```
FFFFFFFFFFFFFFF        111          111     XXX         XXX
FFFFFFFFFFFFFFF        111          111     XXX         XXX
FFFFFFFFFFFFFFF        111          111     XXX         XXX
FFF                 111111       111111     XXX         XXX
FFF                 111111       111111     XXX         XXX
FFF                 111111       111111     XXX         XXX
FFF                    111          111        XXX   XXX
FFF                    111          111        XXX   XXX
FFF                    111          111        XXX   XXX
FFFFFFFF.FFF           111          111           XXX
FFFFFFFFFFFF           111          111           XXX
FFFFFFFFFFFF           111          111           XXX
FFF                    111          111        XXX   XXX
FFF                    111          111        XXX   XXX
FFF                    111          111        XXX   XXX
FFF                    111          111     XXX         XXX
FFF                    111          111     XXX         XXX
FFF                    111          111     XXX         XXX
FFF                 111111111    111111111  XXX         XXX
FFF                 111111111    111111111  XXX         XXX
FFF                 111111111    111111111  XXX         XXX
```

```
_$25

Symb
----
IOC1
IO_C
IO_C
IO_D
IO_F
IO_S
KILL

KILL
KILL
LB_E
LB_D
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

LOCK
LOCK
LOCK
LOC_
LOC_
L_CO
L_CO
L_DA
L_DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAKc
MAP_
MAP_

MAP
MARK
MARK
MARK
MARK
MARK
```

```
**FILE**ID**CHARGEQ                         J  4                                                  CHA
                                                                                                  V04

  CCCCCCCC  HH      HH    AAAAAA    RRRRRRRR      GGGGGGGG  EEEEEEEEEE    QQQQQQ
  CCCCCCCC  HH      HH    AAAAAA    RRRRRRRR      GGGGGGGG  EEEEEEEEEE    QQQQQQ
CC          HH      HH  AA      AA  RR      RR  GG          EE            QQ      QQ
CC          HH      HH  AA      AA  RR      RR  GG          EE            QQ      QQ
CC          HH      HH  AA      AA  RR      RR  GG          EE            QQ      QQ
CC          HH      HH  AA      AA  RR      RR  GG          EE            QQ      QQ
CC          HHHHHHHHHH  AA      AA  RRRRRRRR    GG          EEEEEEEE      QQ      QQ
CC          HHHHHHHHHH  AA      AA  RRRRRRRR    GG          EEEEEEEE      QQ      QQ
CC          HH      HH  AAAAAAAAAA  RR  RR      GG  GGGGGG  EE            QQ  QQ  QQ
CC          HH      HH  AAAAAAAAAA  RR  RR      GG  GGGGGG  EE            QQ  QQ  QQ
CC          HH      HH  AA      AA  RR    RR    GG      GG  EE            QQ    QQ        ....
CC          HH      HH  AA      AA  RR    RR    GG      GG  EE            QQ    QQ        ....
  CCCCCCCC  HH      HH  AA      AA  RR      RR    GGGGGG    EEEEEEEEEE    QQQQ  QQ      ....
  CCCCCCCC  HH      HH  AA      AA  RR      RR    GGGGGG    EEEEEEEEEE    QQQQ  QQ      ....

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
```

```
0 MODULE CHARGEQ (
0                  LANGUAGE (BLISS32),
0                  IDENT = 'V04-000'
0                  ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  F11ACP Structure Level 2
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains the routines for charging disk blocks
1 !       against a particular quota file entry.
1 !
1 ! ENVIRONMENT:
1 !
1 !       STARLET operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  22-May-1979  20:51
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-012 CDS0004         Christian D. Saether    29-Aug-1984
1 !               Be prepared to find multiple headers when rebuilding
1 !               the quota file fcb.  Reread header for PRIMARY_FCB
1 !               when rebuilding the quota fcb if it is not the quota
1 !               file fcb.
1 !
```

```
1 !          V03-011 CDS0003          Christian D. Saether    23-Aug-1984
1 !                  Check quota fcb for staleness and rebuild if necessary.
1 !
1 !          V03-010 ACG0443          Andrew C. Goldstein,    21-Aug-1984  19:51
1 !                  Fix setup of REAL_Q_REC in file search so removal works
1 !                  on a cache miss.
1 !
1 !          V03-009 ACG0438          Andrew C. Goldstein,    19-Jul-1984  16:45
1 !                  Implement write access cache interlock
1 !
1 !          V03-008 ACG0430          Andrew C. Goldstein,    31-May-1984  15:07
1 !                  Fix reference to quota cache value block in REL_QUOTA_LOCK
1 !
1 !          V03-007 ACG0429          Andrew C. Goldstein,    21-May-1984  12:00
1 !                  Fix flow bug in ACG0428
1 !
1 !          V03-006 ACG0428          Andrew C. Goldstein,    18-May-1984  14:29
1 !                  Re-read quota record if value block not valid
1 !
1 !          V03-005 ACG0408          Andrew C. Goldstein,    23-Mar-1984  14:40
1 !                  Add AST parameter so that impure storage is fully based
1 !
1 !          V03-004 ACG0400          Andrew C. Goldstein,     1-Mar-1984  21:09
1 !                  Implement cluster-wide quota cacheing
1 !
1 !          V03-003 CDS0002          Christian D. Saether    29-Dec-1983
1 !                  Use L_NORM linkage and BIND_COMMON macro.
1 !
1 !          V03-002 CDS0001          Christian D. Saether     6-Dec-1983
1 !                  Serialize quota checking operations using allocation lock.
1 !
1 !          V03-001 ACG0337          Andrew C. Goldstein,    16-May-1983  16:04
1 !                  Fix handling of quota cache counters
1 !
1 !          V02-006 ACG0229          Andrew C. Goldstein,    23-Dec-1981  21:45
1 !                  Add counters for quota cache hits and misses
1 !
1 !          V02-005 ACG0167          Andrew C. Goldstein,    16-Apr-1980  19:25
1 !                  Previous revision history moved to F11B.REV
1 !**
1
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32';
1 REQUIRE 'SRC$:FCPDEF.B32';
1
1 FORWARD ROUTINE
1          CHARGE_QUOTA    : L_NORM NOVALUE, ! check and/or charge disk blocks
1          SEARCH_QUOTA    : L_NORM,         ! search for a quota file record
1          WRITE_QUOTA     : L_NORM NOVALUE, ! write back a quota record
1          SCAN_QUO_CACHE  : L_NORM,         ! search the quota cache
1          GET_QUOTA_LOCK  : L_NORM NOVALUE, ! acquire lock on quota file entry
1          REL_QUOTA_LOCK  : L_NORM NOVALUE, ! release lock on quota file entry
1          CLEAN_QUO_CACHE : L_NORM NOVALUE, ! write modified cache entry
1          ENTER_QUO_CACHE : L_NORM NOVALUE; ! make a new cache entry
```

```
1 GLOBAL ROUTINE CHARGE_QUOTA (UIC, BLOCK_COUNT, FLAGS) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine locates the quota file entry identified by the UIC
1 !       given and checks and/or charges the indicated number of blocks,
1 !       as specified by the flags.
1 !
1 ! CALLING SEQUENCE:
1 !       CHARGE_QUOTA (ARG1, ARG2, ARG3)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: UIC of entry to charge
1 !       ARG2: number of blocks to charge (negative to credit)
1 !       ARG3: bit encoded flags
1 !             bit 0 set to check if quota will be exceeded
1 !             bit 1 set to actually charge blocks to the quota entry
1 !
1 ! IMPLICIT INPUTS:
1 !       IO_PACKET: user's I/O packet
1 !       CURRENT_RVN: RVN of volume
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       quota file modified
1 !
1 !--
1
1
2 BEGIN
2
2 MAP
2       FLAGS           : BITVECTOR;    ! flags argument
2
2 LABEL
2       CHECK_QUOTA;                    ! block of code to check quota
2
2 LOCAL
2       SAVE_RVN,                       ! place to save current RVN
2       Q_RECORD        : REF BBLOCK;   ! address of quota file record
2
2 BIND_COMMON;
2
2 EXTERNAL ROUTINE
2       SWITCH_VOLUME   : L_NORM;       ! switch to desired RVN
2
2
```

```
2 ! Save the current RVN and then switch context to the root RVN.
2 ! First locate the quota file record. If there is no quota file enabled, this
2 ! routine is a NOP.
2 !
2
2 SAVE_RVN = .CURRENT_RVN;
2 SWITCH_VOLUME (1);
2
3 CHECK_QUOTA: BEGIN
3
3 Q_RECORD = SEARCH_QUOTA (.UIC, 0, 0, 1);
3 IF .Q_RECORD EQL -1 THEN LEAVE CHECK_QUOTA;
3
3 ! Check for quota exceeded if requested and the user does not have EXQUOTA
3 ! privilege. If we are to check, lack of a quota record is an error; if
3 ! we do not check, this routine is a NOP.
3 !
3
3 IF .FLAGS[QUOTA_CHECK]
3 AND NOT .BBLOCK [BBLOCK [.IO_PACKET[IRP$L_ARB], ARB$Q_PRIV], PRV$V_EXQUOTA]
3 THEN
4     BEGIN
4     IF .Q_RECORD EQL 0
4     THEN ERR_EXIT (SS$_EXDISKQUOTA);
4     IF .Q_RECORD[DQF$L_USAGE] + .BLOCK_COUNT GTRU .Q_RECORD[DQF$L_PERMQUOTA]
4     THEN
5         BEGIN
5         IF .CURRENT_WINDOW NEQ 0
5         THEN
6             BEGIN
6             IF .CURRENT_WINDOW[WCB$V_OVERDRAWN]
6             THEN
7                 BEGIN
7                 IF .Q_RECORD[DQF$L_USAGE] + .BLOCK_COUNT GTRU
7                    .Q_RECORD[DQF$L_PERMQUOTA] + .Q_RECORD[DQF$L_OVERDRAFT]
8                 THEN ERR_EXIT (SS$_EXDISKQUOTA)
7                 ELSE ERR_STATUS (SS$_OVRDSKQUOTA);
7                 END
6             ELSE
7                 BEGIN
7                 CURRENT_WINDOW[WCB$V_OVERDRAWN] = 1;
7                 ERR_EXIT (SS$_EXDISKQUOTA);
6                 END;
6             END
5         ELSE
5             ERR_EXIT (SS$_EXDISKQUOTA);
4         END;
4     END
4
3 ELSE
3     IF .Q_RECORD EQL 0 THEN LEAVE CHECK_QUOTA;
3
3 ! If the record is to be charged, do so. Check the result to see if it
3 ! is negative; if so, zero it to prevent absurd results.
3 !
