                  =  0000002C
FID$C_MFD                       =  00000004
FIND_ENTRY                         00000248 R      01
FIRST_DIR                          000000B9 R      01
FOP                             =  00000020
FREE_UP                            00000120 R      01
FSCB$V_GRPMBR                   =  00000016
FSCB$V_MFD                      =  00000019
FTL$_ENQDEQFAIL                 =  FFFFFFF2
FWA$C_MAXSUBDIR                 =  00000007
FWA$L_DEVNODADR                 =  00000038
FWA$L_LOOKUP                    =  00000034
FWA$Q_CDIR1                     =  000000F0
FWA$Q_CDIR8                     =  00000128
FWA$Q_DIR                       =  0000003C
FWA$Q_DIR1                      =  00000130
FWA$Q_DIR8                      =  00000168
FWA$Q_SHRFIL_LCK                =  00000198
FWA$T_FIBBUF                    =  000001F4
FWA$T_RNM_FID                   =  00000240
FWA$V_ROOT_DIR                  =  0000003A
FWA$V_SL_PASS                   =  00000002
GET_FREE                           00000281 R      01
IFB$L_PRIM_DEV                  =  00000000
IFB$W_CHNL                      =  00000020
IMP$C_ASYQIOEFN                 =  0000001F
IO$_ACCESS                      =  00000032
IOC$VERIFYCHAN                     ********   X    01
IPL$_SYNCH                      =  00000008
IPL_DEST                           000003B5 R      01
LCK$K_CRMODE                    =  00000001
LCK$M_CONVERT                   =  00000002
LCK$M_CVTSYS                    =  00000040
LCK$M_SYNCSTS                   =  00000008
LOOKUP                             00000123 R      01
NAM$W_DID                       =  0000002A
NAM$W_DID_SEQ                   =  0000002C
NAM$W_FID                       =  00000024
NAM$W_FID_RVN                   =  00000028
NEXT_DIR                           00000193 R      01
NOT_FND                            00000108 R      01
NXT_DIR                            0000011E R      01
PIO$A_DIRCACHE                     ********   X    01
PIO$A_TRACE                        ********   X    01
PIO$GL_DIRCACHE                    ********   X    01
PIO$GL_DIRCFRLH                    ********   X    01
PR$_IPL                            ********   X    01
PREFIX_O                           000002DD R      01
PRUNE                              000001F6 R      01
QIO$_ASTADR                     =  00000014
QIO$_ASTPRM                     =  00000018
QIO$_CHAN                       =  00000008
QIO$_EFN                        =  00000004
QIO$_FUNC                       =  0000000C
QIO$_IOSB                       =  00000010
QIO$_NARGS                      =  0000000C
QIO$_P1                         =  0000001C
QIO$_P2                         =  00000020
QIO$_P3                         =  00000024
QIO$_P4                         =  00000028
QIO$_P5                         =  0000002C
QIO$_P6                         =  00000030
RETURN                             00000045 R      01
RMS$ARM_DIRCACHE                   000002FF RG     01
RMS$BUG                            ********   X    01
RMS$CHKNAM                         ********   X    01
RMS$DIRCACHE_BLKAST                ********   X    01
RMS$FCPFNC                         ********   X    01
RMS$GETCCB                         000002F5 RG     01
RMS$MAPERR                         ********   X    01
RMS$SETDID                         00000006 RG     01
RMS$SETDID_ALT                     00000057 RG     01
RMS$_DIR                        =  000184CC
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
RM$RMSO	00003B9 (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
```

RM0XPFN
LIS

RM0SETDID
LIS

RM0SHARE
LIS

RM0WILD
LIS

RM0XAB
LIS

RM0STALL
LIS