3
3 IF .FLAGS[QUOTA_CHARGE]
```

```
;  227          1216  3 THEN
;  228          1217  4     BEGIN
;  229          1218  4     Q_RECORD[DQF$L_USAGE] = .Q_RECORD[DQF$L_USAGE] + .BLOCK_COUNT;
;  230          1219  4     IF .Q_RECORD[DQF$L_USAGE] LSS 0
;  231          1220  4     THEN Q_RECORD[DQF$L_USAGE] = 0;
;  232          1221  4     WRITE_QUOTA (.Q_RECORD);
;  233          1222  3     END;
;  234          1223  3
;  235          1224  2 END;                                    ! end of block CHECK_QUOTA
;  236          1225  2
;  237          1226  2 SWITCH_VOLUME (.SAVE_RVN);
;  238          1227  2
;  239          1228  1 END;                                    ! end of routine CHARGE_QUOTA


                                                                      .TITLE  CHARGEQ
                                                                      .IDENT  \V04-000\

                                                                      .EXTRN  SWITCH_VOLUME

                                                                      .PSECT  $CODE$,NOWRT,2

                                              000C 00000              .ENTRY  CHARGE_QUOTA, Save R2,R3                         ; 1102
                          53        A0     AA  D0 00002               MOVL    -96(BASE), SAVE_RVN                              ; 1164
                                           01  DD 00006               PUSHL   #1                                               ; 1165
                   0000G  CF               01  FB 00008               CALLS   #1, SWITCH_VOLUME
                                           01  DD 0000D               PUSHL   #1                                               ; 1169
                                           7E  7C 0000F               CLRQ    -(SP)
                                    04     AC  DD 00011               PUSHL   UIC
                   0000V  CF               04  FB 00014               CALLS   #4, SEARCH_QUOTA
                FFFFFFFF  8F               50  D1 00019               CMPL    Q_RECORD, #-1                                    ; 1170
                                           62  13 00020               BEQL    6$
                          44        0C     AC  E9 00022               BLBC    FLAGS, 3$                                        ; 1177
                          51        90     AA  D0 00026               MOVL    -112(BASE), R1                                   ; 1178
         3B        58     B1               13  E0 0002A               BBS     #19, @88(R1), 3$
                                           50  D5 0002F               TSTL    Q_RECORD                                         ; 1181
                                           32  13 00031               BEQL    2$
         52        08     A0        08     AC  C1 00033               ADDL3   BLOCK_COUNT, 8(Q_RECORD), R2                     ; 1183
                   0C     A0               52  D1 00039               CMPL    R2, 12(Q_RECORD)
                                           2F  1B 0003D               BLEQU   4$
                          51        0C     AA  D0 0003F               MOVL    12(BASE), R1                                     ; 1186
                                           20  13 00043               BEQL    2$
         17        0B     A1               04  E1 00045               BBC     #4, 11(R1), 1$                                   ; 1189
         51        0C     A0        10     A0  C1 0004A               ADDL3   16(Q_RECORD), 12(Q_RECORD), R1                   ; 1193
                          51               52  D1 00050               CMPL    R2, R1
                                           10  1A 00053               BGTRU   2$
                          15        80     AA  E9 00055               BLBC    -1 3(BASE), 4$                                   ; 1195
                   80     AA      0669     8F  B0 00059               MOVW    #1641, -128(BASE)
                                           0D  11 0005F               BRB     4$                                               ; 1189
                   0B     A1               10  88 00061 1$:           BISB2   #16, 11(R1)                                      ; 1199
                                  03EC     8F  BF 00065 2$:           CHMU    #1004                                            ; 1204
                                               04 00069               RET
                                           50  D5 0006A 3$:           TSTL    Q_RECORD                                         ; 1209
                                           16  13 0006C               BEQL    6$
         11        0C     AC               01  E1 0006E 4$:           BBC     #1, FLAGS, 6$                                    ; 1215
                   08     A0        08     AC  C0 00073               ADDL2   BLOCK_COUNT, 8(Q_RECORD)                         ; 1218
                                           03  18 00078               BGEQ    5$                                               ; 1219
                          08   A0  D4 0007A          CLRL    8(Q_RECORD)                    ; 1220
                               50  DD 0007D 5$:      PUSHL   Q_RECORD                       ; 1221
             0000V   CF        01  FB 0007F          CALLS   #T, WRITE_QUOTA                ;
                               53  DD 00084 6$:      PUSHL   SAVE_RVN                       ; 1226
             0000G   CF        01  FB 00086          CALLS   #1, SWITCH_VOLUME              ;
                               04  0008B             RET                                    ; 1228

; Routine Size:  140 bytes,    Routine Base:  $CODE$ + 0000
```

```
GLOBAL ROUTINE SEARCH_QUOTA (UIC, FLAGS, START_REC, USE_CACHE) : L_NORM =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine searches the quota file for the specified UIC under
!       control of the match flags.
!
! CALLING SEQUENCE:
!       SEARCH_QUOTA (ARG1, ARG2, ARG3, ARG4)
!
! INPUT PARAMETERS:
!       ARG1: UIC to search for
!       ARG2: match control flags from FIB
!       ARG3: record number at which to start
!       ARG4: 1 to find record in the cache
!             0 to unconditionally go to the quota file
!
! IMPLICIT INPUTS:
!       CURRENT_VCB: address of volume's VCB
!       context set to RVN 1 of volume set
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       QUOTA_RECORD: record number of found record
!       FREE_QUOTA: record number of first free quota file entry
!       QUOTA_INDEX: cache index of cache entry if found
!       DUMMY_REC: filled in with cache contents if found
!
! ROUTINE VALUE:
!       address of quota file record found, 0 if none, -1 if not quota file
!
! SIDE EFFECTS:
!       quota file read, contents of buffer cache altered
!
!--

BEGIN

MAP
        FLAGS           : BITVECTOR;    ! match control flags

LITERAL
        ALL_GROUP       = $BITPOSITION (FIB$V_ALL_GRP),
        ALL_MEMBER      = $BITPOSITION (FIB$V_ALL_MEM),
        RECS_PER_BLOCK  = 512 / DQF$C_LENGTH;

LABEL
        QUOTA_SCAN;                     ! search quota file

LOCAL
        FCB             : REF BBLOCK,   ! address of quota file FCB
        QUOTA_CACHE     : REF BBLOCK,   ! address of quota cache
        QUOTA_LIST      : REF BBLOCKVECTOR [,VCA$C_QUOLENGTH],
2                                               ! address of quota cache entries
2           J,                                  ! index into quota cache
2           REC_NUM,                            ! quota file record to read
2           FIRST_REC,                          ! first record in block to use
2           VBN,                                ! block number of quota file
2           Q_RECORD        : REF BBLOCK;       ! address of quota file record
2
2   BIND_COMMON;
2
2   EXTERNAL ROUTINE
2           ALLOCATION_LOCK : L_NORM,           ! allocation lock serialization
2           BUILD_EXT_FCBS  : L_NORM NOVALUE,   ! build extension fcb chain.
2           SERIAL_FILE     : L_NORM,           ! get serialization lock
2           REBLD_PRIM_FCB  : L_NORM NOVALUE,   ! rebuild fcb from header
2           READ_HEADER     : L_NORM,           ! read file header
2           RELEASE_SERIAL_LOCK : L_NORM NOVALUE, ! release serialization lock
2           READ_BLOCK      : L_NORM;           ! read a disk block
2
2
2   ! Get the FCB address for the quota file. If none, take an error exit.
2   !
2
2   FCB = .CURRENT_VCB[VCB$L_QUOTAFCB];
2   IF .FCB EQL 0 THEN RETURN -1;
2
2   ! Serialize quota search using allocation lock.
2   !
2
2   ALLOCATION_LOCK ();
2
2   ! Check to see if the quota fcb is stale, that is, it has been modified
2   ! on another node.  If so, serialize on the quota file itself, read
2   ! the header and rebuild the fcb.
2   !
2
2   IF .FCB [FCB$V_STALE]
2   THEN
3       BEGIN
3       LOCAL
3           HEADER,
3           SAV_CURRLCKINDX;
3
3       SAV_CURRLCKINDX = .CURR_LCKINDX;
3
3       SERIAL_FILE (FCB [FCB$W_FID]);
3
3   ! Setting this flag prevents READ_HEADER from modifying FILE_HEADER.
3   ! It is also set when the BUILD_EXT_FCBS routine calls READ_HEADER
3   ! if BUILD_EXT_FCBS is called with the optional fcb argument.
3   !
3
3       STSFLGS [STS_LEAVE_FILEHDR] = 1;
3
3       HEADER = READ_HEADER (0, .FCB);
3
3       REBLD_PRIM_FCB (.FCB, .HEADER);
3
```

```
3     BUILD_EXT_FCBS (.HEADER, .FCB);
3
3     IF .SAV_CURRLCKINDX NEQ .CURR_LCKINDX
3     THEN
4         BEGIN
4         RELEASE_SERIAL_LOCK (.CURR_LCKINDX);
4         CURR_LCKINDX = .SAV_CURRLCKINDX;
3         END;
3
3 ! If PRIMARY_FCB is nonzero and not the quota file fcb, and also if
3 ! it is the same lockbasis as the current lock index, then reread
3 ! the header for it to re-establish FILE_HEADER.
3 !
3
3     IF .PRIMARY_FCB NEQ 0
3     THEN
3         IF .PRIMARY_FCB NEQ .FCB
3            AND .PRIMARY_FCB [FCB$L_LOCKBASIS] EQL .LB_BASIS [.CURR_LCKINDX]
3         THEN
3             READ_HEADER (0, .PRIMARY_FCB);
3
2     END;
2
2 ! If there are no wild cards in the search, scan the quota cache first.
2 ! If the value block was lost, the cache entry comes back not valid,
2 ! but with contents. In this case, and if this is a write-through operation,
2 ! use the record number in the cache to read the record to save the search.
2 ! As long as the record is read, also update it if the cache entry is dirty.
2 !
2
2 REAL_Q_REC = 0;
2 QUOTA_CACHE = .CURRENT_VCB[VCB$L_QUOCACHE];
2 QUOTA_LIST = QUOTA_CACHE[VCA$L_QUOLIST];
2 IF NOT .FLAGS[ALL_MEMBER] AND NOT .FLAGS[ALL_GROUP]
2 THEN
3     BEGIN
3     J = SCAN_QUO_CACHE (.UIC, .USE_CACHE);
3     IF .QUOTA_LIST[.J-1, VCA$L_QUORECNUM] NEQ 0
3     THEN
4         BEGIN
4         IF NOT .USE_CACHE
4         OR NOT .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
4         OR NOT .QUOTA_CACHE[VCA$V_CACHEVALID]
4         THEN
5             BEGIN
5             QUOTA_RECORD = .QUOTA_LIST[.J-1, VCA$L_QUORECNUM];
5             REC_NUM = .QUOTA_LIST[.J-1, VCA$L_QUORECNUM] - 1;
5             REAL_Q_REC = READ_BLOCK (.REC_NUM / RECS_PER_BLOCK
5                                      + .FCB[FCB$L_STLBN], 1, QUOTA_TYPE)
5                          + (.REC_NUM MOD RECS_PER_BLOCK) * DQF$C_LENGTH;
5             IF .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
5             THEN
5                 CLEAN_QUO_CACHE (.J, .REAL_Q_REC)
5             ELSE
6                 BEGIN
6                 ENTER_QUO_CACHE (.J, .REAL_Q_REC, 0, .USE_CACHE);
6                 CH$MOVE (DQF$C_LENGTH, .REAL_Q_REC, DUMMY_REC);
```

```
5                   END;
4               END;
4           RETURN DUMMY_REC;
3           END;
2       END;
2
2   ! We couldn't find a valid cache entry (either because it's not there
2   ! or the operation won't allow it). Scan the blocks of the quota file,
2   ! looking for a matching record.
2   !
2
3   QUOTA_SCAN: BEGIN
3
3   FIRST_REC = .START_REC MOD RECS_PER_BLOCK;
3   INCR VBN FROM .START_REC/RECS_PER_BLOCK TO .FCB[FCB$L_EFBLK] - 1
3   DO
4       BEGIN
4       Q_RECORD = READ_BLOCK (.VBN + .FCB[FCB$L_STLBN],
4                              .FCB[FCB$L_EFBLK] - .VBN,
4                              QUOTA_TYPE)
4                + .FIRST_REC * DQF$C_LENGTH;
4
4       INCR J FROM .FIRST_REC TO RECS_PER_BLOCK - 1
4       DO
5           BEGIN
5           QUOTA_RECORD = .VBN * RECS_PER_BLOCK + .J + 1;
5
5           IF .Q_RECORD[DQF$V_ACTIVE]
5           THEN
6               BEGIN
7               IF  (.FLAGS[ALL_MEMBER] OR .UIC<00,16> EQL .(Q_RECORD[DQF$L_UIC])<00,16>)
7               AND (.FLAGS[ALL_GROUP]  OR .UIC<16,16> EQL .(Q_RECORD[DQF$L_UIC])<16,16>)
6               THEN LEAVE QUOTA_SCAN;
6               END
6
5           ELSE
6               BEGIN
6               IF .FREE_QUOTA EQL 0
6               THEN FREE_QUOTA = .QUOTA_RECORD;
5               END;
5
5           Q_RECORD = .Q_RECORD + DQF$C_LENGTH;
4           END;                                ! end of inner loop
4
4       FIRST_REC = 0;
3       END;                                    ! end of block scan loop
3
3   IF NOT .FLAGS[ALL_MEMBER] AND NOT .FLAGS[ALL_GROUP]
3   THEN REL_QUOTA_LOCK (.J);
3   RETURN 0;                                   ! return 0 if not found
2   END;                                        ! end of block QUOTA_SCAN
2
2   ! We have found a record in the quota file. If there were wild cards, now
2   ! scan the quota cache to see if an entry is present. With wild cards, the
2   ! file must be scanned first to be able to return the entries in a coherent
2   ! order; yet we must look at the cache in case a modified entry is present.
2   !
```

```
;  469          1457  2
;  470          1458  2 REAL_Q_REC = .Q_RECORD;
;  471          1459  2 IF .FLAGS[ALL_MEMBER] OR .FLAGS[ALL_GROUP]
;  472          1460  2 THEN
;  473          1461  3     BEGIN
;  474          1462  3     J = SCAN_QUO_CACHE (.Q_RECORD[DQF$L_UIC], 0);
;  475          1463  3     IF .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  476          1464  3     THEN
;  477          1465  4         BEGIN
;  478          1466  4         CLEAN_QUO_CACHE (.J, .Q_RECORD);
;  479          1467  4         RETURN DUMMY_REC;
;  480          1468  3         END;
;  481          1469  2     END;
;  482          1470  2
;  483          1471  2 ! Finally enter the new record into the quota cache.
;  484          1472  2 !
;  485          1473  2
;  486          1474  2 ENTER_QUO_CACHE (.J, .Q_RECORD, 0, .USE_CACHE);
;  487          1475  2 CH$MOVE (DQF$C_LENGTH, .Q_RECORD, DUMMY_REC);
;  488          1476  2 RETURN DUMMY_REC;
;  489          1477  2
;  490          1478  1 END;                                   ! end of routine SEARCH_QUOTA


                                                            .EXTRN  ALLOCATION_LOCK
                                                            .EXTRN  BUILD_EXT_FCBS, SERIAL_FILE
                                                            .EXTRN  REBLD_PRIM_FCB, READ_HEADER
                                                            .EXTRN  RELEASE_SERIAL_LOCK
                                                            .EXTRN  READ_BLOCK

                                   0BFC 00000               .ENTRY  SEARCH_QUOTA, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 1229
                                                                    R11                                          ;
                          02B4  CA  9F 00002                PUSHAB  692(BASE)                                    ; 1291
                59        02BC  CA  9E 00006                MOVAB   700(BASE), R9                                ;
                5B        02C4  CA  9E 0000B                MOVAB   708(BASE), R11                               ;
                50          98  AA  D0 00010                MOVL    -104(BASE), R0                               ; 1308
                57          54  A0  D0 00014                MOVL    84(R0), FCB                                  ;
                            04  12 00018                    BNEQ    1$                                           ; 1309
                50              01  CE 0001A                MNEGL   #1, R0                                       ;
                                    04 0001D                RET
        0000G   CF              00  FB 0001E 1$:            CALLS   #0, ALLOCATION_LOCK                          ; 1314
                61          23  A7  E9 00023                BLBC    35(FCB), 3$                                  ; 1321
                52          14  AA  D0 00027                MOVL    20(BASE), SAV_CURRLCKINDX                    ; 1328
                            24  A7  9F 0002B                PUSHAB  36(FCB)                                      ; 1330
        0000G   CF              01  FB 0002E                CALLS   #1, SERIAL_FILE                              ;
           A6   AA              08  88 00033                BISB2   #8, -90(BASE)                                ; 1337
                                57  DD 00037                PUSHL   FCB                                          ; 1339
                                7E  D4 00039                CLRL    -(SP)                                        ;
        0000G   CF              02  FB 0003B                CALLS   #2, READ_HEADER                              ;
                53              50  D0 00040                MOVL    R0, HEADER                                   ;
                                53  DD 00043                PUSHL   HEADER                                       ; 1341
                                57  DD 00045                PUSHL   FCB                                          ;
        0000G   CF              02  FB 00047                CALLS   #2, REBLD_PRIM_FCB                           ;
                          0088  8F  BB 0004C                PUSHR   #^M<R3,R7>                                   ; 1343
        0000G   CF              02  FB 00050                CALLS   #2, BUILD_EXT_FCBS                           ;
           14   AA              52  D1 00055                CMPL    SAV_CURRLCKINDX, 20(BASE)                    ; 1345
```

```
                                            0C  13 00059        BEQL    2$                                            ;
                                    14      AA  DD 0005B        PUSHL   20(BASE)                                      ; 1348
                        0000G  CF           01  FB 0005E        CALLS   #1, RELEASE_SERIAL_LOCK                       ;
                           14  AA           52  D0 00063        MOVL    SAV_CURRLCKINDX, 20(BASE)                     ; 1349
                               51   08      AA  D0 00067 2$:    MOVL    8(BASE), R1                                   ; 1357
                                            1B  13 0006B        BEQL    3$                                            ;
                               57           51  D1 0006D        CMPL    R1, FCB                                       ; 1359
                                            16  13 00070        BEQL    3$                                            ;
                               50   14      AA  D0 00072        MOVL    20(BASE), R0                                  ; 1360
                      0080 CA40     4C      A1  D1 00076        CMPL    76(R1), 128(BASE)[R0]                         ;
                                            09  12 0007D        BNEQ    3$                                            ;
                                            51  DD 0007F        PUSHL   R1                                            ; 1362
                                            7E  D4 00081        CLRL    -(SP)                                         ;
                        0000G  CF           02  FB 00083        CALLS   #2, READ_HEADER                               ;
                                            69  D4 00088 3$:    CLRL    (R9)                                          ; 1373
                               50   98      AA  D0 0008A        MOVL    -104(BASE), R0                                ; 1374
                               52   5C      A0  D0 0008E        MOVL    92(R0), QUOTA_CACHE                           ;
                               56   44      A2  9E 00092        MOVAB   68(R2), QUOTA_LIST                            ; 1375
                               03   08      AC  E9 00096        BLBC    FLAGS, 5$                                     ; 1376
                                          008F  31 0009A 4$:    BRW     10$                                           ;
                  F8        08 AC           01  E0 0009D 5$:    BBS     #1, FLAGS, 4$                                 ;
                                    10      AC  DD 000A2        PUSHL   USE_CACHE                                     ; 1379
                                    04      AC  DD 000A5        PUSHL   UIC                                           ;
                        0000V  CF           02  FB 000A8        CALLS   #2, SCAN_QUO_CACHE                            ;
                               58           50  D0 000AD        MOVL    R0, J                                         ;
                  50           58           1C  C5 000B0        MULL3   #28, J, R0                                    ; 1380
                               50           56  C0 000B4        ADDL2   QUOTA_LIST, R0                                ;
        00         EC   A0     18           00  ED 000B7        CMPZV   #0, #24, -20(R0), #0                          ;
                                            6D  13 000BD        BEQL    10$                                           ;
                               08   10      AC  E9 000BF        BLBC    USE_CACHE, 6$                                 ; 1383
                               04   EF      A0  E9 000C3        BLBC    -17(R0), 6$                                   ; 1384
                               5E   0B      A2  E8 000C7        BLBS    11(QUOTA_CACHE), 9$                           ; 1385
     00    BE      EC   A0     18           00  EF 000CB 6$:    EXTZV   #0, #24, -20(R0), @0(SP)                      ; 1388
                        50     58           1C  C5 000D2        MULL3   #28, J, R0                                    ; 1389
        52         EC A046     18           00  EF 000D6        EXTZV   #0, #24, -20(R0)[QUOTA_LIST], REC_NUM         ;
                                            52  D7 000DD        DECL    REC_NUM                                       ;
                                            05  DD 000DF        PUSHL   #5                                            ; 1390
                                            01  DD 000E1        PUSHL   #1                                            ;
                  50           52           10  C7 000E3        DIVL3   #16, REC_NUM, R0                              ;
                                  30 B740   9F  000E7        PUSHAB  @48(FCB)[R0]                                  ; 1391
                        0000G  CF           03  FB 000EB        CALLS   #3, READ_BLOCK                                ;
        7E                  00 52           01  7A 000F0        EMUL    #1, REC_NUM, #0, -(SP)                        ; 1392
        51                  51 8E           10  7B 000F5        EDIV    #16, (SP)+, R1, R1                            ;
                               51           20  C4 000FA        MULL2   #32, R1                                       ;
                  69           50           51  C1 000FD        ADDL3   R1, R0, (R9)                                  ;
                  50           58           1C  C5 00101        MULL3   #28, J, R0                                    ; 1393
                  0B       EF A046          00  E1 00105        BBC     #0, -17(R0)[QUOTA_LIST], 8$                   ;
                                            69  DD 0010B        PUSHL   (R9)                                          ; 1395
                                            58  DD 0010D 7$:    PUSHL   J                                             ;
                        0000V  CF           02  FB 0010F        CALLS   #2, CLEAN_QUO_CACHE                           ;
                                            13  11 00114        BRB     9$                                            ;
                                    10      AC  DD 00116 8$:    PUSHL   USE_CACHE                                     ; 1398
                                            7E  D4 00119        CLRL    -(SP)                                         ;
                                            69  DD 0011B        PUSHL   (R9)                                          ;
                                            58  DD 0011D        PUSHL   J                                             ;
                        0000V  CF           04  FB 0011F        CALLS   #4, ENTER_QUO_CACHE                           ;
                  6B        00 B9           20  28 00124        MOVC3   #32, @0(R9), (R11)                            ; 1399
```

```
                                         00C1  31 00129 9$:      BRW     21$                                     ; 1402
 7E        00    0C  AC                    01  7A 0012C 10$:     EMUL    #1, START_REC, #0, -(SP)                ; 1413
 53        53        8E                    10  7B 00132          EDIV    #16, (SP)+, FIRST_REC, FIRST_REC        ;
           52    0C  AC                    10  C7 00137          DIVL3   #16, START_REC, R2                      ; 1414
                     55           3C       A7  D0 0013C          MOVL    60(FCB), R5                             ;
                                           52  D7 00140          DECL    VBN                                     ;
                                           59  11 00142          BRB     17$                                     ;
                                           05  DD 00144 11$:     PUSHL   #5                                      ; 1417
           7E        3C           A7       52  C3 00146          SUBL3   VBN, 60(FCB), -(SP)                     ; 1418
                               30 B742     9F  0014B             PUSHAB  @48(FCB)[VBN]                           ; 1417
                 0000G  CF                 03  FB 0014F          CALLS   #3, READ_BLOCK                          ;
           51        53                    05  78 00154          ASHL    #5, FIRST_REC, R1                       ; 1420
           54        50                    51  C1 00158          ADDL3   R1, R0, Q_RECORD                        ;
                     50           FF       A3  9E 0015C          MOVAB   -1(R3), J                               ; 1422
                                           35  11 00160          BRB     16$                                     ;
           51        52                    04  78 00162 12$:     ASHL    #4, VBN, R1                             ; 1425
                 00  BE          01 A041   9E  00166             MOVAB   1(J)[R1], @0(SP)                        ;
                     19                    64  E9 0016C          BLBC    (Q_RECORD), 14$                         ; 1427
                     07           08       AC  E8 0016F          BLBS    FLAGS, 13$                              ; 1430
                 04  A4           04       AC  B1 00173          CMPW    UIC, 4(Q_RECORD)                        ;
                                           1A  12 00178          BNEQ    15$                                     ;
           34    08  AC                    01  E0 0017A 13$:     BBS     #1, FLAGS, 18$                          ; 1431
                 06  A4           06       AC  B1 0017F          CMPW    UIC+2, 6(Q_RECORD)                      ;
                                           0E  12 00184          BNEQ    15$                                     ;
                                           2B  11 00186          BRB     18$                                     ; 1432
                                02B8       CA  D5 00188 14$:     TSTL    696(BASE)                               ; 1437
                                           06  12 0018C          BNEQ    15$                                     ;
              02B8  CA            00       BE  D0 0018E          MOVL    @0(SP), 696(BASE)                       ; 1438
                     54                    20  C0 00194 15$:     ADDL2   #32, Q_RECORD                           ; 1441
           C7        50                    0F  F3 00197 16$:     AOBLEQ  #15, J, 12$                             ; 1422
                                           53  D4 0019B          CLRL    FIRST_REC                               ; 1444
           A3        52                    55  F2 0019D 17$:     AOBLSS  R5, VBN, 11$                            ; 1414
                     4C           08       AC  E8 001A1          BLBS    FLAGS, 22$                              ; 1447
           47    08  AC                    01  E0 001A5          BBS     #1, FLAGS, 22$                          ;
                                           58  DD 001AA          PUSHL   J                                       ; 1448
                 0000V  CF                 01  FB 001AC          CALLS   #1, REL_QUOTA_LOCK                      ;
                                           3E  11 001B1          BRB     22$                                     ; 1449
                     69                    54  D0 001B3 18$:     MOVL    Q_RECORD, (R9)                          ; 1458
                     05           08       AC  E8 001B6          BLBS    FLAGS, 19$                              ; 1459
           1C    08  AC                    01  E1 001BA          BBC     #1, FLAGS, 20$                          ;
                                           7E  D4 001BF 19$:     CLRL    -(SP)                                   ; 1462
                                  04       A4  DD 001C1          PUSHL   4(Q_RECORD)                             ;
                 0000V  CF                 02  FB 001C4          CALLS   #2, SCAN_QUO_CACHE                      ;
                     58                    50  D0 001C9          MOVL    R0, J                                   ;
           50        58                    1C  C5 001CC          MULL3   #28, J, R0                              ; 1463
           05     EF A046                  00  E1 001D0          BBC     #0, -17(R0)[QUOTA_LIST], 20$            ;
                                           54  DD 001D6          PUSHL   Q_RECORD                                ; 1466
                                         FF32  31 001D8          BRW     7$                                      ;
                                  10       AC  DD 001DB 20$:     PUSHL   USE_CACHE                               ; 1474
                                           7E  D4 001DE          CLRL    -(SP)                                   ;
                                           54  DD 001E0          PUSHL   Q_RECORD                                ;
                                           58  DD 001E2          PUSHL   J                                       ;
                 0000V  CF                 04  FB 001E4          CALLS   #4, ENTER_QUO_CACHE                     ;
           6B        64                    20  28 001E9          MOVC3   #32, (Q_RECORD), (R11)                  ; 1475
                     50                    5B  D0 001ED 21$:     MOVL    R11, R0                                 ; 1476
                                               04 001F0          RET                                             ;
                                           50  D4 001F1 22$:     CLRL    R0                                      ; 1478
```

```
                                                            04 001F3          RET                                                       ;

; Routine Size:  500 bytes,    Routine Base:  $CODE$ + 008C
```

```
1 GLOBAL ROUTINE WRITE_QUOTA (Q_RECORD) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine writes the indicated quota record. If a cache entry
1 !       exists for the record being processed (indicated by the record
1 !       being the dummy record), we update the cache entry. If we also
1 !       have the real quota record in memory, then mark it for write-back.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       WRITE_QUOTA (ARG1)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of quota record
1 !
1 ! IMPLICIT INPUTS:
1 !       REAL_Q_REC: buffer of real quota record if exists
1 !       QUOTA_INDEX: cache index of cache entry
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       quota cache modified, quota record marked for write-back
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       Q_RECORD        : REF BBLOCK;   ! address of quota record
2
2 BIND_COMMON;
2
2 EXTERNAL ROUTINE
2       MARK_DIRTY      : L_NORM;       ! mark buffer for write back
2
2
2 ! If the specified record is the dummy record, there is a cache entry.
2 ! Therefore, update it. Also update the associated real record if there
2 ! is one.
2 !
2
2 IF .Q_RECORD EQL DUMMY_REC
2 THEN
3     BEGIN
3     ENTER_QUO_CACHE (.QUOTA_INDEX, .Q_RECORD, .REAL_Q_REC EQL 0, 2);
3     IF .REAL_Q_REC NEQ 0
```

```
;  549        1536  3       THEN
;  550        1537  4           BEGIN
;  551        1538  4           CH$MOVE (DQF$C_LENGTH, .Q_RECORD, .REAL_Q_REC);
;  552        1539  4           MARK_DIRTY (.REAL_Q_REC);
;  553        1540  3           END;
;  554        1541  3       END
;  555        1542  3
;  556        1543  3 ! Otherwise, if there is no cache entry, we just have to mark the
;  557        1544  3 ! buffer dirty.
;  558        1545  3 !
;  559        1546  3
;  560        1547  2 ELSE
;  561        1548  2     MARK_DIRTY (.Q_RECORD);
;  562        1549  2
;  563        1550  1 END;                                   ! end of routine WRITE_QUOTA


                                                             .EXTRN  MARK_DIRTY

                                              007C 00000     .ENTRY  WRITE_QUOTA, Save R2,R3,R4,R5,R6             ; 1479
                           56       02BC   CA  9E 00002      MOVAB   700(BASE), R6                                ; 1518
                           50       02C4   CA  9E 00007      MOVAB   708(BASE), R0                                ; 1531
                           50         04   AC  D1 0000C      CMPL    Q_RECORD, R0                                 ;
                                           24  12 00010      BNEQ    2$                                           ;
                                           02  DD 00012      PUSHL   #2                                           ; 1534
                                           7E  D4 00014      CLRL    -(SP)                                        ;
                                           66  D5 00016      TSTL    (R6)                                         ;
                                           02  12 00018      BNEQ    1$                                           ;
                                           6E  D6 0001A      INCL    (SP)                                         ;
                                      04   AC  DD 0001C 1$:  PUSHL   Q_RECORD                                     ;
                                    02C0   CA  DD 0001F      PUSHL   704(BASE)                                    ;
                      0000V   CF               04  FB 00023  CALLS   #4, ENTER_QUO_CACHE                          ;
                                           66  D5 00028      TSTL    (R6)                                         ; 1535
                                           12  13 0002A      BEQL    4$                                           ;
          00    B6       04   BC           20  28 0002C      MOVC3   #32, @Q_RECORD, @0(R6)                       ; 1538
                                           66  DD 00032      PUSHL   (R6)                                         ; 1539
                                           03  11 00034      BRB     3$                                           ;
                                      04   AC  DD 00036 2$:  PUSHL   Q_RECORD                                     ; 1548
                      0000G   CF           01  FB 00039 3$:  CALLS   #1, MARK_DIRTY                               ;
                                           04  0003E 4$:     RET                                                  ; 1550

; Routine Size:  63 bytes,    Routine Base:  $CODE$ + 0280
```

```
1 ROUTINE SCAN_QUO_CACHE (UIC, MARK_USE) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine scans the quota cache for the indicated UIC. If found,
1 !       it returns the contents, and marks the entry used if requested.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       SCAN_QUO_CACHE (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: UIC to search for
1 !       ARG2: 1 to record new use
1 !             0 to not
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       DUMMY_REC: receives contents of cache entry if found
1 !       QUOTA_INDEX: receives index of cache entry found
1 !       QUOTA_RECORD: quota file record number of found entry
1 !
1 ! ROUTINE VALUE:
1 !       index of entry found
1 !
1 ! SIDE EFFECTS:
1 !       quota cache entry modified
1 !
1 !--
1
2 BEGIN
2
2 LITERAL
2       RECS_PER_BLOCK  = 512 / DQF$C_LENGTH;
2
2 LABEL
2       QUOTA_SEARCH;                   ! body of search code
2
2 LOCAL
2       QUOTA_CACHE     : REF BBLOCK,   ! address of quota cache
2       QUOTA_LIST      : REF BBLOCKVECTOR [,VCA$C_QUOLENGTH],
2                                       ! address of quota cache entries
2       J,                              ! index into quota cache
2       LOWEST_LRU,                     ! oldest quota LRU index
2       LOWEST_J,                       ! oldest quota cache entry index
2       LRU_DELTA,                      ! LRU index of current entry
2       OLD_RECORD      : REF BBLOCK,   ! address of old quota record
2       REC_NUM,                        ! quota file record to read
2       FCB             : REF BBLOCK;   ! address of quota file FCB
2
```

```
2 EXTERNAL
2         PMS$GL_QUOHIT    : ADDRESSING_MODE (GENERAL),
2                                            ! count of quota cache hits
2         PMS$GL_QUOMISS   : ADDRESSING_MODE (GENERAL);
2                                            ! count of quota cache misses
2
2 EXTERNAL ROUTINE
2         CACHE_LOCK       : L_NORM,         ! acquire cache lock
2         READ_BLOCK       : L_NORM;         ! read a disk block
2
2
2 BIND_COMMON;
2
2
2 ! If the cache is not currently marked valid, do so if possible.
2 ! This involves taking out the cache lock if the volume is cluster
2 ! accessible, and checking for quota file writers and a non-null
2 ! cache size.
2 !
2
2 QUOTA_CACHE = .CURRENT_VCB[VCB$L_QUOCACHE];
2 FCB = .CURRENT_VCB[VCB$L_QUOTAFCB];
2
2 IF NOT .QUOTA_CACHE[VCA$V_CACHEVALID]
2 AND NOT .QUOTA_CACHE[VCA$V_CACHEFLUSH]
2 THEN
3     BEGIN
3     IF   .QUOTA_CACHE[VCA$W_QUOSIZE] GTRU 1
3     AND NOT .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_DMT]
3     AND .FCB[FCB$W_WCNT] LEQU 1
3     AND
4         BEGIN
4         IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
4         THEN CACHE_LOCK (.FCB[FCB$L_LOCKBASIS], QUOTA_CACHE[VCA$L_QUOCLKID], 0)
4         ELSE 1
4         END
3     THEN
3         QUOTA_CACHE[VCA$V_CACHEVALID] = 1
3     ELSE
3         QUOTA_CACHE[VCA$V_CACHEFLUSH] = 1;
2     END;
2
2 ! Search the quota cache for an active entry with a matching UIC.
2 !
2
3 QUOTA_SEARCH: BEGIN
3
3 QUOTA_LIST = QUOTA_CACHE[VCA$L_QUOLIST];
3 INCR K FROM 1 TO .QUOTA_CACHE[VCA$W_QUOSIZE]
3 DO
4     BEGIN
4     IF   .QUOTA_LIST[.K-1, VCA$L_QUORECNUM] NEQ 0
4     AND .QUOTA_LIST[.K-1, VCA$L_QUOUIC] EQL .UIC
4     THEN
5         BEGIN
5         IF .MARK_USE
5         THEN
```

```
6                   BEGIN
6                   QUOTA_LIST[.K-1, VCA$W_QUOLRUX] = .QUOTA_CACHE[VCA$W_QUOLRU];
6                   QUOTA_CACHE[VCA$W_QUOLRU] = .QUOTA_CACHE[VCA$W_QUOLRU] + 1;
5                   END;
5               PMS$GL_QUOHIT = .PMS$GL_QUOHIT + 1;
5               J = .K;
5               LEAVE QUOTA_SEARCH;
4               END;
3           END;
3
3   ! We failed to find a match in the quota cache. Search the cache for a free
3   ! entry, or, failing that, the entry with the oldest LRU index.
3   !
3
3   PMS$GL_QUOMISS = .PMS$GL_QUOMISS + 1;
3   LOWEST_LRU = 0;
3   LOWEST_J = 1;
3   INCR J FROM 1 TO .QUOTA_CACHE[VCA$W_QUOSIZE]
3   DO
4       BEGIN
4       IF .QUOTA_LIST[.J-1, VCA$L_QUORECNUM] EQL 0
4       THEN
5           BEGIN
5           LOWEST_J = .J;
5           EXITLOOP;
4           END;
4       LRU_DELTA = .QUOTA_CACHE[VCA$W_QUOLRU] - .QUOTA_LIST[.J-1, VCA$W_QUOLRUX];
4       IF .LRU_DELTA GTRU .LOWEST_LRU
4       THEN
5           BEGIN
5           LOWEST_LRU = .LRU_DELTA;
5           LOWEST_J = .J;
4           END;
3       END;
3
3   ! If the cache entry we are about to use contains a modified entry, we must
3   ! read the corresponding record, update it, and write it. If it represents a
3   ! held lock, we must release it.
3   !
3
3   J = .LOWEST_J;
3   IF  .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
3   AND .QUOTA_LIST[.J-1, VCA$V_QUODIRTY]
3   THEN
4       BEGIN
4       REC_NUM = .QUOTA_LIST[.J-1, VCA$L_QUORECNUM] - 1;
4       OLD_RECORD = READ_BLOCK (.REC_NUM / RECS_PER_BLOCK
4                                     + .FCB[FCB$L_STLBN], 1, QUOTA_TYPE)
4                   + (.REC_NUM MOD RECS_PER_BLOCK) * DQF$C_LENGTH;
4       CLEAN_QUO_CACHE (.J, .OLD_RECORD);
3       END;
3   REL_QUOTA_LOCK (.J);
2   END;                                    ! end of block QUOTA_SEARCH
2
2   ! If the quota cache entry is not marked valid, take out the lock on it.
2   ! If thereafter it is valid, fill in the dummy record with its contents.
2   !
```

```
;  736          1722  2
;  737          1723  2 IF NOT .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  738          1724  2 THEN
;  739          1725  3     BEGIN
;  740          1726  3     QUOTA_LIST[.J-1, VCA$L_QUOUIC] = .UIC;
;  741          1727  3     GET_QUOTA_LOCK (.J, LCK$K_PWMODE);
;  742          1728  2     END;
;  743          1729  2 IF .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  744          1730  2 THEN
;  745          1731  3     BEGIN
;  746          1732  3     DUMMY_REC[DQF$L_FLAGS] = DQF$M_ACTIVE;
;  747          1733  3     QUOTA_RECORD = .QUOTA_LIST[.J-1, VCA$L_QUORECNUM];
;  748          1734  3     DUMMY_REC[DQF$L_UIC] = .QUOTA_LIST[.J-1, VCA$L_QUOUIC];
;  749          1735  3     CH$MOVE (12, QUOTA_LIST[.J-1, VCA$L_USAGE], DUMMY_REC[DQF$L_USAGE]);
;  750          1736  2     END;
;  751          1737  2 QUOTA_INDEX = .J;
;  752          1738  2
;  753          1739  2 .J
;  754          1740  1 END;                                       ! end of routine SCAN_QUO_CACHE


                                                                        .EXTRN  PMS$GL_QUOHIT, PMS$GL_QUOMISS
                                                                        .EXTRN  CACHE_LOCK

                                                         0BFC 00000 SCAN_QUO_CACHE:
                                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R11                 ; 1551
                                    5E              04    C2 00002      SUBL2   #4, SP                                           ;
                                    5B      02C4    CA    9E 00005      MOVAB   708(BASE), R11                                   ; 1616
                                    50        98    AA    D0 0000A      MOVL    -104(BASE), R0                                   ; 1628
                                    53        5C    A0    D0 0000E      MOVL    92(R0), QUOTA_CACHE                              ;
                                    50        98    AA    D0 00012      MOVL    -104(BASE), R0                                   ; 1629
                                    54        54    A0    D0 00016      MOVL    84(R0), FCB                                      ;
                                    3B        0B    A3    E8 0001A      BLBS    11(QUOTA_CACHE), 3$                              ; 1631
                     36       0B    A3              01    E0 0001E      BBS     #1, 11(QUOTA_CACHE), 3$                          ; 1632
                                    01              63    B1 00023      CMPW    (QUOTA_CACHE), #1                                ; 1635
                                                    2D    1B 00026      BLEQU   2$                                               ;
                                    50        94    AA    D0 00028      MOVL    -108(BASE), R0                                   ; 1636
                     24       3A    A0              05    E0 0002C      BBS     #5, 58(R0), 2$                                   ;
                                    01        1C    A4    B1 00031      CMPW    28(FCB), #1                                      ; 1637
                                                    1E    1A 00035      BGTRU   2$                                               ;
                                    50        94    AA    D0 00037      MOVL    -108(BASE), R0                                   ; 1640
                                    10        3C    A0    E9 0003B      BLBC    60(R0), 1$                                       ;
                                                    7E    D4 0003F      CLRL    -(SP)                                            ; 1641
                                              04    A3    9F 00041      PUSHAB  4(QUOTA_CACHE)                                   ;
                                              4C    A4    DD 00044      PUSHL   76(FCB)                                          ;
                          0000G     CF              03    FB 00047      CALLS   #3, CACHE_LOCK                                   ;
                                    06              50    E9 0004C      BLBC    R0, 2$                                           ;
                              0B    A3              01    88 0004F 1$:  BISB2   #1, 11(QUOTA_CACHE)                              ; 1645
                                                    04    11 00053      BRB     3$                                               ;
                              0B    A3              02    88 00055 2$:  BISB2   #2, 11(QUOTA_CACHE)                              ; 1647
                                    52        44    A3    9E 00059 3$:  MOVAB   68(R3), QUOTA_LIST                               ; 1655
                                    55              63    3C 0005D      MOVZWL  (QUOTA_CACHE), R5                                ; 1656
                                                    50    D4 00060      CLRL    K                                                ;
                                                    2E    11 00062      BRB     6$                                               ;
                     51             50              1C    C5 00064 4$:  MULL3   #28, K, R1                                       ; 1659
                                    51              52    C0 00068      ADDL2   QUOTA_LIST, R1                                   ;
```

```
CHARGEQ                                              E 6
V04-000                                         15-Sep-1984 23:56:13    VAX-11 Bliss-32 V4.0-742                  Page  21
                                                14-Sep-1984 12:30:09    DISK$VMSMASTER:[F11X.SRC]CHARGEQ.B32;1     (5)

00      EC    A1           18           00 ED 0006B        CMPZV   #0, #24, -20(R1), #0                 ;
                                        1F 13 00071        BEQL    6$                                   ;
                    04    AC     FC     A1 D1 00073        CMPL    -4(R1), UIC                          ; 1660
                                        18 12 00078        BNEQ    6$                                   ;
                           08     08    AC E9 0007A        BLBC    MARK_USE, 5$                         ; 1663
                    E6    A1     02     A3 B0 0007E        MOVW    2(QUOTA_CACHE), -26(R1)              ; 1666
                                 02     A3 B6 00083        INCW    2(QUOTA_CACHE)                       ; 1667
                          00000000G     00 D6 00086  5$:   INCL    PMS$GL_QUOHIT                        ; 1669
                           56           50 D0 0008C        MOVL    K, J                                 ; 1670
                                      0090 31 0008F        BRW     12$                                  ; 1671
              CE           50           55 F3 00092  6$:   AOBLEQ  R5, K, 4$                            ; 1656
                          00000000G     00 D6 00096        INCL    PMS$GL_QUOMISS                       ; 1679
                                        58 D4 0009C        CLRL    LOWEST_LRU                           ; 1680
                           55           01 D0 0009E        MOVL    #1, LOWEST_J                         ; 1681
                           59           63 3C 000A1        MOVZWL  (QUOTA_CACHE), R9                    ; 1682
                                        50 D4 000A4        CLRL    J                                    ; 1691
                                        2A 11 000A6        BRB     9$                                   ;
              51           50           1C C5 000A8  7$:   MULL3   #28, J, R1                           ; 1685
                           51           52 C0 000AC        ADDL2   QUOTA_LIST, R1                       ;
00      EC    A1           18           00 ED 000AF        CMPZV   #0, #24, -20(R1), #0                 ;
                                        05 12 000B5        BNEQ    8$                                   ;
                           55           50 D0 000B7        MOVL    J, LOWEST_J                          ; 1688
                                        1A 11 000BA        BRB     10$                                  ; 1687
                           57     02    A3 3C 000BC  8$:   MOVZWL  2(QUOTA_CACHE), LRU_DELTA            ; 1691
                           6E     E6    A1 3C 000C0        MOVZWL  -26(R1), (SP)                        ;
                           57           6E C2 000C4        SUBL2   (SP), LRU_DELTA                      ;
                           58           57 D1 000C7        CMPL    LRU_DELTA, LOWEST_LRU                ; 1692
                                        06 1B 000CA        BLEQU   9$                                   ;
                           58           57 D0 000CC        MOVL    LRU_DELTA, LOWEST_LRU                ; 1695
                           55           50 D0 000CF        MOVL    J, LOWEST_J                          ; 1696
              D2           50           59 F3 000D2  9$:   AOBLEQ  R9, J, 7$                            ; 1682
                           56           55 D0 000D6  10$:  MOVL    LOWEST_J, J                          ; 1705
              50           56           1C C5 000D9        MULL3   #28, J, R0                           ; 1706
                           50           52 C0 000DD        ADDL2   QUOTA_LIST, R0                       ;
                           37     EF    A0 E9 000E0        BLBC    -17(R0), 11$                         ;
                           32    EF     A0 01 E1 000E4     BBC     #1, -17(R0), 11$                     ; 1707
55      EC    A0           18           00 EF 000E9        EXTZV   #0, #24, -20(R0), REC_NUM            ; 1710
                                        55 D7 000EF        DECL    REC_NUM                              ;
                                        05 DD 000F1        PUSHL   #5                                   ; 1711
                                        01 DD 000F3        PUSHL   #1                                   ;
              50           55           10 C7 000F5        DIVL3   #16, REC_NUM, R0                     ;
                                 30 B440 9F 000F9          PUSHAB  @48(FCB)[R0]                         ; 1712
                   0000G   CF           03 FB 000FD        CALLS   #3, READ_BLOCK                       ;
7E            00           55           01 7A 00102        EMUL    #1, REC_NUM, #0, -(SP)               ; 1713
51            51           8E           10 7B 00107        EDIV    #16, (SP)+, R1, R1                   ;
                           51           20 C4 0010C        MULL2   #32, R1                              ;
                           50           51 C0 0010F        ADDL2   R1, OLD_RECORD                       ;
                                        50 DD 00112        PUSHL   OLD_RECORD                           ; 1714
                                        56 DD 00114        PUSHL   J                                    ;
                   0000V   CF           02 FB 00116        CALLS   #2, CLEAN_QUO_CACHE                  ;
                                        56 DD 0011B  11$:  PUSHL   J                                    ; 1716
                   0000V   CF           01 FB 0011D        CALLS   #1, REL_QUOTA_LOCK                   ;
              50           56           1C C5 00122  12$:  MULL3   #28, J, R0                           ; 1723
                           50           52 C0 00126        ADDL2   QUOTA_LIST, R0                       ;
                           0E     EF    A0 E8 00129        BLBS    -17(R0), 13$                         ;
                    FC     A0     04    AC D0 0012D        MOVL    UIC, -4(R0)                          ; 1726
                                        04 DD 00132        PUSHL   #4                                   ; 1727
```

```
                                                        56  DD 00134          PUSHL   J                                                   ;
                                      0000V  CF         02  FB 00136          CALLS   #2, GET_QUOTA_LOCK                                  ;
                           50                56         1C  C5 0013B 13$:     MULL3   #28, J, R0                                          ; 1729
                           27          EF A042          00  E1 0013F          BBC     #0, -17(R0)[QUOTA_LIST], 14$                        ;
                                             6B         01  D0 00145          MOVL    #1, (R11)                                           ; 1732
                           50                56         1C  C5 00148          MULL3   #28, J, R0                                          ; 1733
     02B4    CA         EC A042              18         00  EF 0014C          EXTZV   #0, #24, -20(R0)[QUOTA_LIST], 692(BASE)             ;
                           50                56         1C  C5 00155          MULL3   #28, J, R0                                          ; 1734
                                                   FC A042  9F 00159          PUSHAB  -4(R0)[QUOTA_LIST]                                  ;
                                      04     AB         9E  D0 0015D          MOVL    @(SP)+, 4(R11)                                      ;
                           50                56         1C  C5 00161          MULL3   #28, J, R0                                          ; 1735
                      08   AB          F0 A042          0C  28 00165          MOVC3   #12, -16(R0)[QUOTA_LIST], 8(R11)                    ;
                                    02C0     CA         56  D0 0016C 14$:     MOVL    J, 704(BASE)                                        ; 1737
                                             50         56  D0 00171          MOVL    J, R0                                               ; 1740
                                                            04 00174          RET                                                         ;

; Routine Size:  373 bytes,    Routine Base:  $CODE$ + 02BF
```

```
1 GLOBAL ROUTINE GET_QUOTA_LOCK (J, MODE) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine acquires the lock associated with a quota cache
1 !       entry. The lock is raised to PW, and the value block is stored
1 !       in the quota cache entry.
1 !
1 ! CALLING SEQUENCE:
1 !       GET_QUOTA_LOCK (J, MODE)
1 !
1 ! INPUT PARAMETERS:
1 !       J: index of quota cache entry
1 !       MODE: lock mode to use
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !       CURRENT_RVT: RVT of volume set
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       Lock taken out; value block written into cache entry.
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       CACHE_ENTRY     : REF BBLOCK,   ! quota cache entry pointer
2       STATUS,                         ! general status value
2       LOCK_FLAGS      : BBLOCK [4],   ! flags to $ENQ call
2       SAVE_LRU,                       ! save cache entry LRU index
2       RESNAM          : VECTOR [22, BYTE], ! resource name buffer
2       RESNAM_D        : VECTOR [2] INITIAL (22, RESNAM);
2
2 EXTERNAL ROUTINE
2       WAIT_FOR_AST    : L_NORM,       ! wait for completion AST
2       CONTINUE_THREAD : L_NORM,       ! continue execution thread
2       XQP$REL_QUOTA : ADDRESSING_MODE (GENERAL);
2                                       ! unlock cache entry on blocking AST
2
2 BIND_COMMON;
2
2 ! If the volume is not cluster accessible, we don't have to bother with
2 ! locks.
2 !
2
```

```
  2 IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_ALL]
  2 OR NOT .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
  2 THEN RETURN;
  2
  2
  2 ! See if we have a lock ID for the cache entry. If so, this is just a
  2 ! conversion. Otherwise, generate the resource name, using the facility
  2 ! prefix and the volume or volume set name.
  2 !
  2
  2 CACHE_ENTRY = BBLOCKVECTOR [BBLOCK [.CURRENT_VCB[VCB$L_QUOCACHE],
  2                  VCA$L_QUOLIST], .J-1, VCA$R_QUOLOCK; ,VCA$C_QUOLENGTH];
  2
  2 LOCK_FLAGS = LCK$M_SYSTEM + LCK$M_VALBLK + LCK$M_NOQUOTA;
  2
  2 IF .CACHE_ENTRY[VCA$L_QUOLKID] NEQ 0
  2 THEN LOCK_FLAGS[LCK$V_CONVERT] = 1
  2
  2 ELSE
  3     BEGIN
  3     CH$MOVE (6, UPLIT BYTE ('F11B$q'), RESNAM[0]);
  3     CH$MOVE (12,
  3              IF .CURRENT_VCB[VCB$W_RVN] EQL 0
  3              THEN CURRENT_VCB[VCB$T_VOLCKNAM]
  3              ELSE CURRENT_RVT[RVT$T_VLSLCKNAM],
  3              RESNAM[6]);
  3     (RESNAM[18]) = .CACHE_ENTRY[VCA$L_QUOUIC];
  2     END;
  2
  2 ! Acquire the lock.
  2 !
  2
  2 SAVE_LRU = .CACHE_ENTRY[VCA$W_QUOLRUX];
P 2 STATUS = $ENQ (EFN     = EFN,
P 2                LKMODE  = .MODE,
P 2                FLAGS   = .LOCK_FLAGS,
P 2                LKSB    = CACHE_ENTRY[VCA$R_QUOLOCK],
P 2                ASTADR  = CONTINUE_THREAD,
P 2                ASTPRM  = .BASE,
P 2                RESNAM  = RESNAM_D
  2                );
  2 IF NOT .STATUS
  2 THEN
  3     BEGIN
  3     CH$FILL (0, VCA$C_QUOLENGTH, .CACHE_ENTRY);
  3     IF .LOCK_FLAGS[LCK$V_CONVERT]
  4     THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error')
  3     ELSE ERR_EXIT (.STATUS);
  2     END;
  2
  2 IF .STATUS EQL SS$_NORMAL
  2 THEN WAIT_FOR_AST ();
  2
  2 ! Deal with lock completion and handle eny errors. If the lock comes back
  2 ! with value not valid, turn off the valid bit but preserve the contents.
  2 ! We will wtill use the record number to avoid a complete search.
  2 !
```

```
;       2
;       2 STATUS = .CACHE_ENTRY[VCA$W_QUOSTATUS];
;       2
;       2 IF NOT .STATUS
;       2 THEN
;       3     BEGIN
;       3     IF .STATUS EQL SS$_VALNOTVALID
;       3     THEN CACHE_ENTRY[VCA$V_QUOVALID] = 0
;       3     ELSE
;       4         BEGIN
;       4         CH$FILL (0, VCA$C_QUOLENGTH, .CACHE_ENTRY);
;       4         IF .LOCK_FLAGS[LCK$V_CONVERT]
;       5         THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error')
;       4         ELSE ERR_EXIT (.STATUS);
;       3         END;
;       2     END;
;       2
;       2 ! Having acquired the lock, convert it to system owned.
;       2 !
;       2
;     P 2 STATUS = $ENQ (EFN    = EFN,
;     P 2                LKMODE = .MODE,
;     P 2                FLAGS  = LCK$M_NOQUEUE OR LCK$M_SYNCSTS OR LCK$M_CVTSYS OR LCK$M_CONVERT,
;     P 2                LKSB   = CACHE_ENTRY[VCA$R_QUOLOCK],
;     P 2                BLKAST = XQP$REL_QUOTA,
;     P 2                ASTPRM = .CACHE_ENTRY
;       2                );
;       2 IF .STATUS
;       2 THEN STATUS = .CACHE_ENTRY[VCA$W_QUOSTATUS];
;       2 IF NOT .STATUS
;       2 THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error');
;       2
;       2 CACHE_ENTRY[VCA$W_QUOINDEX] = .J;
;       2 CACHE_ENTRY[VCA$W_QUOLRUX] = .SAVE_LRU;
;       2
;       1 END;                                  ! End of routine GET_QUOTA_LOCK


                 71  24  42  31  31  46  00434 P.AAA:  .ASCII  \F11B$q\                               ;

                                                       .EXTRN  WAIT_FOR_AST, CONTINUE_THREAD
                                                       .EXTRN  XQP$REL_QUOTA, SYS$ENQ
                                                       .EXTRN  BUG$_XQPERR

                                   0BFC 00000          .ENTRY  GET_QUOTA_LOCK, Save R2,R3,R4,R5,R6,R7,R8,- ; 1741
                                                               R9,R11
                5B 00000000G  00   9E 00002            MOVAB   SYS$ENQ, R11                        ;
                5E            1C   C2 00009            SUBL2   #28, SP                             ;
                              16   DD 0000C            PUSHL   #22                                 ; 1776
          04    AE        08  AE   9E 0000E            MOVAB   RESNAM, RESNAM_D+4                  ;
                50        94  AA   D0 00013            MOVL    -108(BASE), R0                      ; 1798
                          3A  A0   95 00017            TSTB    58(R0)                              ;
                              01   18 0001A            BGEQ    1$                                  ;
                                   04 0001C            RET                                         ;
                01        3C  A0   E8 0001D 1$:        BLBS    60(R0), 2$                          ; 1799
                                   04 00021            RET                                         ;
                          50      98   AA  D0 00022 2$:     MOVL    -104(BASE), R0                              ; 1808
          56        04    AC           1C  C5 00026         MULL3   #28, J, R6                                  ; 1809
                          56      5C   A0  C0 0002B         ADDL2   92(R0), R6                                  ;
                          56           28  C0 0002F         ADDL2   #40, CACHE_ENTRY                            ;
                          58           31  D0 00032         MOVL    #49, LOCK_FLAGS                             ; 1811
                                  04   A6  D5 00035         TSTL    4(CACHE_ENTRY)                              ; 1813
                                       05  13 00038         BEQL    3$                                          ;
                          58           02  88 0003A         BISB2   #2, LOCK_FLAGS                              ; 1814
                                       25  11 0003D         BRB     6$                                          ;
     08   AE        B7    AF           06  28 0003F 3$:     MOVC3   #6, P.AAA, RESNAM                           ; 1818
                          50      98   AA  D0 00045         MOVL    -104(BASE), R0                              ; 1820
                                  0E   A0  B5 00049         TSTW    14(R0)                                      ;
                                       07  12 0004C         BNEQ    4$                                          ;
                          50    0080   C0  9E 0004E         MOVAB   128(R0), R0                                 ; 1821
                                       05  11 00053         BRB     5$                                          ; 1822
          50        9C    AA           18  C1 00055 4$:     ADDL3   #24, -100(BASE), R0                         ;
     0E   AE              60           0C  28 0005A 5$:     MOVC3   #12, (R0), RESNAM+6                         ; 1823
                    1A    AE      18   A6  D0 0005F         MOVL    24(CACHE_ENTRY), RESNAM+18                  ; 1824
                          59      02   A6  3C 00064 6$:     MOVZWL  2(CACHE_ENTRY), SAVE_LRU                    ; 1830
                                       7E  7C 00068         CLRQ    -(SP)                                       ; 1838
                                       7E  D4 0006A         CLRL    -(SP)                                       ;
                                       5A  DD 0006C         PUSHL   BASE                                        ;
                                0000G  CF  9F 0006E         PUSHAB  CONTINUE_THREAD                             ;
                                       7E  D4 00072         CLRL    -(SP)                                       ;
                                  18   AE  9F 00074         PUSHAB  RESNAM_D                                    ;
                                0140   8F  BB 00077         PUSHR   #^M<R6,R8>                                  ;
                                  08   AC  DD 0007B         PUSHL   MODE                                        ;
                                       1E  DD 0007E         PUSHL   #30                                         ;
                          6B           0B  FB 00080         CALLS   #11, SYS$ENQ                                ;
                          57           50  D0 00083         MOVL    R0, STATUS                                  ;
                          0E           57  E8 00086         BLBS    STATUS, 7$                                  ; 1839
 1C             00        6E           00  2C 00089         MOVC5   #0, (SP), #0, #28, (CACHE_ENTRY)            ; 1842
                                       66     0008E                                                             ;
          33              58           01  E1 0008F         BBC     #1, LOCK_FLAGS, 10$                         ; 1843
                                      FEFF    00093         BUGW                                                ; 1844
                                     0000*    00095         .WORD   <BUG$_XQPERR!4>                             ;
                          01           57  D1 00097 7$:     CMPL    STATUS, #1                                  ; 1848
                                       05  12 0009A         BNEQ    8$                                          ;
                    0000G CF           00  FB 0009C         CALLS   #0, WAIT_FOR_AST                            ; 1849
                          57           66  3C 000A1 8$:     MOVZWL  (CACHE_ENTRY), STATUS                       ; 1856
                          22           57  E8 000A4         BLBS    STATUS, 11$                                 ; 1858
               000009F0   8F           57  D1 000A7         CMPL    STATUS, #2544                               ; 1861
                                       06  12 000AE         BNEQ    9$                                          ;
                    0B    A6           01  8A 000B0         BICB2   #1, 11(CACHE_ENTRY)                         ; 1862
                                       13  11 000B4         BRB     11$                                         ;
 1C             00        6E           00  2C 000B6 9$:     MOVC5   #0, (SP), #0, #28, (CACHE_ENTRY)            ; 1865
                                       66     000BB                                                             ;
          06              58           01  E1 000BC         BBC     #1, LOCK_FLAGS, 10$                         ; 1866
                                      FEFF    000C0         BUGW                                                ; 1867
                                     0000*    000C2         .WORD   <BUG$_XQPERR!4>                             ;
                                       03  11 000C4         BRB     11$                                         ;
                                       57  BF 000C6 10$:    CHMU    STATUS                                      ; 1868
                                           04 000C8         RET                                                 ;
                                       7E  7C 000C9 11$:    CLRQ    -(SP)                                       ; 1881
                            00000000G  00  9F 000CB         PUSHAB  XQP$REL_QUOTA                               ;
                                       56  DD 000D1         PUSHL   CACHE_ENTRY                                 ;
                                       7E  7C 000D3         CLRQ    -(SP)                                       ;
```

```
                             7E  D4 000D5           CLRL    -(SP)                                   ;
                 7E      4E  8F  9A 000D7           MOVZBL  #78, -(SP)                              ;
                             56  DD 000DB           PUSHL   CACHE_ENTRY                             ;
                         08  AC  DD 000DD           PUSHL   MODE                                    ;
                             1E  DD 000E0           PUSHL   #30                                     ;
                 6B          0B  FB 000E2           CALLS   #11, SYS$ENQ                            ;
                 57          50  D0 000E5           MOVL    R0, STATUS                              ;
                 06          57  E9 000E8           BLBC    STATUS, 12$                             ; 1882
                 57          66  3C 000EB           MOVZWL  (CACHE_ENTRY), STATUS                   ; 1883
                 04          57  E8 000EE           BLBS    STATUS, 13$                             ; 1884
                           FEFF 000F1  12$:         BUGW                                            ; 1885
                          0000* 000F3               .WORD   <BUG$_XQPERR!4>                         ;
                 66      04  AC  B0 000F5  13$:     MOVW    J, (CACHE_ENTRY)                        ; 1887
            02   A6          59  B0 000F9           MOVW    SAVE_LRU, 2(CACHE_ENTRY)                ; 1888
                                 04 000FD           RET                                             ; 1890

; Routine Size:  254 bytes,    Routine Base:  $CODE$ + 043A
```

```
  1 GLOBAL ROUTINE REL_QUOTA_LOCK (J) : L_NORM NOVALUE =
  1
  1 !++
  1 !
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       This routine releases the lock associated with a quota cache
  1 !       entry. The value block held in the cache entry is written to
  1 !       the lock.
  1 !
  1 ! CALLING SEQUENCE:
  1 !       REL_QUOTA_LOCK (J)
  1 !
  1 ! INPUT PARAMETERS:
  1 !       J: index of quota cache entry
  1 !
  1 ! IMPLICIT INPUTS:
  1 !       NONE
  1 !
  1 ! OUTPUT PARAMETERS:
  1 !       NONE
  1 !
  1 ! IMPLICIT OUTPUTS:
  1 !       NONE
  1 !
  1 ! ROUTINE VALUE:
  1 !       NONE
  1 !
  1 ! SIDE EFFECTS:
  1 !       Lock released, value block written, cache entry marked non valid.
  1 !
  1 !--
  1
  2 BEGIN
  2
  2 LOCAL
  2         CACHE_ENTRY     : REF BBLOCK;   ! quota cache entry pointer
  2
  2 BIND_COMMON;
  2
  2
  2 ! Release the lock.
  2 !
  2
  2 CACHE_ENTRY = BBLOCKVECTOR [BBLOCK [.CURRENT_VCB[VCB$L_QUOCACHE],
  2                   VCA$L_QUOLIST], .J-1, VCA$R_QUOLOCK; .VCA$C_QUOLENGTH];
  2
  2 IF .CACHE_ENTRY[VCA$L_QUOLKID] NEQ 0
  2 THEN
  3     BEGIN
  3
P 3     IF NOT $DEQ (LKID   = .CACHE_ENTRY[VCA$L_QUOLKID],
P 3                  VALBLK = CACHE_ENTRY[VCA$L_QUORECNUM]
  4                  )
  3     THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error');
  2     END;
  2
```

```
;  964        1948  2 ! Mark the cache entry no longer valid
;  965        1949  2 !
;  966        1950  2
;  967        1951  2 CH$FILL (0, VCA$C_QUOLENGTH, .CACHE_ENTRY);
;  968        1952  2
;  969        1953  1 END;                                   ! End of routine REL_QUOTA_LOCK



                                                                .EXTRN  SYS$DEQ

                                          003C 00000            .ENTRY  REL_QUOTA_LOCK, Save R2,R3,R4,R5               ; 1891
                        50          98  AA  D0 00002            MOVL    -104(BASE), R0                                 ; 1935
           52      04   AC              1C  C5 00006            MULL3   #28, J, R2                                     ; 1936
                        52          5C  A0  C0 0000B            ADDL2   92(R0), R2                                     ;
                        52              28  C0 0000F            ADDL2   #40, CACHE_ENTRY                               ;
                                    04  A2  D5 00012            TSTL    4(CACHE_ENTRY)                                 ; 1938
                                        16  13 00015            BEQL    1$                                             ;
                                        7E  7C 00017            CLRQ    -(SP)                                          ; 1944
                                    08  A2  9F 00019            PUSHAB  8(CACHE_ENTRY)                                 ;
                                    04  A2  DD 0001C            PUSHL   4(CACHE_ENTRY)                                 ;
              00000000G 00              04  FB 0001F            CALLS   #4, SYS$DEQ                                    ;
                        04              50  E8 00026            BLBS    R0, 1$                                         ;
                                          FEFF 00029            BUGW                                                   ; 1945
                                         0000* 0002B            .WORD   <BUG$_XQPERR!4>                                ;
      1C           00   6E              00  2C 0002D 1$:        MOVC5   #0, (SP), #0, #28, (CACHE_ENTRY)               ; 1951
                                        62     00032                                                                   ;
                                            04 00033            RET                                                    ; 1953

; Routine Size:  52 bytes,    Routine Base:  $CODE$ + 0538
```

```
1 GLOBAL ROUTINE CLEAN_QUO_CACHE (J, Q_RECORD) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine updates the indicated quota record buffer from the
1 !       indicated cache entry, and marks the record dirty and marks the
1 !       cache entry clean if necessary.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       CLEAN_QUO_CACHE (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: index in quota cache
1 !             0 to not
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !
1 ! OUTPUT PARAMETERS:
1 !       ARG2: address of record buffer
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !       quota cache entry modified, buffer marked dirty
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       Q_RECORD        : REF BBLOCK;   ! address of quota record
2
2 LOCAL
2       CACHE_ENTRY     : REF BBLOCK;   ! quota cache entry pointer
2
2 BIND_COMMON;
2
2 EXTERNAL ROUTINE
2       MARK_DIRTY      : L_NORM;       ! mark buffer for write back
2
2
2 ! Copy the cache entry to the record buffer. If the cache entry is marked
2 ! dirty, mark it clean and mark the record dirty.
2 !
2
2 CACHE_ENTRY = BBLOCKVECTOR [BBLOCK [.CURRENT_VCB[VCB$L_QUOCACHE],
2                 VCA$L_QUOLIST], .J-1, VCA$R_QUOLOCK; ,VCA$C_QUOLENGTH];
2
2 Q_RECORD[DQF$L_UIC] = .CACHE_ENTRY[VCA$L_QUOUIC];
```

```
; 1028          2011  2 CH$MOVE (12, CACHE_ENTRY[VCA$L_USAGE], Q_RECORD[DQF$L_USAGE]);
; 1029          2012  2 IF .CACHE_ENTRY[VCA$V_QUODIRTY]
; 1030          2013  2 THEN
; 1031          2014  3     BEGIN
; 1032          2015  3     CACHE_ENTRY[VCA$V_QUODIRTY] = 0;
; 1033          2016  3     MARK_DIRTY (.Q_RECORD);
; 1034          2017  2     END;
; 1035          2018  2
; 1036          2019  1 END;                                    ! end of routine CLEAN_QUO_CACHE


                                      007C 00000         .ENTRY  CLEAN_QUO_CACHE, Save R2,R3,R4,R5,R6          ; 1954
                     50       98  AA  D0 00002          MOVL    -104(BASE), R0                                 ; 2007
       56      04    AC           1C  C5 00006          MULL3   #28, J, R6                                     ; 2008
                     56       5C  A0  C0 0000B          ADDL2   92(R0), R6                                     ;
                     56           28  C0 0000F          ADDL2   #40, CACHE_ENTRY                               ;
                     50       08  AC  D0 00012          MOVL    Q_RECORD, R0                                   ; 2010
               04    A0       18  A6  D0 00016          MOVL    24(CACHE_ENTRY), 4(R0)                         ;
                     50       08  AC  D0 0001B          MOVL    Q_RECORD, R0                                   ; 2011
   08  A0      0C    A6           0C  28 0001F          MOVC3   #12, 12(CACHE_ENTRY), 8(R0)                    ;
       0C      0B    A6           01  E1 00025          BBC     #1, 11(CACHE_ENTRY), 1$                        ; 2012
               0B    A6           02  8A 0002A          BICB2   #2, 11(CACHE_ENTRY)                            ; 2015
                              08  AC  DD 0002E          PUSHL   Q_RECORD                                       ; 2016
            0000G    CF           01  FB 00031          CALLS   #1, MARK_DIRTY                                 ;
                                      04 00036 1$:      RET                                                    ; 2019

; Routine Size:  55 bytes,    Routine Base:  $CODE$ + 056C
```

```
1 ROUTINE ENTER_QUO_CACHE (J, Q_RECORD, MARK_DIRTY, MARK_USE) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine enters the given quota record into the cache at the
1 !       indicated cache index. If requested, the cache entry is marked dirty.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       ENTER_QUO_CACHE (ARG1, ARG2, ARG3, ARG4)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: index in quota cache
1 !       ARG2: address of record buffer
1 !       ARG3: 1 to mark record dirty
1 !             0 to not
1 !       ARG4: 0 to set lowest possible LRU
1 !             1 to set current LRU
1 !             2 to leave LRU alone
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !       QUOTA_RECORD: record number of quota record
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !       quota cache entry modified
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       Q_RECORD        : REF BBLOCK;   ! address of quota record
2
2 LOCAL
2       QUOTA_CACHE     : REF BBLOCK,   ! address of quota cache
2       CACHE_ENTRY     : REF BBLOCK;   ! quota cache entry pointer
2
2 BIND_COMMON;
2
2 ! Copy the record data to the cache entry. If requested, mark the cache
2 ! entry dirty.
2 !
2
2 QUOTA_CACHE = .CURRENT_VCB[VCB$L_QUOCACHE];
2 CACHE_ENTRY = BBLOCKVECTOR [BBLOCK [.QUOTA_CACHE, VCA$L_QUOLIST],
; 1095        2077  2              .J-1, VCA$R_QUOLOCK; ,VCA$C_QUOLENGTH];
; 1096        2078  2
; 1097        2079  2
; 1098        2080  2  CACHE_ENTRY[VCA$L_QUOUIC] = .Q_RECORD[DQF$L_UIC];
; 1099        2081  2  CH$MOVE (12, Q_RECORD[DQF$L_USAGE], CACHE_ENTRY[VCA$L_USAGE]);
; 1100        2082  2  CACHE_ENTRY[VCA$B_QUOFLAGS] = VCA$M_QUOVALID;
; 1101        2083  2  CACHE_ENTRY[VCA$W_QUOINDEX] = .J;
; 1102        2084  3  CACHE_ENTRY[VCA$L_QUORECNUM] = (IF .Q_RECORD[DQF$V_ACTIVE]
; 1103        2085  3                                 THEN .QUOTA_RECORD
; 1104        2086  2                                 ELSE 0);
; 1105        2087  2  IF .MARK_USE
; 1106        2088  2  THEN
; 1107        2089  3      BEGIN
; 1108        2090  3      CACHE_ENTRY[VCA$W_QUOLRUX] = .QUOTA_CACHE[VCA$W_QUOLRU];
; 1109        2091  3      QUOTA_CACHE[VCA$W_QUOLRU] = .QUOTA_CACHE[VCA$W_QUOLRU] + 1;
; 1110        2092  3      END
; 1111        2093  2  ELSE IF .MARK_USE EQL 0
; 1112        2094  2  THEN
; 1113        2095  3      BEGIN
; 1114        2096  3      CACHE_ENTRY[VCA$W_QUOLRUX] = .QUOTA_CACHE[VCA$W_QUOLRU] - 1^15;
; 1115        2097  2      END;
; 1116        2098  2
; 1117        2099  2  IF .MARK_DIRTY
; 1118        2100  2  THEN CACHE_ENTRY[VCA$V_QUODIRTY] = 1;
; 1119        2101  2
; 1120        2102  1  END;                                   ! end of routine ENTER_QUO_CACHE


                                        00FC 00000 ENTER_QUO_CACHE:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7                    ; 2020
                       50       98   AA  D0 00002         MOVL    -104(BASE), R0                            ; 2075
                       57       5C   A0  D0 00006         MOVL    92(R0), QUOTA_CACHE                       ;
          50       04  AC            1C  C5 0000A         MULL3   #28, J, R0                                ; 2077
                       56       28 A047  9E 0000F         MOVAB   40(R0)[QUOTA_CACHE], CACHE_ENTRY          ;
                       50       08   AC  D0 00014         MOVL    Q_RECORD, R0                              ; 2080
                  18   A6       04   A0  D0 00018         MOVL    4(R0), 24(CACHE_ENTRY)                    ;
                       50       08   AC  D0 0001D         MOVL    Q_RECORD, R0                              ; 2081
     0C    A6     08   A0            0C  28 00021         MOVC3   #12, 8(R0), 12(CACHE_ENTRY)               ;
                  0B   A6            01  90 00027         MOVB    #1, 11(CACHE_ENTRY)                       ; 2082
                       66       04   AC  B0 0002B         MOVW    J, (CACHE_ENTRY)                          ; 2083
                       07       08   BC  E9 0002F         BLBC    @Q_RECORD, 1$                             ; 2084
                       50     02B4   CA  D0 00033         MOVL    692(BASE), R0                             ; 2085
                                     02  11 00038         BRB     2$                                        ;
                                     50  D4 0003A 1$:     CLRL    R0                                        ; 2084
08   A6           18   00            50  F0 0003C 2$:     INSV    R0, #0, #24, 8(CACHE_ENTRY)               ;
                       0A       10   AC  E9 00042         BLBC    MARK_USE, 3$                              ; 2087
                  02   A6       02   A7  B0 00046         MOVW    2(QUOTA_CACHE), 2(CACHE_ENTRY)            ; 2090
                                02   A7  B6 0004B         INCW    2(QUOTA_CACHE)                            ; 2091
                                     0D  11 0004E         BRB     4$                                        ; 2087
                                10   AC  D5 00050 3$:     TSTL    MARK_USE                                  ; 2093
                                     08  12 00053         BNEQ    4$                                        ;
     02    A6     02   A7     8000   8F  A1 00055         ADDW3   #-32768, 2(QUOTA_CACHE), 2(CACHE_ENTRY)   ; 2096
                       04       0C   AC  E9 0005D 4$:     BLBC    MARK_DIRTY, 5$                            ; 2099
                  0B   A6            02  88 00061         BISB2   #2, 11(CACHE_ENTRY)                       ; 2100
```

```
                                     04 00065 5$:     RET                                              ; 2102

; Routine Size:  102 bytes,    Routine Base:  $CODE$ + 05A3


; 1121          2103  1
; 1122          2104  1 END
; 1123          2105  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                        Bytes                       Attributes
;
;  $CODE$                              1545  NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)


;                               Library Statistics
;
;                                                -------- Symbols --------       Pages       Processing
;       File                                     Total     Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1                 18619         82         0      1000          00:02.0



;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:CHARGEQ/OBJ=OBJ$:CHARGEQ MSRC$:CHARGEQ/UPDATE=(ENH$:CHARGEQ)

; Size:          1539 code + 6 data bytes
; Run Time:         01:30.3
; Elapsed Time:     03:00.2
; Lines/CPU Min:     1398
; Lexemes/CPU-Min: 61595
; Memory Used:  310 pages
; Compilation Complete
```

0168 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY