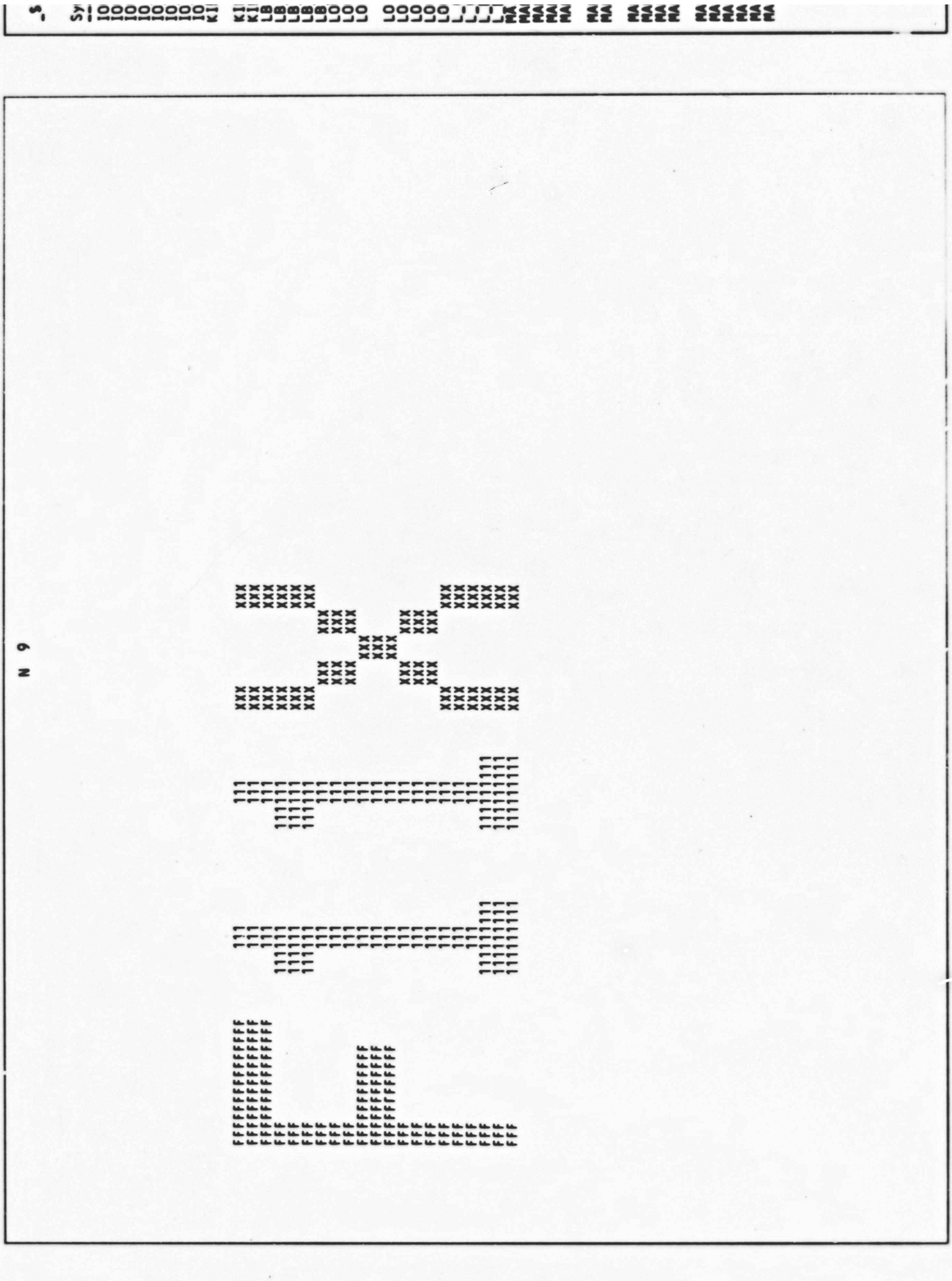


CCCCCCCC HH HH AAAAAA RRRRRRRR GGGGGGGG EEEEEEEEEE QQQQQQ
CCCCCCCC HH HH AAAAAA RRRRRRRR GGGGGGGG EEEEEEEEEE QQQQQQ
CC HH HH AA AA RR RR GG EE QQ QQ
CC HH HH AA AA RR RR GG EE QQ QQ
CC HH HH AA AA RR RR GG EE QQ QQ
CC HH HH AA AA RR RR GG EE QQ QQ
CC HHHHHHHHHH AA AA RRRRRRRR GG EEEEEEEE QQ QQ
CC HHHHHHHHHH 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

```
;     1          0001  0 MODULE CHARGEQ (
;     2          0002  0                 LANGUAGE (BLISS32),
;001  !CDS0005   0003  0                 IDENT = 'V04-001'
;     4-1        0004  0                 ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !**************************************************************************
;     9          0009  1 !*                                                                        *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                               *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                   *
;    13          0013  1 !*                                                                        *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    19          0019  1 !*  TRANSFERRED.                                                           *
;    20          0020  1 !*                                                                        *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    23          0023  1 !*  CORPORATION.                                                           *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    27          0027  1 !*                                                                        *
;    28          0028  1 !*                                                                        *
;    29          0029  1 !**************************************************************************
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:  F11ACP Structure Level 2
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This module contains the routines for charging disk blocks
;    38          0038  1 !       against a particular quota file entry.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT:
;    41          0041  1 !
;    42          0042  1 !       STARLET operating system, including privileged system services
;    43          0043  1 !       and internal exec routines.
;    44          0044  1 !
;    45          0045  1 !--
;    46          0046  1 !
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  22-May-1979  20:51
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;001  !CDS0005   0052  1 !       V04-001 CDS0005         Christian D. Saether    15-Nov-1984
;002  !CDS0005   0053  1 !               Expand test for clusterness to check clu$gl_club.
;003  !CDS0005   0054  1 !
;    52          0055  1 !       V03-012 CDS0004         Christian D. Saether    29-Aug-1984
;    53          0056  1 !               Be prepared to find multiple headers when rebuilding
;    54          0057  1 !               the quota file fcb.  Reread header for PRIMARY_FCB
```

```
;  55          0058  1 !               when rebuilding the quota fcb if it is not the quota
;  56          0059  1 !               file fcb.
;  57          0060  1 !
;  58          0061  1 !       V03-011 CDS0003         Christian D. Saether    23-Aug-1984
;  59          0062  1 !               Check quota fcb for staleness and rebuild if necessary.
;  60          0063  1 !
;  61          0064  1 !       V03-010 ACG0443         Andrew C. Goldstein,    21-Aug-1984  19:51
;  62          0065  1 !               Fix setup of REAL_Q_REC in file search so removal works
;  63          0066  1 !               on a cache miss.
;  64          0067  1 !
;  65          0068  1 !       V03-009 ACG0438         Andrew C. Goldstein,    19-Jul-1984  16:45
;  66          0069  1 !               Implement write access cache interlock
;  67          0070  1 !
;  68          0071  1 !       V03-008 ACG0430         Andrew C. Goldstein,    31-May-1984  15:07
;  69          0072  1 !               Fix reference to quota cache value block in REL_QUOTA_LOCK
;  70          0073  1 !
;  71          0074  1 !       V03-007 ACG0429         Andrew C. Goldstein,    21-May-1984  12:00
;  72          0075  1 !               Fix flow bug in ACG0428
;  73          0076  1 !
;  74          0077  1 !       V03-006 ACG0428         Andrew C. Goldstein,    18-May-1984  14:29
;  75          0078  1 !               Re-read quota record if value block not valid
;  76          0079  1 !
;  77          0080  1 !       V03-005 ACG0408         Andrew C. Goldstein,    23-Mar-1984  14:40
;  78          0081  1 !               Add AST parameter so that impure storage is fully based
;  79          0082  1 !
;  80          0083  1 !       V03-004 ACG0400         Andrew C. Goldstein,    1-Mar-1984  21:09
;  81          0084  1 !               Implement cluster-wide quota cacheing
;  82          0085  1 !
;  83          0086  1 !       V03-003 CDS0002         Christian D. Saether    29-Dec-1983
;  84          0087  1 !               Use L_NORM linkage and BIND_COMMON macro.
;  85          0088  1 !
;  86          0089  1 !       V03-002 CDS0001         Christian D. Saether     6-Dec-1983
;  87          0090  1 !               Serialize quota checking operations using allocation lock.
;  88          0091  1 !
;  89          0092  1 !       V03-001 ACG0337         Andrew C. Goldstein,    16-May-1983  16:04
;  90          0093  1 !               Fix handling of quota cache counters
;  91          0094  1 !
;  92          0095  1 !       V02-006 ACG0229         Andrew C. Goldstein,    23-Dec-1981  21:45
;  93          0096  1 !               Add counters for quota cache hits and misses
;  94          0097  1 !
;  95          0098  1 !       V02-005 ACG0167         Andrew C. Goldstein,    16-Apr-1980  19:25
;  96          0099  1 !               Previous revision history moved to F11B.REV
;  97          0100  1 !**
;  98          0101  1
;  99          0102  1
; 100          0103  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
; 101          0104  1 REQUIRE 'SRC$:FCPDEF.B32';
; 102          1095  1
; 103          1096  1 FORWARD ROUTINE
; 104          1097  1         CHARGE_QUOTA    : L_NORM NOVALUE,  ! check and/or charge disk blocks
; 105          1098  1         SEARCH_QUOTA    : L_NORM,          ! search for a quota file record
; 106          1099  1         WRITE_QUOTA     : L_NORM NOVALUE,  ! write back a quota record
; 107          1100  1         SCAN_QUO_CACHE  : L_NORM,          ! search the quota cache
; 108          1101  1         GET_QUOTA_LOCK  : L_NORM NOVALUE,  ! acquire lock on quota file entry
; 109          1102  1         REL_QUOTA_LOCK  : L_NORM NOVALUE,  ! release lock on quota file entry
; 110          1103  1         CLEAN_QUO_CACHE : L_NORM NOVALUE,  ! write modified cache entry
; 111          1104  1         ENTER_QUO_CACHE : L_NORM NOVALUE; ! make a new cache entry
```

```
;  113          1105  1 GLOBAL ROUTINE CHARGE_QUOTA (UIC, BLOCK_COUNT, FLAGS) : L_NORM NOVALUE =
;  114          1106  1
;  115          1107  1 !++
;  116          1108  1 !
;  117          1109  1 ! FUNCTIONAL DESCRIPTION:
;  118          1110  1 !
;  119          1111  1 !       This routine locates the quota file entry identified by the UIC
;  120          1112  1 !       given and checks and/or charges the indicated number of blocks,
;  121          1113  1 !       as specified by the flags.
;  122          1114  1 !
;  123          1115  1 ! CALLING SEQUENCE:
;  124          1116  1 !       CHARGE_QUOTA (ARG1, ARG2, ARG3)
;  125          1117  1 !
;  126          1118  1 ! INPUT PARAMETERS:
;  127          1119  1 !       ARG1: UIC of entry to charge
;  128          1120  1 !       ARG2: number of blocks to charge (negative to credit)
;  129          1121  1 !       ARG3: bit encoded flags
;  130          1122  1 !             bit 0 set to check if quota will be exceeded
;  131          1123  1 !             bit 1 set to actually charge blocks to the quota entry
;  132          1124  1 !
;  133          1125  1 ! IMPLICIT INPUTS:
;  134          1126  1 !       IO_PACKET: user's I/O packet
;  135          1127  1 !       CURRENT_RVN: RVN of volume
;  136          1128  1 !
;  137          1129  1 ! OUTPUT PARAMETERS:
;  138          1130  1 !       NONE
;  139          1131  1 !
;  140          1132  1 ! IMPLICIT OUTPUTS:
;  141          1133  1 !       NONE
;  142          1134  1 !
;  143          1135  1 ! ROUTINE VALUE:
;  144          1136  1 !       NONE
;  145          1137  1 !
;  146          1138  1 ! SIDE EFFECTS:
;  147          1139  1 !       quota file modified
;  148          1140  1 !
;  149          1141  1 !--
;  150          1142  1
;  151          1143  1
;  152          1144  2 BEGIN
;  153          1145  2
;  154          1146  2 MAP
;  155          1147  2         FLAGS           : BITVECTOR;    ! flags argument
;  156          1148  2
;  157          1149  2 LABEL
;  158          1150  2         CHECK_QUOTA;                    ! block of code to check quota
;  159          1151  2
;  160          1152  2 LOCAL
;  161          1153  2         SAVE_RVN,                       ! place to save current RVN
;  162          1154  2         Q_RECORD        : REF BBLOCK;   ! address of quota file record
;  163          1155  2
;  164          1156  2 BIND_COMMON;
;  165          1157  2
;  166          1158  2 EXTERNAL ROUTINE
;  167          1159  2         SWITCH_VOLUME   : L_NORM;       ! switch to desired RVN
;  168          1160  2
;  169          1161  2
```

```
;  170          1162  2 ! Save the current RVN and then switch context to the root RVN.
;  171          1163  2 ! First locate the quota file record. If there is no quota file enabled, this
;  172          1164  2 ! routine is a NOP.
;  173          1165  2 !
;  174          1166  2
;  175          1167  2 SAVE_RVN = .CURRENT_RVN;
;  176          1168  2 SWITCH_VOLUME (1);
;  177          1169  2
;  178          1170  2 CHECK_QUOTA: BEGIN
;  179          1171  2
;  180          1172  2 Q_RECORD = SEARCH_QUOTA (.UIC, 0, 0, 1);
;  181          1173  2 IF .Q_RECORD EQL -1 THEN LEAVE CHECK_QUOTA;
;  182          1174  2
;  183          1175  2 ! Check for quota exceeded if requested and the user does not have EXQUOTA
;  184          1176  2 ! privilege. If we are to check, lack of a quota record is an error; if
;  185          1177  2 ! we do not check, this routine is a NOP.
;  186          1178  2 !
;  187          1179  2
;  188          1180  3 IF .FLAGS[QUOTA_CHECK]
;  189          1181  3 AND NOT .BBLOCK [BBLOCK [.IO_PACKET[IRP$L_ARB], ARB$Q_PRIV], PRV$V_EXQUOTA]
;  190          1182  3 THEN
;  191          1183  4     BEGIN
;  192          1184  4     IF .Q_RECORD EQL 0
;  193          1185  4     THEN ERR_EXIT (SS$_EXDISKQUOTA);
;  194          1186  4     IF .Q_RECORD[DQF$L_USAGE] + .BLOCK_COUNT GTRU .Q_RECORD[DQF$L_PERMQUOTA]
;  195          1187  4     THEN
;  196          1188  5         BEGIN
;  197          1189  5         IF .CURRENT_WINDOW NEQ 0
;  198          1190  5         THEN
;  199          1191  6             BEGIN
;  200          1192  6             IF .CURRENT_WINDOW[WCB$V_OVERDRAWN]
;  201          1193  6             THEN
;  202          1194  7                 BEGIN
;  203          1195  7                 IF .Q_RECORD[DQF$L_USAGE] + .BLOCK_COUNT GTRU
;  204          1196  7                     .Q_RECORD[DQF$L_PERMQUOTA] + .Q_RECORD[DQF$L_OVERDRAFT]
;  205          1197  8                 THEN ERR_EXIT (SS$_EXDISKQUOTA)
;  206          1198  7                 ELSE ERR_STATUS (SS$_OVRDSKQUOTA);
;  207          1199  7                 END
;  208          1200  6             ELSE
;  209          1201  7                 BEGIN
;  210          1202  7                 CURRENT_WINDOW[WCB$V_OVERDRAWN] = 1;
;  211          1203  7                 ERR_EXIT (SS$_EXDISKQUOTA);
;  212          1204  6                 END;
;  213          1205  6             END
;  214          1206  5         ELSE
;  215          1207  5             ERR_EXIT (SS$_EXDISKQUOTA);
;  216          1208  4         END;
;  217          1209  4     END
;  218          1210  4
;  219          1211  3 ELSE
;  220          1212  3     IF .Q_RECORD EQL 0 THEN LEAVE CHECK_QUOTA;
;  221          1213  3
;  222          1214  3 ! If the record is to be charged, do so. Check the result to see if it
;  223          1215  3 ! is negative; if so, zero it to prevent absurd results.
;  224          1216  3 !
;  225          1217  3
;  226          1218  3 IF .FLAGS[QUOTA_CHARGE]
```

```
;  227     1219  3 THEN
;  228     1220  4     BEGIN
;  229     1221  4     Q_RECORD[DQF$L_USAGE] = .Q_RECORD[DQF$L_USAGE] + .BLOCK_COUNT;
;  230     1222  4     IF .Q_RECORD[DQF$L_USAGE] [SS 0
;  231     1223  4     THEN Q_RECORD[DQF$L_USAGE] = 0;
;  232     1224  4     WRITE_QUOTA (.Q_RECORD);
;  233     1225  3     END;
;  234     1226  3
;  235     1227  2 END;                                    ! end of block CHECK_QUOTA
;  236     1228  2
;  237     1229  2 SWITCH_VOLUME (.SAVE_RVN);
;  238     1230  2
;  239     1231  1 END;                                    ! end of routine CHARGE_QUOTA

                                                        .TITLE  CHARGEQ
                                                        .IDENT  \V04-001\

                                                        .EXTRN  SWITCH_VOLUME

                                                        .PSECT  $CODE$,NOWRT,2

                                        000C 00000      .ENTRY  CHARGE_QUOTA, Save R2,R3        ; 1105
                    53      A0      AA   D0 00002       MOVL    -96(BASE), SAVE_RVN             ; 1167
                                    01   DD 00006       PUSHL   #1                              ; 1168
                0000G   CF          01   FB 00008       CALLS   #1, SWITCH_VOLUME
                                    01   DD 0000D       PUSHL   #1                              ; 1172
                                    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                   ; 1173
                                    62   13 00020       BEQL    6$
                        44      0C  AC   E9 00022       BLBC    FLAGS, 3$                       ; 1180
                        51      90  AA   D0 00026       MOVL    -112(BASE), R1                  ; 1181
            3B          58      B1  13   E0 0002A       BBS     #19, @88(R1), 3$
                                    50   D5 0002F       TSTL    Q_RECORD                        ; 1184
                                    32   13 00031       BEQL    2$
            52          08      A0  08   AC   C1 00033  ADDL3   BLOCK_COUNT, 8(Q_RECORD), R2   ; 1186
                        0C      A0       52   D1 00039  CMPL    R2, 12(Q_RECORD)
                                         2F   1B 0003D  BLEQU   4$
                        51      0C  AA   D0 0003F       MOVL    12(BASE), R1                    ; 1189
                                    20   13 00043       BEQL    2$
            17          0B      A1       04   E1 00045  BBC     #4, 11(R1), 1$                  ; 1192
            51          0C      A0  10   A0   C1 0004A  ADDL3   16(Q_RECORD), 12(Q_RECORD), R1  ; 1196
                        51          52   D1 00050       CMPL    R2, R1
                                    10   1A 00053       BGTRU   2$
                        15      80  AA   E9 00055       BLBC    -128(BASE), 4$                  ; 1198
                        80      AA  0669 8F   B0 00059  MOVW    #1641, -128(BASE)
                                    0D   11 0005F       BRB     4$                              ; 1192
                        0B      A1       10   88 00061  1$:    BISB2   #16, 11(R1)             ; 1202
                                03EC     8F   BF 00065  2$:    CHMU    #1004                   ; 1207
                                         04   00069            RET
                                    50   D5 0006A  3$:    TSTL    Q_RECORD                        ; 1212
                                    16   13 0006C       BEQL    6$
            11          0C      AC       01   E1 0006E  4$:    BBC     #1, FLAGS, 6$           ; 1218
                        08      A0  08   AC   C0 00073       ADDL2   BLOCK_COUNT, 8(Q_RECORD)    ; 1221
                                         03   18 00078       BGEQ    5$                      ; 1222
```

```
                         08  A0  D4 0007A         CLRL    8(Q_RECORD)                             ; 1223
                             50  DD 0007D 5$:     PUSHL   Q_RECORD                                ; 1224
                 0000V   CF      01  FB 0007F     CALLS   #T, WRITE_QUOTA                         ;
                             53  DD 00084 6$:     PUSHL   SAVE_RVN                                ; 1229
                 0000G   CF      01  FB 00086     CALLS   #1, SWITCH_VOLUME                       ;
                                 04  0008B         RET                                             ; 1231

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

```
;  241     1232  1 GLOBAL ROUTINE SEARCH_QUOTA (UIC, FLAGS, START_REC, USE_CACHE) : L_NORM =
;  242     1233  1
;  243     1234  1 !++
;  244     1235  1 !
;  245     1236  1 ! FUNCTIONAL DESCRIPTION:
;  246     1237  1 !
;  247     1238  1 !       This routine searches the quota file for the specified UIC under
;  248     1239  1 !       control of the match flags.
;  249     1240  1 !
;  250     1241  1 ! CALLING SEQUENCE:
;  251     1242  1 !       SEARCH_QUOTA (ARG1, ARG2, ARG3, ARG4)
;  252     1243  1 !
;  253     1244  1 ! INPUT PARAMETERS:
;  254     1245  1 !       ARG1: UIC to search for
;  255     1246  1 !       ARG2: match control flags from FIB
;  256     1247  1 !       ARG3: record number at which to start
;  257     1248  1 !       ARG4: 1 to find record in the cache
;  258     1249  1 !             0 to unconditionally go to the quota file
;  259     1250  1 !
;  260     1251  1 ! IMPLICIT INPUTS:
;  261     1252  1 !       CURRENT_VCB: address of volume's VCB
;  262     1253  1 !       context set to RVN 1 of volume set
;  263     1254  1 !
;  264     1255  1 ! OUTPUT PARAMETERS:
;  265     1256  1 !       NONE
;  266     1257  1 !
;  267     1258  1 ! IMPLICIT OUTPUTS:
;  268     1259  1 !       QUOTA_RECORD: record number of found record
;  269     1260  1 !       FREE_QUOTA: record number of first free quota file entry
;  270     1261  1 !       QUOTA_INDEX: cache index of cache entry if found
;  271     1262  1 !       DUMMY_REC: filled in with cache contents if found
;  272     1263  1 !
;  273     1264  1 ! ROUTINE VALUE:
;  274     1265  1 !       address of quota file record found, 0 if none, -1 if not quota file
;  275     1266  1 !
;  276     1267  1 ! SIDE EFFECTS:
;  277     1268  1 !       quota file read, contents of buffer cache altered
;  278     1269  1 !
;  279     1270  1 !--
;  280     1271  1
;  281     1272  2 BEGIN
;  282     1273  2
;  283     1274  2 MAP
;  284     1275  2         FLAGS           : BITVECTOR;    ! match control flags
;  285     1276  2
;  286     1277  2 LITERAL
;  287     1278  2         ALL_GROUP       = $BITPOSITION (FIB$V_ALL_GRP),
;  288     1279  2         ALL_MEMBER      = $BITPOSITION (FIB$V_ALL_MEM),
;  289     1280  2         RECS_PER_BLOCK  = 512 / DQF$C_LENGTH;
;  290     1281  2
;  291     1282  2 LABEL
;  292     1283  2         QUOTA_SCAN;                     ! search quota file
;  293     1284  2
;  294     1285  2 LOCAL
;  295     1286  2         FCB             : REF BBLOCK,   ! address of quota file FCB
;  296     1287  2         QUOTA_CACHE     : REF BBLOCK,   ! address of quota cache
;  297     1288  2         QUOTA_LIST      : REF BBLOCKVECTOR [,VCA$C_QUOLENGTH],
```

```
;  298          1289  2                                   ! address of quota cache entries
;  299          1290  2          J,                       ! index into quota cache
;  300          1291  2          REC_NUM,                 ! quota file record to read
;  301          1292  2          FIRST_REC,               ! first record in block to use
;  302          1293  2          VBN,                     ! block number of quota file
;  303          1294  2          Q_RECORD        : REF BBLOCK;   ! address of quota file record
;  304          1295  2
;  305          1296  2      BIND_COMMON;
;  306          1297  2
;  307          1298  2      EXTERNAL ROUTINE
;  308          1299  2              ALLOCATION_LOCK : L_NORM,       ! allocation lock serialization
;  309          1300  2              BUILD_EXT_FCBS  : L_NORM NOVALUE, ! build extension fcb chain.
;  310          1301  2              SERIAL_FILE     : L_NORM,       ! get serialization lock
;  311          1302  2              REBLD_PRIM_FCB  : L_NORM NOVALUE, ! rebuild fcb from header
;  312          1303  2              READ_HEADER     : L_NORM,       ! read file header
;  313          1304  2              RELEASE_SERIAL_LOCK : L_NORM NOVALUE, ! release serialization lock
;  314          1305  2              READ_BLOCK      : L_NORM;       ! read a disk block
;  315          1306  2
;  316          1307  2
;  317          1308  2      ! Get the FCB address for the quota file. If none, take an error exit.
;  318          1309  2      !
;  319          1310  2
;  320          1311  2      FCB = .CURRENT_VCB[VCB$L_QUOTAFCB];
;  321          1312  2      IF .FCB EQL 0 THEN RETURN -1;
;  322          1313  2
;  323          1314  2      ! Serialize quota search using allocation lock.
;  324          1315  2      !
;  325          1316  2
;  326          1317  2      ALLOCATION_LOCK ();
;  327          1318  2
;  328          1319  2      ! Check to see if the quota fcb is stale, that is, it has been modified
;  329          1320  2      ! on another node.  If so, serialize on the quota file itself, read
;  330          1321  2      ! the header and rebuild the fcb.
;  331          1322  2      !
;  332          1323  2
;  333          1324  2      IF .FCB [FCB$V_STALE]
;  334          1325  2      THEN
;  335          1326  2          BEGIN
;  336          1327  2          LOCAL
;  337          1328  2              HEADER,
;  338          1329  2              SAV_CURRLCKINDX;
;  339          1330  2
;  340          1331  2          SAV_CURRLCKINDX = .CURR_LCKINDX;
;  341          1332  2
;  342          1333  2          SERIAL_FILE (FCB [FCB$W_FID]);
;  343          1334  2
;  344          1335  2      ! Setting this flag prevents READ_HEADER from modifying FILE_HEADER.
;  345          1336  2      ! It is also set when the BUILD_EXT_FCBS routine calls READ_HEADER
;  346          1337  2      ! if BUILD_EXT_FCBS is called with the optional fcb argument.
;  347          1338  2      !
;  348          1339  2
;  349          1340  2          STSFLGS [STS_LEAVE_FILEHDR] = 1;
;  350          1341  2
;  351          1342  2          HEADER = READ_HEADER (0, .FCB);
;  352          1343  2
;  353          1344  2          REBLD_PRIM_FCB (.FCB, .HEADER);
;  354          1345  3
```

```
;  355          1346  3     BUILD_EXT_FCBS (.HEADER, .FCB);
;  356          1347
;  357          1348        IF .SAV_CURRLCKINDX NEQ .CURR_LCKINDX
;  358          1349        THEN
;  359          1350  4         BEGIN
;  360          1351  4         RELEASE_SERIAL_LOCK (.CURR_LCKINDX);
;  361          1352  4         CURR_LCKINDX = .SAV_CURRLCKINDX;
;  362          1353  4         END;
;  363          1354
;  364          1355        ! If PRIMARY_FCB is nonzero and not the quota file fcb, and also if
;  365          1356        ! it is the same lockbasis as the current lock index, then reread
;  366          1357        ! the header for it to re-establish FILE_HEADER.
;  367          1358        !
;  368          1359
;  369          1360        IF .PRIMARY_FCB NEQ 0
;  370          1361        THEN
;  371          1362            IF .PRIMARY_FCB NEQ .FCB
;  372          1363               AND .PRIMARY_FCB [FCB$L_LOCKBASIS] EQL .LB_BASIS [.CURR_LCKINDX]
;  373          1364            THEN
;  374          1365                READ_HEADER (0, .PRIMARY_FCB);
;  375          1366
;  376          1367        END;
;  377          1368
;  378          1369        ! If there are no wild cards in the search, scan the quota cache first.
;  379          1370        ! If the value block was lost, the cache entry comes back not valid,
;  380          1371        ! but with contents. In this case, and if this is a write-through operation,
;  381          1372        ! use the record number in the cache to read the record to save the search.
;  382          1373        ! As long as the record is read, also update it if the cache entry is dirty.
;  383          1374        !
;  384          1375    2
;  385          1376    2   REAL_Q_REC = 0;
;  386          1377    2   QUOTA_CACHE = .CURRENT_VCB[VCB$L_QUOCACHE];
;  387          1378    2   QUOTA_LIST = QUOTA_CACHE[VCA$L_QUOLIST];
;  388          1379    2   IF NOT .FLAGS[ALL_MEMBER] AND NOT .FLAGS[ALL_GROUP]
;  389          1380    2   THEN
;  390          1381    3       BEGIN
;  391          1382    3       J = SCAN_QUO_CACHE (.UIC, .USE_CACHE);
;  392          1383    3       IF .QUOTA_LIST[.J-1, VCA$L_QUORECNUM] NEQ 0
;  393          1384    3       THEN
;  394          1385    4           BEGIN
;  395          1386    4           IF NOT .USE_CACHE
;  396          1387    4           OR NOT .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  397          1388    4           OR NOT .QUOTA_CACHE[VCA$V_CACHEVALID]
;  398          1389    4           THEN
;  399          1390    5               BEGIN
;  400          1391    5               QUOTA_RECORD = .QUOTA_LIST[.J-1, VCA$L_QUORECNUM];
;  401          1392    5               REC_NUM = .QUOTA_LIST[.J-1, VCA$L_QUORECNUM] - 1;
;  402          1393    5               REAL_Q_REC = READ_BLOCK (.REC_NUM / RECS_PER_BLOCK
;  403          1394    5                                       + .FCB[FCB$L_ST[BN], 1, QUOTA_TYPE)
;  404          1395    5                                   + (.REC_NUM MOD RECS_PER_BLOCK) * DQF$C_LENGTH;
;  405          1396    5               IF .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  406          1397    5               THEN
;  407          1398    5                   CLEAN_QUO_CACHE (.J, .REAL_Q_REC)
;  408          1399    5               ELSE
;  409          1400    6                   BEGIN
;  410          1401    6                   ENTER_QUO_CACHE (.J, .REAL_Q_REC, 0, .USE_CACHE);
;  411          1402    6                   CH$MOVE (DQF$C_LENGTH, .REAL_Q_REC, DUMMY_REC);
```

```
;  412     1403  5                 END;
;  413     1404  4             END;
;  414     1405  4         RETURN DUMMY_REC;
;  415     1406  3         END;
;  416     1407  2     END;
;  417     1408  2
;  418     1409  2 ! We couldn't find a valid cache entry (either because it's not there
;  419     1410  2 ! or the operation won't allow it). Scan the blocks of the quota file,
;  420     1411  2 ! looking for a matching record.
;  421     1412  2 !
;  422     1413  2
;  423     1414  3 QUOTA_SCAN: BEGIN
;  424     1415  3
;  425     1416  3 FIRST_REC = .START_REC MOD RECS_PER_BLOCK;
;  426     1417  3 INCR VBN FROM .START_REC/RECS_PER_BLOCK TO .FCB[FCB$L_EFBLK] - 1
;  427     1418  3 DO
;  428     1419  4     BEGIN
;  429     1420  4     Q_RECORD = READ_BLOCK (.VBN + .FCB[FCB$L_STLBN],
;  430     1421  4                           .FCB[FCB$L_EFBLK] - .VBN,
;  431     1422  4                           QUOTA_TYPE)
;  432     1423  4               + .FIRST_REC * DQF$C_LENGTH;
;  433     1424  4
;  434     1425  4     INCR J FROM .FIRST_REC TO RECS_PER_BLOCK - 1
;  435     1426  4     DO
;  436     1427  5         BEGIN
;  437     1428  5         QUOTA_RECORD = .VBN * RECS_PER_BLOCK + .J + 1;
;  438     1429  5
;  439     1430  5         IF .Q_RECORD[DQF$V_ACTIVE]
;  440     1431  5         THEN
;  441     1432  6             BEGIN
;  442     1433  7             IF  (.FLAGS[ALL_MEMBER] OR .UIC<00,16> EQL .(Q_RECORD[DQF$L_UIC])<00,16>)
;  443     1434  7             AND (.FLAGS[ALL_GROUP]  OR .UIC<16,16> EQL .(Q_RECORD[DQF$L_UIC])<16,16>)
;  444     1435  6             THEN LEAVE QUOTA_SCAN;
;  445     1436  6             END
;  446     1437  6
;  447     1438  5         ELSE
;  448     1439  6             BEGIN
;  449     1440  6             IF .FREE_QUOTA EQL 0
;  450     1441  6             THEN FREE_QUOTA = .QUOTA_RECORD;
;  451     1442  5             END;
;  452     1443  5
;  453     1444  5         Q_RECORD = .Q_RECORD + DQF$C_LENGTH;
;  454     1445  4         END;                                    ! end of inner loop
;  455     1446  4
;  456     1447  4     FIRST_REC = 0;
;  457     1448  3     END;                                        ! end of block scan loop
;  458     1449  3
;  459     1450  3 IF NOT .FLAGS[ALL_MEMBER] AND NOT .FLAGS[ALL_GROUP]
;  460     1451  3 THEN REL_QUOTA_LOCK (.J);
;  461     1452  3 RETURN 0;                                       ! return 0 if not found
;  462     1453  2 END;                                           ! end of block QUOTA_SCAN
;  463     1454  2
;  464     1455  2 ! We have found a record in the quota file. If there were wild cards, now
;  465     1456  2 ! scan the quota cache to see if an entry is present. With wild cards, the
;  466     1457  2 ! file must be scanned first to be able to return the entries in a coherent
;  467     1458  2 ! order; yet we must look at the cache in case a modified entry is present.
;  468     1459  2 !
```

```
;  469          1460  2
;  470          1461  2 REAL_Q_REC = .Q_RECORD;
;  471          1462  2 IF .FLAGS[ALL_MEMBER] OR .FLAGS[ALL_GROUP]
;  472          1463  2 THEN
;  473          1464  3     BEGIN
;  474          1465  3     J = SCAN_QUO_CACHE (.Q_RECORD[DQF$L_UIC], 0);
;  475          1466  3     IF .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  476          1467  3     THEN
;  477          1468  4         BEGIN
;  478          1469  4         CLEAN_QUO_CACHE (.J, .Q_RECORD);
;  479          1470  4         RETURN DUMMY_REC;
;  480          1471  3         END;
;  481          1472  2     END;
;  482          1473  2
;  483          1474  2 ! Finally enter the new record into the quota cache.
;  484          1475  2 !
;  485          1476  2
;  486          1477  2 ENTER_QUO_CACHE (.J, .Q_RECORD, 0, .USE_CACHE);
;  487          1478  2 CH$MOVE (DQF$C_LENGTH, .Q_RECORD, DUMMY_REC);
;  488          1479  2 RETURN DUMMY_REC;
;  489          1480  2
;  490          1481  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,-; 1232
                                                                R11
                          02B4   CA  9F 00002           PUSHAB  692(BASE)                                     ; 1294
                       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                                ; 1311
                       57   54   A0  D0 00014           MOVL    84(R0), FCB
                                 04  12 00018           BNEQ    1$                                            ; 1312
                       50        01  CE 0001A           MNEGL   #1, R0
                                 04  04 0001D           RET
                0000G  CF        00  FB 0001E 1$:       CALLS   #0, ALLOCATION_LOCK                           ; 1317
                       61   23   A7  E9 00023           BLBC    35(FCB), 3$                                   ; 1324
                       52   14   AA  D0 00027           MOVL    20(BASE), SAV_CURRLCKINDX                     ; 1331
                            24   A7  9F 0002B           PUSHAB  36(FCB)                                       ; 1332
                0000G  CF        01  FB 0002E           CALLS   #1, SERIAL_FILE
                  A6   AA        08  88 00033           BISB2   #8, -90(BASE)                                 ; 1340
                                 57  DD 00037           PUSHL   FCB                                           ; 1342
                                 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                                        ; 1344
                                 57  DD 00045           PUSHL   FCB
                0000G  CF        02  FB 00047           CALLS   #2, REBLD_PRIM_FCB
                          0088   8F  BB 0004C           PUSHR   #^M<R3,R7>                                    ; 1346
                0000G  CF        02  FB 00050           CALLS   #2, BUILD_EXT_FCBS
                  14   AA        52  D1 00055           CMPL    SAV_CURRLCKINDX, 20(BASE)                     ; 1348
```

```
                                     0C  13 00059      BEQL    2$                                         ;
                                14   AA  DD 0005B      PUSHL   20(BASE)                                   ; 1351
                      0000G  CF      01  FB 0005E      CALLS   #1, RELEASE_SERIAL_LOCK                    ;
                         14  AA      52  D0 00063      MOVL    SAV_CURRLCKINDX, 20(BASE)                  ; 1352
                             51 08   AA  D0 00067 2$:  MOVL    8(BASE), R1                                ; 1360
                                     1B  13 0006B      BEQL    3$                                         ;
                             57      51  D1 0006D      CMPL    R1, FCB                                    ; 1362
                                     16  13 00070      BEQL    3$                                         ;
                             50 14   AA  D0 00072      MOVL    20(BASE), R0                               ; 1363
                  0080 CA40     4C   A1  D1 00076      CMPL    76(R1), 128(BASE)[R0]                      ;
                                     09  12 0007D      BNEQ    3$                                         ;
                                     51  DD 0007F      PUSHL   R1                                         ; 1365
                                     7E  D4 00081      CLRL    -(SP)                                      ;
                      0000G  CF      02  FB 00083      CALLS   #2, READ_HEADER                            ;
                                     69  D4 00088 3$:  CLRL    (R9)                                       ; 1376
                             50 98   AA  D0 0008A      MOVL    -104(BASE), R0                             ; 1377
                             52 5C   A0  D0 0008E      MOVL    92(R0), QUOTA_CACHE                        ;
                             56 44   A2  9E 00092      MOVAB   68(R2), QUOTA_LIST                         ; 1378
                             03 08   AC  E9 00096      BLBC    FLAGS, 5$                                  ; 1379
                                   008F  31 0009A 4$:  BRW     10$                                        ;
             F8       08     AC      01  E0 0009D 5$:  BBS     #1, FLAGS, 4$                              ;
                                10   AC  DD 000A2      PUSHL   USE_CACHE                                  ; 1382
                                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                                 ; 1383
                             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$                              ; 1386
                             04 EF   A0  E9 000C3      BLBC    -17(R0), 6$                                ; 1387
                             5E 0B   A2  E8 000C7      BLBS    11(QUOTA_CACHE), 9$                        ; 1388
   00   BE        EC  A0     18      00  EF 000CB 6$:  EXTZV   #0, #24, -20(R0), @0(SP)                   ; 1391
                      50     58      1C  C5 000D2      MULL3   #28, J, R0                                 ; 1392
        52        EC A046    18      00  EF 000D6      EXTZV   #0, #24, -20(R0)[QUOTA_LIST], REC_NUM      ;
                                     52  D7 000DD      DECL    REC_NUM                                    ;
                                     05  DD 000DF      PUSHL   #5                                         ; 1393
                                     01  DD 000E1      PUSHL   #1                                         ;
             50              52      10  C7 000E3      DIVL3   #16, REC_NUM, R0                           ;
                              30 B740    9F 000E7      PUSHAB  @48(FCB)[R0]                               ; 1394
                      0000G  CF      03  FB 000EB      CALLS   #3, READ_BLOCK                             ;
        7E   00              52      01  7A 000F0      EMUL    #1, REC_NUM, #0, -(SP)                     ; 1395
        51   51              8E      10  7B 000F5      EDIV    #16, (SP)+, R1, R1                         ;
                             51      20  C4 000FA      MULL2   #32, R1                                    ;
             69              50      51  C1 000FD      ADDL3   R1, R0, (R9)                               ;
             50              58      1C  C5 00101      MULL3   #28, J, R0                                 ; 1396
             0B          EF A046     00  E1 00105      BBC     #0, -17(R0)[QUOTA_LIST], 8$                ;
                                     69  DD 0010B      PUSHL   (R9)                                       ; 1398
                                     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                                  ; 1401
                                     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)                         ; 1402
```

```
                                     00C1  31 00129 9$:     BRW     21$                                        ; 1405
     7E      00      0C   AC           01  7A 0012C 10$:    EMUL    #1, START_REC, #0, -(SP)                   ; 1416
     53      53           8E           10  7B 00132         EDIV    #16, (SP)+, FIRST_REC, FIRST_REC
             52      0C   AC           10  C7 00137         DIVL3   #16, START_REC, R2                         ; 1417
                          55      3C   A7  D0 0013C         MOVL    60(FCB), R5
                                       52  D7 00140         DECL    VBN
                                       59  11 00142         BRB     17$
                                       05  DD 00144 11$:    PUSHL   #5                                         ; 1420
             7E      3C   A7           52  C3 00146         SUBL3   VBN, 60(FCB), -(SP)                        ; 1421
                                30 B742    9F 0014B         PUSHAB  @48(FCB)[VBN]                              ; 1420
                  0000G   CF           03  FB 0014F         CALLS   #3, READ_BLOCK
             51           53           05  78 00154         ASHL    #5, FIRST_REC, R1                          ; 1423
             54           50           51  C1 00158         ADDL3   R1, R0, Q_RECORD
                          50      FF   A3  9E 0015C         MOVAB   -1(R3), J                                  ; 1425
                                       35  11 00160         BRB     16$
             51           52           04  78 00162 12$:    ASHL    #4, VBN, R1                                ; 1428
                     00   BE   01 A041     9E 00166         MOVAB   1(J)[R1], @0(SP)
                          19           64  E9 0016C         BLBC    (Q_RECORD), 14$                            ; 1430
                          07      08   AC  E8 0016F         BLBS    FLAGS, 13$                                 ; 1433
                     04   A4      04   AC  B1 00173         CMPW    UIC, 4(Q_RECORD)
                                       1A  12 00178         BNEQ    15$
             34      08   AC           01  E0 0017A 13$:    BBS     #1, FLAGS, 18$                             ; 1434
                     06   A4      06   AC  B1 0017F         CMPW    UIC+2, 6(Q_RECORD)
                                       0E  12 00184         BNEQ    15$
                                       2B  11 00186         BRB     18$                                        ; 1435
                                02B8   CA  D5 00188 14$:    TSTL    696(BASE)                                  ; 1440
                                       06  12 0018C         BNEQ    15$
                  02B8    CA      00   BE  D0 0018E         MOVL    @0(SP), 696(BASE)                          ; 1441
                          54           20  C0 00194 15$:    ADDL2   #32, Q_RECORD                              ; 1444
             C7           50           0F  F3 00197 16$:    AOBLEQ  #15, J, 12$                                ; 1425
                                       53  D4 0019B         CLRL    FIRST_REC                                  ; 1447
             A3           52           55  F2 0019D 17$:    AOBLSS  R5, VBN, 11$                               ; 1417
                          4C      08   AC  E8 001A1         BLBS    FLAGS, 22$                                 ; 1450
             47      08   AC           01  E0 001A5         BBS     #1, FLAGS, 22$
                                       58  DD 001AA         PUSHL   J                                          ; 1451
                  0000V   CF           01  FB 001AC         CALLS   #1, REL_QUOTA_LOCK
                                       3E  11 001B1         BRB     22$                                        ; 1452
                          69           54  D0 001B3 18$:    MOVL    Q_RECORD, (R9)                             ; 1461
                          05      08   AC  E8 001B6         BLBS    FLAGS, 19$                                 ; 1462
             1C      08   AC           01  E1 001BA         BBC     #1, FLAGS, 20$
                                       7E  D4 001BF 19$:    CLRL    -(SP)                                      ; 1465
                                  04   A4  DD 001C1         PUSHL   4(Q_RECORD)
                  0000V   CF           02  FB 001C4         CALLS   #2, SCAN_QUO_CACHE
                          58           50  D0 001C9         MOVL    R0, J
             50           58           1C  C5 001CC         MULL3   #28, J, R0                                 ; 1466
             05      EF A046           00  E1 001D0         BBC     #0, -17(R0)[QUOTA_LIST], 20$
                                       54  DD 001D6         PUSHL   Q_RECORD                                   ; 1469
                                     FF32  31 001D8         BRW     7$
                                  10   AC  DD 001DB 20$:    PUSHL   USE_CACHE                                  ; 1477
                                       7E  D4 001DE         CLRL    -(SP)
                                       54  DD 001E0         PUSHL   Q_RECORD
                                       58  DD 001E2         PUSHL   J
                  0000V   CF           04  FB 001E4         CALLS   #4, ENTER_QUO_CACHE
             6B           64           20  28 001E9         MOVC3   #32, (Q_RECORD), (R11)                     ; 1478
                          50           5B  D0 001ED 21$:    MOVL    R11, R0                                    ; 1479
                                           04 001F0         RET
                                       50  D4 001F1 22$:    CLRL    R0                                         ; 1481
```

```
                                04 001F3        RET                                     ;

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

```
;  492     1482  1 GLOBAL ROUTINE WRITE_QUOTA (Q_RECORD) : L_NORM NOVALUE =
;  493     1483  1
;  494     1484  1 !++
;  495     1485  1 !
;  496     1486  1 ! FUNCTIONAL DESCRIPTION:
;  497     1487  1 !
;  498     1488  1 !       This routine writes the indicated quota record. If a cache entry
;  499     1489  1 !       exists for the record being processed (indicated by the record
;  500     1490  1 !       being the dummy record), we update the cache entry. If we also
;  501     1491  1 !       have the real quota record in memory, then mark it for write-back.
;  502     1492  1 !
;  503     1493  1 !
;  504     1494  1 ! CALLING SEQUENCE:
;  505     1495  1 !       WRITE_QUOTA (ARG1)
;  506     1496  1 !
;  507     1497  1 ! INPUT PARAMETERS:
;  508     1498  1 !       ARG1: address of quota record
;  509     1499  1 !
;  510     1500  1 ! IMPLICIT INPUTS:
;  511     1501  1 !       REAL_Q_REC: buffer of real quota record if exists
;  512     1502  1 !       QUOTA_INDEX: cache index of cache entry
;  513     1503  1 !
;  514     1504  1 ! OUTPUT PARAMETERS:
;  515     1505  1 !       NONE
;  516     1506  1 !
;  517     1507  1 ! IMPLICIT OUTPUTS:
;  518     1508  1 !       NONE
;  519     1509  1 !
;  520     1510  1 ! ROUTINE VALUE:
;  521     1511  1 !       NONE
;  522     1512  1 !
;  523     1513  1 ! SIDE EFFECTS:
;  524     1514  1 !       quota cache modified, quota record marked for write-back
;  525     1515  1 !
;  526     1516  1 !--
;  527     1517  1
;  528     1518  2 BEGIN
;  529     1519  2
;  530     1520  2 MAP
;  531     1521  2         Q_RECORD        : REF BBLOCK;   ! address of quota record
;  532     1522  2
;  533     1523  2 BIND_COMMON;
;  534     1524  2
;  535     1525  2 EXTERNAL ROUTINE
;  536     1526  2         MARK_DIRTY      : L_NORM;       ! mark buffer for write back
;  537     1527  2
;  538     1528  2
;  539     1529  2 ! If the specified record is the dummy record, there is a cache entry.
;  540     1530  2 ! Therefore, update it. Also update the associated real record if there
;  541     1531  2 ! is one.
;  542     1532  2 !
;  543     1533  2
;  544     1534  2 IF .Q_RECORD EQL DUMMY_REC
;  545     1535  2 THEN
;  546     1536  3     BEGIN
;  547     1537  3     ENTER_QUO_CACHE (.QUOTA_INDEX, .Q_RECORD, .REAL_Q_REC EQL 0, 2);
;  548     1538  3     IF .REAL_Q_REC NEQ 0
```

```
;  549          1539  3     THEN
;  550          1540  4         BEGIN
;  551          1541  4         CH$MOVE (DQF$C_LENGTH, .Q_RECORD, .REAL_Q_REC);
;  552          1542  4         MARK_DIRTY (.REAL_Q_REC);
;  553          1543  3         END;
;  554          1544  3     END
;  555          1545  3
;  556          1546  3 ! Otherwise, if there is no cache entry, we just have to mark the
;  557          1547  3 ! buffer dirty.
;  558          1548  3 !
;  559          1549  3
;  560          1550  2 ELSE
;  561          1551  2     MARK_DIRTY (.Q_RECORD);
;  562          1552  2
;  563          1553  1 END;                                    ! end of routine WRITE_QUOTA


                                                        .EXTRN  MARK_DIRTY

                                        007C 00000      .ENTRY  WRITE_QUOTA, Save R2,R3,R4,R5,R6        ; 1482
                                    56  02BC CA  9E 00002       MOVAB   700(BASE), R6                   ; 1521
                                    50  02C4 CA  9E 00007       MOVAB   708(BASE), R0                   ; 1534
                                    50    04 AC  D1 0000C       CMPL    Q_RECORD, R0                    ;
                                              24  12 00010       BNEQ    2$                              ;
                                              02  DD 00012       PUSHL   #2                              ; 1537
                                              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$:    PUSHL   Q_RECORD                        ;
                                    02C0      CA  DD 0001F       PUSHL   704(BASE)                       ;
                                        0000V CF      04  FB 00023       CALLS   #4, ENTER_QUO_CACHE     ;
                                              66  D5 00028       TSTL    (R6)                            ; 1538
                                              12  13 0002A       BEQL    4$                              ;
                                00    B6      04  BC            20  28 0002C       MOVC3   #32, @Q_RECORD, @0(R6)  ; 1541
                                              66  DD 00032       PUSHL   (R6)                            ; 1542
                                              03  11 00034       BRB     3$                              ;
                                      04      AC  DD 00036 2$:    PUSHL   Q_RECORD                        ; 1551
                                        0000G CF      01  FB 00039 3$:    CALLS   #1, MARK_DIRTY          ;
                                              04  0003E 4$:     RET                                     ; 1553

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

```
;  565     1554  1 ROUTINE SCAN_QUO_CACHE (UIC, MARK_USE) : L_NORM =
;  566     1555  1
;  567     1556  1 !++
;  568     1557  1 !
;  569     1558  1 ! FUNCTIONAL DESCRIPTION:
;  570     1559  1 !
;  571     1560  1 !       This routine scans the quota cache for the indicated UIC. If found,
;  572     1561  1 !       it returns the contents, and marks the entry used if requested.
;  573     1562  1 !
;  574     1563  1 !
;  575     1564  1 ! CALLING SEQUENCE:
;  576     1565  1 !       SCAN_QUO_CACHE (ARG1, ARG2)
;  577     1566  1 !
;  578     1567  1 ! INPUT PARAMETERS:
;  579     1568  1 !       ARG1: UIC to search for
;  580     1569  1 !       ARG2: 1 to record new use
;  581     1570  1 !             0 to not
;  582     1571  1 !
;  583     1572  1 ! IMPLICIT INPUTS:
;  584     1573  1 !       CURRENT_VCB: VCB of volume
;  585     1574  1 !
;  586     1575  1 ! OUTPUT PARAMETERS:
;  587     1576  1 !       NONE
;  588     1577  1 !
;  589     1578  1 ! IMPLICIT OUTPUTS:
;  590     1579  1 !       DUMMY_REC: receives contents of cache entry if found
;  591     1580  1 !       QUOTA_INDEX: receives index of cache entry found
;  592     1581  1 !       QUOTA_RECORD: quota file record number of found entry
;  593     1582  1 !
;  594     1583  1 ! ROUTINE VALUE:
;  595     1584  1 !       index of entry found
;  596     1585  1 !
;  597     1586  1 ! SIDE EFFECTS:
;  598     1587  1 !       quota cache entry modified
;  599     1588  1 !
;  600     1589  1 !--
;  601     1590  1
;  602     1591  2 BEGIN
;  603     1592  2
;  604     1593  2 LITERAL
;  605     1594  2         RECS_PER_BLOCK  = 512 / DQF$C_LENGTH;
;  606     1595  2
;  607     1596  2 LABEL
;  608     1597  2         QUOTA_SEARCH;                   ! body of search code
;  609     1598  2
;  610     1599  2 LOCAL
;  611     1600  2         QUOTA_CACHE     : REF BBLOCK,   ! address of quota cache
;  612     1601  2         QUOTA_LIST      : REF BBLOCKVECTOR [,VCA$C_QUOLENGTH],
;  613     1602  2                                         ! address of quota cache entries
;  614     1603  2         J,                              ! index into quota cache
;  615     1604  2         LOWEST_LRU,                     ! oldest quota LRU index
;  616     1605  2         LOWEST_J,                       ! oldest quota cache entry index
;  617     1606  2         LRU_DELTA,                      ! LRU index of current entry
;  618     1607  2         OLD_RECORD      : REF BBLOCK,   ! address of old quota record
;  619     1608  2         REC_NUM,                        ! quota file record to read
;  620     1609  2         FCB             : REF BBLOCK;   ! address of quota file FCB
;  621     1610  2
```

```
;  622          1611  2 EXTERNAL
:001 !CDS0005   1612  2         CLU$GL_CLUB     : ADDRESSING_MODE (GENERAL),
;  623          1613  2         PMS$GL_QUOHIT   : ADDRESSING_MODE (GENERAL),
;  624          1614  2                                         ! count of quota cache hits
;  625          1615  2         PMS$GL_QUOMISS  : ADDRESSING_MODE (GENERAL);
;  626          1616  2                                         ! count of quota cache misses
;  627          1617  2
;  628          1618  2 EXTERNAL ROUTINE
;  629          1619  2         CACHE_LOCK      : L_NORM,       ! acquire cache lock
;  630          1620  2         READ_BLOCK      : L_NORM;       ! read a disk block
;  631          1621  2
;  632          1622  2
;  633          1623  2 BIND_COMMON;
;  634          1624  2
;  635          1625  2
;  636          1626  2 ! If the cache is not currently marked valid, do so if possible.
;  637          1627  2 ! This involves taking out the cache lock if the volume is cluster
;  638          1628  2 ! accessible, and checking for quota file writers and a non-null
;  639          1629  2 ! cache size.
;  640          1630  2 !
;  641          1631  2
;  642          1632  2 QUOTA_CACHE = .CURRENT_VCB[VCB$L_QUOCACHE];
;  643          1633  2 FCB = .CURRENT_VCB[VCB$L_QUOTAFCB];
;  644          1634  2
;  645          1635  2 IF NOT .QUOTA_CACHE[VCA$V_CACHEVALID]
;  646          1636  2 AND NOT .QUOTA_CACHE[VCA$V_CACHEFLUSH]
;  647          1637  2 THEN
;  648          1638  2     BEGIN
;  649          1639  2     IF  .QUOTA_CACHE[VCA$W_QUOSIZE] GTRU 1
;  650          1640  2     AND NOT .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_DMT]
;  651          1641  2     AND .FCB[FCB$W_WCNT] LEQU 1
;  652          1642  3     AND
;  653          1643  4         BEGIN
;  654          1644  4         IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
:001 !CDS0005   1645  4             AND .CLU$GL_CLUB NEQ 0
;  655          1646  4         THEN CACHE_LOCK (.FCB[FCB$L_LOCKBASIS], QUOTA_CACHE[VCA$L_QUOCLKID], 0)
;  656          1647  4         ELSE 1
;  657          1648  4         END
;  658          1649  3     THEN
;  659          1650  3         QUOTA_CACHE[VCA$V_CACHEVALID] = 1
;  660          1651  3     ELSE
;  661          1652  3         QUOTA_CACHE[VCA$V_CACHEFLUSH] = 1;
;  662          1653  3     END;
;  663          1654  2
;  664          1655  2 ! Search the quota cache for an active entry with a matching UIC.
;  665          1656  2 !
;  666          1657  2
;  667          1658  2 QUOTA_SEARCH: BEGIN
;  668          1659  2
;  669          1660  2 QUOTA_LIST = QUOTA_CACHE[VCA$L_QUOLIST];
;  670          1661  2 INCR R FROM 1 TO .QUOTA_CACHE[VCA$W_QUOSIZE]
;  671          1662  3 DO
;  672          1663  4     BEGIN
;  673          1664  4     IF  .QUOTA_LIST[.K-1, VCA$L_QUORECNUM] NEQ 0
;  674          1665  4     AND .QUOTA_LIST[.K-1, VCA$L_QUOUIC] EQL .UIC
;  675          1666  4     THEN
;  676          1667  5         BEGIN
```

```
;  677          1668  5          IF .MARK_USE
;  678          1669  5          THEN
;  679          1670  6              BEGIN
;  680          1671  6              QUOTA_LIST[.K-1, VCA$W_QUOLRUX] = .QUOTA_CACHE[VCA$W_QUOLRU];
;  681          1672  6              QUOTA_CACHE[VCA$W_QUOLRU] = .QUOTA_CACHE[VCA$W_QUOLRU] + 1;
;  682          1673  5              END;
;  683          1674  5          PMS$GL_QUOHIT = .PMS$GL_QUOHIT + 1;
;  684          1675  5          J = .K;
;  685          1676  5          LEAVE QUOTA_SEARCH;
;  686          1677  4          END;
;  687          1678  3      END;
;  688          1679  3
;  689          1680  3  ! We failed to find a match in the quota cache. Search the cache for a free
;  690          1681  3  ! entry, or, failing that, the entry with the oldest LRU index.
;  691          1682  3  !
;  692          1683  3
;  693          1684  3  PMS$GL_QUOMISS = .PMS$GL_QUOMISS + 1;
;  694          1685  3  LOWEST_LRU = 0;
;  695          1686  3  LOWEST_J = 1;
;  696          1687  3  INCR J FROM 1 TO .QUOTA_CACHE[VCA$W_QUOSIZE]
;  697          1688  3  DO
;  698          1689  4      BEGIN
;  699          1690  4      IF .QUOTA_LIST[.J-1, VCA$L_QUORECNUM] EQL 0
;  700          1691  4      THEN
;  701          1692  5          BEGIN
;  702          1693  5          LOWEST_J = .J;
;  703          1694  5          EXITLOOP;
;  704          1695  4          END;
;  705          1696  4      LRU_DELTA = .QUOTA_CACHE[VCA$W_QUOLRU] - .QUOTA_LIST[.J-1, VCA$W_QUOLRUX];
;  706          1697  4      IF .LRU_DELTA GTRU .LOWEST_LRU
;  707          1698  4      THEN
;  708          1699  5          BEGIN
;  709          1700  5          LOWEST_LRU = .LRU_DELTA;
;  710          1701  5          LOWEST_J = .J;
;  711          1702  4          END;
;  712          1703  3      END;
;  713          1704  3
;  714          1705  3  ! If the cache entry we are about to use contains a modified entry, we must
;  715          1706  3  ! read the corresponding record, update it, and write it. If it represents a
;  716          1707  3  ! held lock, we must release it.
;  717          1708  3  !
;  718          1709  3
;  719          1710  3  J = .LOWEST_J;
;  720          1711  3  IF  .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  721          1712  3  AND .QUOTA_LIST[.J-1, VCA$V_QUODIRTY]
;  722          1713  3  THEN
;  723          1714  4      BEGIN
;  724          1715  4      REC_NUM = .QUOTA_LIST[.J-1, VCA$L_QUORECNUM] - 1;
;  725          1716  4      OLD_RECORD = READ_BLOCK (.REC_NUM / RECS_PER_BLOCK
;  726          1717  4                              + .FCB[FCB$L_ST[BN], 1, QUOTA_TYPE)
;  727          1718  4                      + (.REC_NUM MOD RECS_PER_BLOCK) * DQF$C_LENGTH;
;  728          1719  4      CLEAN_QUO_CACHE (.J, .OLD_RECORD);
;  729          1720  3      END;
;  730          1721  3  REL_QUOTA_LOCK (.J);
;  731          1722  2  END;                                     ! end of block QUOTA_SEARCH
;  732          1723  2
;  733          1724  2  ! If the quota cache entry is not marked valid, take out the lock on it.
```

```
;  734          1725  2 ! If thereafter it is valid, fill in the dummy record with its contents.
;  735          1726  2 !
;  736          1727  2
;  737          1728  2 IF NOT .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  738          1729  2 THEN
;  739          1730  2     BEGIN
;  740          1731  2     QUOTA_LIST[.J-1, VCA$L_QUOUIC] = .UIC;
;  741          1732  2     GET_QUOTA_LOCK (.J, LCK$K_PWMODE);
;  742          1733  2     END;
;  743          1734  2 IF .QUOTA_LIST[.J-1, VCA$V_QUOVALID]
;  744          1735  2 THEN
;  745          1736  2     BEGIN
;  746          1737  2     DUMMY_REC[DQF$L_FLAGS] = DQF$M_ACTIVE;
;  747          1738  2     QUOTA_RECORD = .QUOTA_LIST[.J-1, VCA$L_QUORECNUM];
;  748          1739  2     DUMMY_REC[DQF$L_UIC] = .QUOTA_LIST[.J-1, VCA$L_QUOUIC];
;  749          1740  2     CH$MOVE (12, QUOTA_LIST[.J-1, VCA$L_USAGE], DUMMY_REC[DQF$L_USAGE]);
;  750          1741  2     END;
;  751          1742  2 QUOTA_INDEX = .J;
;  752          1743  2
;  753          1744  2 .J
;  754          1745  1 END;                                   ! end of routine SCAN_QUO_CACHE


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

                                OBFC 00000 SCAN_QUO_CACHE:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R11        ; 1554
                             5E                 04  C2 00002                SUBL2   #4, SP          ;
                             5B         02C4    CA  9E 00005                MOVAB   708(BASE), R11  ; 1620
                             50           98    AA  D0 0000A                MOVL    -104(BASE), R0  ; 1632
                             53           5C    A0  D0 0000E                MOVL    92(R0), QUOTA_CACHE
                             50           98    AA  D0 00012                MOVL    -104(BASE), R0  ; 1633
                             54           54    A0  D0 00016                MOVL    84(R0), FCB     ;
                             43           0B    A3  E8 0001A                BLBS    11(QUOTA_CACHE), 3$ ; 1635
                      3E     0B    A3           01  E0 0001E                BBS     #1, 11(QUOTA_CACHE), 3$ ; 1636
                             01                 63  B1 00023                CMPW    (QUOTA_CACHE), #1 ; 1639
                                                35  1B 00026                BLEQU   2$              ;
                             50           94    AA  D0 00028                MOVL    -108(BASE), R0  ; 1640
                      2C     3A    A0           05  E0 0002C                BBS     #5, 58(R0), 2$  ;
                             01           1C    A4  B1 00031                CMPW    28(FCB), #1     ; 1641
                                                26  1A 00035                BGTRU   2$              ;
                             50           94    AA  D0 00037                MOVL    -108(BASE), R0  ; 1644
                             18           3C    A0  E9 0003B                BLBC    60(R0), 1$      ;
                                  00000000G      00  D5 0003F                TSTL    CLU$GL_CLUB     ; 1645
                                                10  13 00045                BEQL    1$              ;
                                                7E  D4 00047                CLRL    -(SP)           ; 1646
                                           04   A3  9F 00049                PUSHAB  4(QUOTA_CACHE)  ;
                                           4C   A4  DD 0004C                PUSHL   76(FCB)         ;
                                0000G      CF        03  FB 0004F                CALLS   #3, CACHE_LOCK  ;
                                           06        50  E9 00054                BLBC    R0, 2$          ;
                                0B         A3        01  88 00057 1$:          BISB2   #1, 11(QUOTA_CACHE) ; 1650
                                                     04  11 0005B              BRB     3$              ;
                                0B         A3        02  88 0005D 2$:          BISB2   #2, 11(QUOTA_CACHE) ; 1652
                             52           44    A3  9E 00061 3$:          MOVAB   68(R3), QUOTA_LIST ; 1660
                             55                 63  3C 00065                MOVZWL  (QUOTA_CACHE), R5 ; 1661
```

```
                                     50  D4 00068       CLRL    K                               ;
                                     2E  11 0006A       BRB     6$                              ;
            51             50        1C  C5 0006C 4$:   MULL3   #28, K, R1                      ; 1664
                           51        52  C0 00070       ADDL2   QUOTA_LIST, R1                  ;
00      EC  A1             18        00  ED 00073       CMPZV   #0, #24, -20(R1), #0            ;
                                     1F  13 00079       BEQL    6$                              ;
                    04     AC    FC  A1  D1 0007B       CMPL    -4(R1), UIC                     ; 1665
                                     18  12 00080       BNEQ    6$                              ;
                           08    08  AC  E9 00082       BLBC    MARK_USE, 5$                    ; 1668
                    E6     A1    02  A3  B0 00086       MOVW    2(QUOTA_CACHE), -26(R1)         ; 1671
                                 02  A3  B6 0008B       INCW    2(QUOTA_CACHE)                  ; 1672
                          00000000G  00  D6 0008E 5$:   INCL    PMS$GL_QUOHIT                   ; 1674
                           56        50  D0 00094       MOVL    K, J                            ; 1675
                                   0090  31 00097       BRW     12$                             ; 1676
            CE             50        55  F3 0009A 6$:   AOBLEQ  R5, K, 4$                       ; 1661
                          00000000G  00  D6 0009E       INCL    PMS$GL_QUOMISS                  ; 1684
                                     58  D4 000A4       CLRL    LOWEST_LRU                      ; 1685
                           55        01  D0 000A6       MOVL    #1, LOWEST_J                    ; 1686
                           59        63  3C 000A9       MOVZWL  (QUOTA_CACHE), R9               ; 1687
                                     50  D4 000AC       CLRL    J                               ; 1696
                                     2A  11 000AE       BRB     9$                              ;
            51             50        1C  C5 000B0 7$:   MULL3   #28, J, R1                      ; 1690
                           51        52  C0 000B4       ADDL2   QUOTA_LIST, R1                  ;
00      EC  A1             18        00  ED 000B7       CMPZV   #0, #24, -20(R1), #0            ;
                                     05  12 000BD       BNEQ    8$                              ;
            55                       50  D0 000BF       MOVL    J, LOWEST_J                     ; 1693
                                     1A  11 000C2       BRB     10$                             ; 1692
                           57    02  A3  3C 000C4 8$:   MOVZWL  2(QUOTA_CACHE), LRU_DELTA       ; 1696
                           6E    E6  A1  3C 000C8       MOVZWL  -26(R1), (SP)                   ;
                           57        6E  C2 000CC       SUBL2   (SP), LRU_DELTA                 ;
                           58        57  D1 000CF       CMPL    LRU_DELTA, LOWEST_LRU           ; 1697
                                     06  1B 000D2       BLEQU   9$                              ;
                           58        57  D0 000D4       MOVL    LRU_DELTA, LOWEST_LRU           ; 1700
                           55        50  D0 000D7       MOVL    J, LOWEST_J                     ; 1701
            D2             50        59  F3 000DA 9$:   AOBLEQ  R9, J, 7$                       ; 1687
                           56        55  D0 000DE 10$:  MOVL    LOWEST_J, J                     ; 1710
            50             56        1C  C5 000E1       MULL3   #28, J, R0                      ; 1711
                           50        52  C0 000E5       ADDL2   QUOTA_LIST, R0                  ;
                           37    EF  A0  E9 000E8       BLBC    -17(R0), 11$                    ;
            32      EF     A0        01  E1 000EC       BBC     #1, -17(R0), 11$                ; 1712
55      EC  A0             18        00  EF 000F1       EXTZV   #0, #24, -20(R0), REC_NUM       ; 1715
                                     55  D7 000F7       DECL    REC_NUM                         ;
                                     05  DD 000F9       PUSHL   #5                              ; 1716
                                     01  DD 000FB       PUSHL   #1                              ;
            50             55        10  C7 000FD       DIVL3   #16, REC_NUM, R0                ;
                              30 B440    9F 00101       PUSHAB  @48(FCB)[R0]                    ; 1717
                    0000G  CF        03  FB 00105       CALLS   #3, READ_BLOCK                  ;
7E          00             55        01  7A 0010A       EMUL    #1, REC_NUM, #0, -(SP)          ; 1718
51          51             8E        10  7B 0010F       EDIV    #16, (SP)+, R1, R1              ;
                           51        20  C4 00114       MULL2   #32, R1                         ;
                           50        51  C0 00117       ADDL2   R1, OLD_RECORD                  ;
                                     50  DD 0011A       PUSHL   OLD_RECORD                      ; 1719
                                     56  DD 0011C       PUSHL   J                               ;
                    0000V  CF        02  FB 0011E       CALLS   #2, CLEAN_QUO_CACHE             ;
                                     56  DD 00123 11$:  PUSHL   J                               ; 1721
                    0000V  CF        01  FB 00125       CALLS   #1, REL_QUOTA_LOCK              ;
            50             56        1C  C5 0012A 12$:  MULL3   #28, J, R0                      ; 1728
```

```
                                50          52  C0 0012E           ADDL2   QUOTA_LIST, R0                                          ;
                                0E      EF  A0  E8 00131           BLBS    -17(R0), 13$                                            ;
                             FC     A0  04  AC  D0 00135           MOVL    UIC, -4(R0)                                             ; 1731
                                            04  DD 0013A           PUSHL   #4                                                      ; 1732
                                            56  DD 0013C           PUSHL   J
                           0000V    CF              02  FB 0013E   CALLS   #2, GET_QUOTA_LOCK                                      ;
                       50              56          1C  C5 00143 13$: MULL3   #28, J, R0                                             ; 1734
                       27          EF A042         00  E1 00147   BBC     #0, -17(R0)[QUOTA_LIST], 14$                            ;
                                    6B             01  D0 0014D   MOVL    #1, (R11)                                               ; 1737
                       50           56             1C  C5 00150   MULL3   #28, J, R0                                              ; 1738
    02B4    CA      EC A042         18             00  EF 00154   EXTZV   #0, #24, -20(R0)[QUOTA_LIST], 692(BASE)                 ;
                       50           56             1C  C5 0015D   MULL3   #28, J, R0                                              ; 1739
                                                FC A042 9F 00161   PUSHAB  -4(R0)[QUOTA_LIST]                                      ;
                             04     AB              9E  D0 00165   MOVL    @(SP)+, 4(R11)                                          ;
                       50           56             1C  C5 00169   MULL3   #28, J, R0                                              ; 1740
                 08    AB        F0 A042         0C  28 0016D   MOVC3   #12, -16(R0)[QUOTA_LIST], 8(R11)                         ;
                           02C0     CA             56  D0 00174 14$: MOVL    J, 704(BASE)                                            ; 1742
                                    50             56  D0 00179   MOVL    J, R0                                                   ; 1745
                                                       04 0017C   RET                                                             ;

; Routine Size:  381 bytes,    Routine Base:  $CODE$ + 02BF
```

```
;  756          1746  1 GLOBAL ROUTINE GET_QUOTA_LOCK (J, MODE) : L_NORM NOVALUE =
;  757          1747  1
;  758          1748  1 !++
;  759          1749  1 !
;  760          1750  1 ! FUNCTIONAL DESCRIPTION:
;  761          1751  1 !
;  762          1752  1 !       This routine acquires the lock associated with a quota cache
;  763          1753  1 !       entry. The lock is raised to PW, and the value block is stored
;  764          1754  1 !       in the quota cache entry.
;  765          1755  1 !
;  766          1756  1 ! CALLING SEQUENCE:
;  767          1757  1 !       GET_QUOTA_LOCK (J, MODE)
;  768          1758  1 !
;  769          1759  1 ! INPUT PARAMETERS:
;  770          1760  1 !       J: index of quota cache entry
;  771          1761  1 !       MODE: lock mode to use
;  772          1762  1 !
;  773          1763  1 ! IMPLICIT INPUTS:
;  774          1764  1 !       CURRENT_VCB: VCB of volume
;  775          1765  1 !       CURRENT_RVT: RVT of volume set
;  776          1766  1 !
;  777          1767  1 ! OUTPUT PARAMETERS:
;  778          1768  1 !       NONE
;  779          1769  1 !
;  780          1770  1 ! IMPLICIT OUTPUTS:
;  781          1771  1 !       NONE
;  782          1772  1 !
;  783          1773  1 ! ROUTINE VALUE:
;  784          1774  1 !       NONE
;  785          1775  1 !
;  786          1776  1 ! SIDE EFFECTS:
;  787          1777  1 !       Lock taken out; value block written into cache entry.
;  788          1778  1 !
;  789          1779  1 !--
;  790          1780  1
;  791          1781  2 BEGIN
;  792          1782  2
;  793          1783  2 LOCAL
;  794          1784  2         CACHE_ENTRY     : REF BBLOCK,   ! quota cache entry pointer
;  795          1785  2         STATUS,                         ! general status value
;  796          1786  2         LOCK_FLAGS      : BBLOCK [4],   ! flags to $ENQ call
;  797          1787  2         SAVE_LRU,                       ! save cache entry LRU index
;  798          1788  2         RESNAM          : VECTOR [22, BYTE], ! resource name buffer
;  799          1789  2         RESNAM_D        : VECTOR [2] INITIAL (22, RESNAM);
;  800          1790  2
;001  !CDS0005  1791  2 EXTERNAL
;002  !CDS0005  1792  2         CLU$GL_CLUB     : ADDRESSING_MODE (GENERAL);
;003  !CDS0005  1793  2
;  801          1794  2 EXTERNAL ROUTINE
;  802          1795  2         WAIT_FOR_AST    : L_NORM,       ! wait for completion AST
;  803          1796  2         CONTINUE_THREAD : L_NORM,       ! continue execution thread
;  804          1797  2         XQP$REL_QUOTA : ADDRESSING_MODE (GENERAL);
;  805          1798  2                                         ! unlock cache entry on blocking AST
;  806          1799  2
;  807          1800  2 BIND_COMMON;
;  808          1801  2
;  809          1802  2 ! If the volume is not cluster accessible, we don't have to bother with
```

```
;   810          1803  2 ! locks.
;   811          1804  2 !
;   812          1805  2
;   813          1806  2 IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_ALL]
;   814          1807  2 OR NOT .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
;001 !CDS0005    1808  2 OR CLU$GL_CLUB EQL 0
;   815          1809  2 THEN RETURN;
;   816          1810  2
;   817          1811  2
;   818          1812  2 ! See if we have a lock ID for the cache entry. If so, this is just a
;   819          1813  2 ! conversion. Otherwise, generate the resource name, using the facility
;   820          1814  2 ! prefix and the volume or volume set name.
;   821          1815  2 !
;   822          1816  2
;   823          1817  2 CACHE_ENTRY = BBLOCKVECTOR [BBLOCK [.CURRENT_VCB[VCB$L_QUOCACHE],
;   824          1818                   VCA$L_QUOLIST], .J-1, VCA$R_QUOLOCK; ,VCA$C_QUOLENGTH];
;   825          1819  2
;   826          1820  2 LOCK_FLAGS = LCK$M_SYSTEM + LCK$M_VALBLK + LCK$M_NOQUOTA;
;   827          1821  2
;   828          1822  2 IF .CACHE_ENTRY[VCA$L_QUOLKID] NEQ 0
;   829          1823  2 THEN LOCK_FLAGS[LCK$V_CONVERT] = 1
;   830          1824  2
;   831          1825  2 ELSE
;   832          1826  3     BEGIN
;   833          1827  3     CH$MOVE (6, UPLIT BYTE ('F11B$q'), RESNAM[0]);
;   834          1828  3     CH$MOVE (12,
;   835          1829  3              IF .CURRENT_VCB[VCB$W_RVN] EQL 0
;   836          1830  3              THEN CURRENT_VCB[VCB$T_VOLCKNAM]
;   837          1831  3              ELSE CURRENT_RVT[RVT$T_VLSLCKNAM],
;   838          1832  3              RESNAM[6]);
;   839          1833  3     (RESNAM[18]) = .CACHE_ENTRY[VCA$L_QUOUIC];
;   840          1834  3     END;
;   841          1835  2
;   842          1836  2 ! Acquire the lock.
;   843          1837  2 !
;   844          1838  2
;   845          1839  2 SAVE_LRU = .CACHE_ENTRY[VCA$W_QUOLRUX];
;   846        P 1840  2 STATUS = $ENQ (EFN    = EFN,
;   847        P 1841  2                LKMODE = .MODE,
;   848        P 1842  2                FLAGS  = .LOCK_FLAGS,
;   849        P 1843  2                LKSB   = CACHE_ENTRY[VCA$R_QUOLOCK],
;   850        P 1844  2                ASTADR = CONTINUE_THREAD,
;   851        P 1845  2                ASTPRM = .BASE,
;   852        P 1846  2                RESNAM = RESNAM_D
;   853          1847  2                );
;   854          1848  2 IF NOT .STATUS
;   855          1849  2 THEN
;   856          1850  3     BEGIN
;   857          1851  3     CH$FILL (0, VCA$C_QUOLENGTH, .CACHE_ENTRY);
;   858          1852  3     IF .LOCK_FLAGS[LCK$V_CONVERT]
;   859          1853  4     THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error')
;   860          1854  3     ELSE ERR_EXIT (.STATUS);
;   861          1855  2     END;
;   862          1856  2
;   863          1857  2 IF .STATUS EQL SS$_NORMAL
;   864          1858  2 THEN WAIT_FOR_AST ();
;   865          1859  2
```

```
;  866          1860  2 ! Deal with lock completion and handle any errors. If the lock comes back
;  867          1861  2 ! with value not valid, turn off the valid bit but preserve the contents.
;  868          1862  2 ! We will wtill use the record number to avoid a complete search.
;  869          1863  2 !
;  870          1864  2
;  871          1865  2 STATUS = .CACHE_ENTRY[VCA$W_QUOSTATUS];
;  872          1866  2
;  873          1867  2 IF NOT .STATUS
;  874          1868  2 THEN
;  875          1869  3     BEGIN
;  876          1870  3     IF .STATUS EQL SS$_VALNOTVALID
;  877          1871  3     THEN CACHE_ENTRY[VCA$V_QUOVALID] = 0
;  878          1872  3     ELSE
;  879          1873  4         BEGIN
;  880          1874  4         CH$FILL (0, VCA$C_QUOLENGTH, .CACHE_ENTRY);
;  881          1875  4         IF .LOCK_FLAGS[LCK$V_CONVERT]
;  882          1876  5         THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error')
;  883          1877  4         ELSE ERR_EXIT (.STATUS);
;  884          1878  3         END;
;  885          1879  2     END;
;  886          1880  2
;  887          1881  2 ! Having acquired the lock, convert it to system owned.
;  888          1882  2 !
;  889          1883  2
;  890        P 1884  2 STATUS = $ENQ (EFN    = EFN,
;  891        P 1885  2                LKMODE = .MODE,
;  892        P 1886  2                FLAGS  = LCK$M_NOQUEUE OR LCK$M_SYNCSTS OR LCK$M_CVTSYS OR LCK$M_CONVERT,
;  893        P 1887  2                LKSB   = CACHE_ENTRY[VCA$R_QUOLOCK],
;  894        P 1888  2                BLKAST = XQP$REL_QUOTA,
;  895        P 1889  2                ASTPRM = .CACHE_ENTRY
;  896          1890  2                );
;  897          1891  2 IF .STATUS
;  898          1892  2 THEN STATUS = .CACHE_ENTRY[VCA$W_QUOSTATUS];
;  899          1893  2 IF NOT .STATUS
;  900          1894  2 THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error');
;  901          1895  2
;  902          1896  2 CACHE_ENTRY[VCA$W_QUOINDEX] = .J;
;  903          1897  2 CACHE_ENTRY[VCA$W_QUOLRUX] = .SAVE_LRU;
;  904          1898  2
;  905          1899  1 END;                                   ! End of routine GET_QUOTA_LOCK


                                71 24 42 31 31 46  0043C 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,- ; 1746
                                                                         R9,R11
                                   5B 00000000G  00  9E 00002    MOVAB   SYS$ENQ, R11                                ;
                                   5E            1C  C2 00009    SUBL2   #28, SP                                      ;
                                                 16  DD 0000C    PUSHL   #22                                          ; 1781
                                04 AE       08    AE  9E 0000E  MOVAB   RESNAM, RESNAM_D+4                           ;
                                   50       94    AA  D0 00013  MOVL    -108(BASE), R0                               ; 1806
                                            3A    A0  95 00017  TSTB    58(R0)                                       ;
```

```
                                                01  18 0001A          BGEQ    1$
                                                    04 0001C          RET
                                        01       3C  A0  E8 0001D 1$:  BLBS    60(R0), 2$                                  1807
                                                    04 00021          RET
                                        50 00000000G 00  9E 00022 2$:  MOVAB   CLU$GL_CLUB, R0                             1808
                                                01  12 00029          BNEQ    3$
                                                    04 0002B          RET
                                        50       98  AA  D0 0002C 3$:  MOVL    -104(BASE), R0                              1817
                                56      04       AC      1C C5 00030       MULL3   #28, J, R6                              1818
                                        56       5C  A0  C0 00035       ADDL2   92(R0), R6
                                        56           28  C0 00039       ADDL2   #40, CACHE_ENTRY
                                        58           31  D0 0003C       MOVL    #49, LOCK_FLAGS                             1820
                                                 04  A6  D5 0003F       TSTL    4(CACHE_ENTRY)                              1822
                                                     05  13 00042       BEQL    4$
                                        58           02  88 00044       BISB2   #2, LOCK_FLAGS                              1823
                                                     25  11 00047       BRB     7$
                08      AE      AD      AF           06  28 00049 4$:  MOVC3   #6, P.AAA, RESNAM                          1827
                                        50       98  AA  D0 0004F       MOVL    -104(BASE), R0                              1829
                                                 0E  A0  B5 00053       TSTW    14(R0)
                                                     07  12 00056       BNEQ    5$
                                        50       0080    C0  9E 00058       MOVAB   128(R0), R0                             1830
                                                     05  11 0005D       BRB     6$                                          1831
                        50      9C      AA           18  C1 0005F 5$:  ADDL3   #24, -100(BASE), R0
                0E      AE              60           0C  28 00064 6$:  MOVC3   #12, (R0), RESNAM+6                         1832
                                1A      AE       18  A6  D0 00069       MOVL    24(CACHE_ENTRY), RESNAM+18                  1833
                                        59       02  A6  3C 0006E 7$:  MOVZWL  2(CACHE_ENTRY), SAVE_LRU                    1839
                                                     7E  7C 00072       CLRQ    -(SP)                                       1847
                                                     7E  D4 00074       CLRL    -(SP)
                                                     5A  DD 00076       PUSHL   BASE
                                                0000G    CF  9F 00078       PUSHAB  CONTINUE_THREAD
                                                     7E  D4 0007C       CLRL    -(SP)
                                                     18  AE  9F 0007E       PUSHAB  RESNAM_D
                                                   0140  8F  BB 00081       PUSHR   #^M<R6,R8>
                                                     08  AC  DD 00085       PUSHL   MODE
                                                     1E  DD 00088       PUSHL   #30
                                            6B           0B  FB 0008A       CALLS   #11, SYS$ENQ
                                            57           50  D0 0008D       MOVL    R0, STATUS
                                            0E           57  E8 00090       BLBS    STATUS, 8$                              1848
                1C      00              6E           00  2C 00093       MOVC5   #0, (SP), #0, #28, (CACHE_ENTRY)            1851
                                                     66      00098
                        33              58           01  E1 00099       BBC     #1, LOCK_FLAGS, 11$                         1852
                                                       FEFF  0009D       BUGW                                                1853
                                                      0000*  0009F       .WORD   <BUG$_XQPERR!4>
                                            01           57  D1 000A1 8$:  CMPL    STATUS, #1                                1857
                                                     05  12 000A4       BNEQ    9$
                                    0000G   CF           00  FB 000A6       CALLS   #0, WAIT_FOR_AST                        1858
                                            57           66  3C 000AB 9$:  MOVZWL  (CACHE_ENTRY), STATUS                   1865
                                            22           57  E8 000AE       BLBS    STATUS, 12$                             1867
                            000009F0        8F           57  D1 000B1       CMPL    STATUS, #2544                           1870
                                                     06  12 000B8       BNEQ    10$
                                    0B      A6           01  8A 000BA       BICB2   #1, 11(CACHE_ENTRY)                     1871
                                                     13  11 000BE       BRB     12$
                1C      00              6E           00  2C 000C0 10$:  MOVC5   #0, (SP), #0, #28, (CACHE_ENTRY)           1874
                                                     66      000C5
                        06              58           01  E1 000C6       BBC     #1, LOCK_FLAGS, 11$                         1875
                                                       FEFF  000CA       BUGW                                                1876
                                                      0000*  000CC       .WORD   <BUG$_XQPERR!4>
```

```
                                    03  11 000CE             BRB     12$                                     ;
                                    57  BF 000D0 11$:        CHMU    STATUS                                  ; 1877
                                        04 000D2             RET                                             ;
                                    7E  7C 000D3 12$:        CLRQ    -(SP)                                   ; 1890
                         00000000G  00  9F 000D5             PUSHAB  XQP$REL_QUOTA                           ;
                                    56  DD 000DB             PUSHL   CACHE_ENTRY                             ;
                                    7E  7C 000DD             CLRQ    -(SP)                                   ;
                                    7E  D4 000DF             CLRL    -(SP)                                   ;
                              7E        8F 9A 000E1          MOVZBL  #78, -(SP)                              ;
                                    56  DD 000E5             PUSHL   CACHE_ENTRY                             ;
                                    08  AC DD 000E7          PUSHL   MODE                                    ;
                                    1E  DD 000EA             PUSHL   #30                                     ;
                              6B        0B FB 000EC          CALLS   #11, SYS$ENQ                            ;
                              57        50 D0 000EF          MOVL    R0, STATUS                              ;
                              06        57 E9 000F2          BLBC    STATUS, 13$                             ; 1891
                              57        66 3C 000F5          MOVZWL  (CACHE_ENTRY), STATUS                   ; 1892
                              04        57 E8 000F8          BLBS    STATUS, 14$                             ; 1893
                                      FEFF 000FB 13$:        BUGW                                            ; 1894
                                     0000* 000FD             .WORD   <BUG$_XQPERR!4>                         ;
                              66        04 AC B0 000FF 14$:  MOVW    J, (CACHE_ENTRY)                        ; 1896
                        02    A6        59 B0 00103          MOVW    SAVE_LRU, 2(CACHE_ENTRY)                ; 1897
                                        04 00107             RET                                             ; 1899

; Routine Size:  264 bytes,    Routine Base:  $CODE$ + 0442
```

```
;  907     1900  1 GLOBAL ROUTINE REL_QUOTA_LOCK (J) : L_NORM NOVALUE =
;  908     1901  1
;  909     1902  1 !++
;  910     1903  1 !
;  911     1904  1 ! FUNCTIONAL DESCRIPTION:
;  912     1905  1 !
;  913     1906  1 !       This routine releases the lock associated with a quota cache
;  914     1907  1 !       entry. The value block held in the cache entry is written to
;  915     1908  1 !       the lock.
;  916     1909  1 !
;  917     1910  1 ! CALLING SEQUENCE:
;  918     1911  1 !       REL_QUOTA_LOCK (J)
;  919     1912  1 !
;  920     1913  1 ! INPUT PARAMETERS:
;  921     1914  1 !       J: index of quota cache entry
;  922     1915  1 !
;  923     1916  1 ! IMPLICIT INPUTS:
;  924     1917  1 !       NONE
;  925     1918  1 !
;  926     1919  1 ! OUTPUT PARAMETERS:
;  927     1920  1 !       NONE
;  928     1921  1 !
;  929     1922  1 ! IMPLICIT OUTPUTS:
;  930     1923  1 !       NONE
;  931     1924  1 !
;  932     1925  1 ! ROUTINE VALUE:
;  933     1926  1 !       NONE
;  934     1927  1 !
;  935     1928  1 ! SIDE EFFECTS:
;  936     1929  1 !       Lock released, value block written, cache entry marked non valid.
;  937     1930  1 !
;  938     1931  1 !--
;  939     1932  1
;  940     1933  2 BEGIN
;  941     1934  2
;  942     1935  2 LOCAL
;  943     1936  2         CACHE_ENTRY     : REF BBLOCK;   ! quota cache entry pointer
;  944     1937  2
;  945     1938  2 BIND_COMMON;
;  946     1939  2
;  947     1940  2
;  948     1941  2 ! Release the lock.
;  949     1942  2 !
;  950     1943  2
;  951     1944  2 CACHE_ENTRY = BBLOCKVECTOR [BBLOCK [.CURRENT_VCB[VCB$L_QUOCACHE],
;  952     1945  2                     VCA$L_QUOLIST], .J-1, VCA$R_QUOLOCK; ,VCA$C_QUOLENGTH];
;  953     1946  2
;  954     1947  2 IF .CACHE_ENTRY[VCA$L_QUOLKID] NEQ 0
;  955     1948  2 THEN
;  956     1949  3     BEGIN
;  957     1950  3
;  958   P 1951  3     IF NOT $DEQ (LKID   = .CACHE_ENTRY[VCA$L_QUOLKID],
;  959   P 1952  3                  VALBLK = CACHE_ENTRY[VCA$L_QUORECNUM]
;  960     1953  4                 )
;  961     1954  3     THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error');
;  962     1955  2     END;
;  963     1956  2
```

```
;  964          1957  2 ! Mark the cache entry no longer valid.
;  965          1958  2 !
;  966          1959  2
;  967          1960  2 CH$FILL (0, VCA$C_QUOLENGTH, .CACHE_ENTRY);
;  968          1961  2
;  969          1962  1 END;                                    ! End of routine REL_QUOTA_LOCK


                                                                .EXTRN  SYS$DEQ

                                                 003C 00000     .ENTRY  REL_QUOTA_LOCK, Save R2,R3,R4,R5        ; 1900
                                        50   98    AA   D0 00002  MOVL   -104(BASE), R0                        ; 1944
                                52      04   AC    1C   C5 00006  MULL3  #28, J, R2                           ; 1945
                                        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)                       ; 1947
                                                   16   13 00015  BEQL   1$                                   ;
                                                   7E   7C 00017  CLRQ   -(SP)                               ; 1953
                                             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                                      ; 1954
                                                       0000* 0002B  .WORD  <BUG$_XQPERR!4>                    ;
          1C         00          6E                   00   2C 0002D 1$:  MOVC5  #0, (SP), #0, #28, (CACHE_ENTRY)  ; 1960
                                                       62        00032                                       ;
                                                          04 00033  RET                                       ; 1962

; Routine Size:  52 bytes,    Routine Base:  $CODE$ + 054A
```

```
;  971          1963  1 GLOBAL ROUTINE CLEAN_QUO_CACHE (J, Q_RECORD) : L_NORM NOVALUE =
;  972          1964  1
;  973          1965  1 !++
;  974          1966  1 !
;  975          1967  1 ! FUNCTIONAL DESCRIPTION:
;  976          1968  1 !
;  977          1969  1 !       This routine updates the indicated quota record buffer from the
;  978          1970  1 !       indicated cache entry, and marks the record dirty and marks the
;  979          1971  1 !       cache entry clean if necessary.
;  980          1972  1 !
;  981          1973  1 !
;  982          1974  1 ! CALLING SEQUENCE:
;  983          1975  1 !       CLEAN_QUO_CACHE (ARG1, ARG2)
;  984          1976  1 !
;  985          1977  1 ! INPUT PARAMETERS:
;  986          1978  1 !       ARG1: index in quota cache
;  987          1979  1 !             0 to not
;  988          1980  1 !
;  989          1981  1 ! IMPLICIT INPUTS:
;  990          1982  1 !       CURRENT_VCB: VCB of volume
;  991          1983  1 !
;  992          1984  1 ! OUTPUT PARAMETERS:
;  993          1985  1 !       ARG2: address of record buffer
;  994          1986  1 !
;  995          1987  1 ! IMPLICIT OUTPUTS:
;  996          1988  1 !       NONE
;  997          1989  1 !
;  998          1990  1 ! ROUTINE VALUE:
;  999          1991  1 !       1
; 1000          1992  1 !
; 1001          1993  1 ! SIDE EFFECTS:
; 1002          1994  1 !       quota cache entry modified, buffer marked dirty
; 1003          1995  1 !
; 1004          1996  1 !--
; 1005          1997  1
; 1006          1998  2 BEGIN
; 1007          1999  2
; 1008          2000  2 MAP
; 1009          2001  2         Q_RECORD        : REF BBLOCK;   ! address of quota record
; 1010          2002  2
; 1011          2003  2 LOCAL
; 1012          2004  2         CACHE_ENTRY     : REF BBLOCK;   ! quota cache entry pointer
; 1013          2005  2
; 1014          2006  2 BIND_COMMON;
; 1015          2007  2
; 1016          2008  2 EXTERNAL ROUTINE
; 1017          2009  2         MARK_DIRTY      : L_NORM;       ! mark buffer for write back
; 1018          2010  2
; 1019          2011  2
; 1020          2012  2 ! Copy the cache entry to the record buffer. If the cache entry is marked
; 1021          2013  2 ! dirty, mark it clean and mark the record dirty.
; 1022          2014  2 !
; 1023          2015  2
; 1024          2016  2 CACHE_ENTRY = BBLOCKVECTOR [BBLOCK [.CURRENT_VCB[VCB$L_QUOCACHE],
; 1025          2017  2                 VCA$L_QUOLIST], .J-1, VCA$R_QUOLOCK; ,VCA$C_QUOLENGTH];
; 1026          2018  2
; 1027          2019  2 Q_RECORD[DQF$L_UIC] = .CACHE_ENTRY[VCA$L_QUOUIC];
```

```
;  1028          2020  2 CH$MOVE (12, CACHE_ENTRY[VCA$L_USAGE], Q_RECORD[DQF$L_USAGE]);
;  1029          2021  2 IF .CACHE_ENTRY[VCA$V_QUODIRTY]
;  1030          2022  2 THEN
;  1031          2023  3     BEGIN
;  1032          2024  3     CACHE_ENTRY[VCA$V_QUODIRTY] = 0;
;  1033          2025  3     MARK_DIRTY (.Q_RECORD);
;  1034          2026  2     END;
;  1035          2027  2
;  1036          2028  1 END;                                    ! end of routine CLEAN_QUO_CACHE


                                                 007C 00000          .ENTRY  CLEAN_QUO_CACHE, Save R2,R3,R4,R5,R6    ; 1963
                                  50         98  AA   D0 00002          MOVL    -104(BASE), R0                          ; 2016
                          56      04         AC   1C   C5 00006          MULL3   #28, J, R6                              ; 2017
                                  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                            ; 2019
                                  04         A0   18   A6 D0 00016       MOVL    24(CACHE_ENTRY), 4(R0)                  ;
                                  50         08   AC   D0 0001B          MOVL    Q_RECORD, R0                            ; 2020
                      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$                 ; 2021
                                  0B         A6        02 8A 0002A       BICB2   #2, 11(CACHE_ENTRY)                     ; 2024
                                                   08  AC   DD 0002E      PUSHL   Q_RECORD                            ; 2025
                                0000G        CF        01   FB 00031      CALLS   #1, MARK_DIRTY                          ;
                                                       04 00036 1$:      RET                                             ; 2028

; Routine Size:  55 bytes,    Routine Base:  $CODE$ + 057E
```

```
;  1038     2029  1 ROUTINE ENTER_QUO_CACHE (J, Q_RECORD, MARK_DIRTY, MARK_USE) : L_NORM NOVALUE =
;  1039     2030  1
;  1040     2031  1 !++
;  1041     2032  1 !
;  1042     2033  1 ! FUNCTIONAL DESCRIPTION:
;  1043     2034  1 !
;  1044     2035  1 !       This routine enters the given quota record into the cache at the
;  1045     2036  1 !       indicated cache index. If requested, the cache entry is marked dirty.
;  1046     2037  1 !
;  1047     2038  1 !
;  1048     2039  1 ! CALLING SEQUENCE:
;  1049     2040  1 !       ENTER_QUO_CACHE (ARG1, ARG2, ARG3, ARG4)
;  1050     2041  1 !
;  1051     2042  1 ! INPUT PARAMETERS:
;  1052     2043  1 !       ARG1: index in quota cache
;  1053     2044  1 !       ARG2: address of record buffer
;  1054     2045  1 !       ARG3: 1 to mark record dirty
;  1055     2046  1 !             0 to not
;  1056     2047  1 !       ARG4: 0 to set lowest possible LRU
;  1057     2048  1 !             1 to set current LRU
;  1058     2049  1 !             2 to leave LRU alone
;  1059     2050  1 !
;  1060     2051  1 ! IMPLICIT INPUTS:
;  1061     2052  1 !       CURRENT_VCB: VCB of volume
;  1062     2053  1 !       QUOTA_RECORD: record number of quota record
;  1063     2054  1 !
;  1064     2055  1 ! OUTPUT PARAMETERS:
;  1065     2056  1 !       NONE
;  1066     2057  1 !
;  1067     2058  1 ! IMPLICIT OUTPUTS:
;  1068     2059  1 !       NONE
;  1069     2060  1 !
;  1070     2061  1 ! ROUTINE VALUE:
;  1071     2062  1 !       1
;  1072     2063  1 !
;  1073     2064  1 ! SIDE EFFECTS:
;  1074     2065  1 !       quota cache entry modified
;  1075     2066  1 !
;  1076     2067  1 !--
;  1077     2068  1
;  1078     2069  2 BEGIN
;  1079     2070  2
;  1080     2071  2 MAP
;  1081     2072  2         Q_RECORD        : REF BBLOCK;   ! address of quota record
;  1082     2073  2
;  1083     2074  2 LOCAL
;  1084     2075  2         QUOTA_CACHE     : REF BBLOCK,   ! address of quota cache
;  1085     2076  2         CACHE_ENTRY     : REF BBLOCK;   ! quota cache entry pointer
;  1086     2077  2
;  1087     2078  2 BIND_COMMON;
;  1088     2079  2
;  1089     2080  2 ! Copy the record data to the cache entry. If requested, mark the cache
;  1090     2081  2 ! entry dirty.
;  1091     2082  2 !
;  1092     2083  2
;  1093     2084  2 QUOTA_CACHE = .CURRENT_VCB[VCB$L_QUOCACHE];
;  1094     2085  2 CACHE_ENTRY = BBLOCKVECTOR [BBLOCK [.QUOTA_CACHE, VCA$L_QUOLIST],
```

```
; 1095        2086  2                  .J-1, VCA$R_QUOLOCK; ,VCA$C_QUOLENGTH];
; 1096        2087  2
; 1097        2088  2
; 1098        2089  2  CACHE_ENTRY[VCA$L_QUOUIC] = .Q_RECORD[DQF$L_UIC];
; 1099        2090  2  CH$MOVE (12, Q_RECORD[DQF$L_USAGE], CACHE_ENTRY[VCA$L_USAGE]);
; 1100        2091  2  CACHE_ENTRY[VCA$B_QUOFLAGS] = VCA$M_QUOVALID;
; 1101        2092  2  CACHE_ENTRY[VCA$W_QUOINDEX] = .J;
; 1102        2093  3  CACHE_ENTRY[VCA$L_QUORECNUM] = (IF .Q_RECORD[DQF$V_ACTIVE]
; 1103        2094  3                                 THEN .QUOTA_RECORD
; 1104        2095  2                                 ELSE 0);
; 1105        2096  2  IF .MARK_USE
; 1106        2097  2  THEN
; 1107        2098  3      BEGIN
; 1108        2099  3      CACHE_ENTRY[VCA$W_QUOLRUX] = .QUOTA_CACHE[VCA$W_QUOLRU];
; 1109        2100  3      QUOTA_CACHE[VCA$W_QUOLRU] = .QUOTA_CACHE[VCA$W_QUOLRU] + 1;
; 1110        2101  3      END
; 1111        2102  2  ELSE IF .MARK_USE EQL 0
; 1112        2103  2  THEN
; 1113        2104  3      BEGIN
; 1114        2105  3      CACHE_ENTRY[VCA$W_QUOLRUX] = .QUOTA_CACHE[VCA$W_QUOLRU] - 1^15;
; 1115        2106  2      END;
; 1116        2107  2
; 1117        2108  2  IF .MARK_DIRTY
; 1118        2109  2  THEN CACHE_ENTRY[VCA$V_QUODIRTY] = 1;
; 1119        2110  2
; 1120        2111  1  END;                                   ! end of routine ENTER_QUO_CACHE


                                        00FC 00000 ENTER_QUO_CACHE:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7                     ; 2029
                        50      98   AA  D0 00002        MOVL    -104(BASE), R0                              ; 2084
                        57      5C   A0  D0 00006        MOVL    92(R0), QUOTA_CACHE                         ;
              50   04   AC           1C  C5 0000A        MULL3   #28, J, R0                                  ; 2086
                        56   28 A047     9E 0000F        MOVAB   40(R0)[QUOTA_CACHE], CACHE_ENTRY            ;
                        50      08   AC  D0 00014        MOVL    Q_RECORD, R0                                ; 2089
                   18   A6      04   A0  D0 00018        MOVL    4(R0), 24(CACHE_ENTRY)                      ;
                        50      08   AC  D0 0001D        MOVL    Q_RECORD, R0                                ; 2090
         0C   A6   08   A0           0C  28 00021        MOVC3   #12, 8(R0), 12(CACHE_ENTRY)                 ;
                   0B   A6           01  90 00027        MOVB    #1, 11(CACHE_ENTRY)                         ; 2091
                        66      04   AC  B0 0002B        MOVW    J, (CACHE_ENTRY)                            ; 2092
                        07      08   BC  E9 0002F        BLBC    @Q_RECORD, 1$                               ; 2093
                        50    02B4   CA  D0 00033        MOVL    692(BASE), R0                               ; 2094
                                     02  11 00038        BRB     2$                                          ;
                                     50  D4 0003A 1$:    CLRL    R0                                          ; 2093
08   A6        18       00           50  F0 0003C 2$:    INSV    R0, #0, #24, 8(CACHE_ENTRY)                 ;
                        0A      10   AC  E9 00042        BLBC    MARK_USE, 3$                                ; 2096
                   02   A6      02   A7  B0 00046        MOVW    2(QUOTA_CACHE), 2(CACHE_ENTRY)              ; 2099
                                02   A7  B6 0004B        INCW    2(QUOTA_CACHE)                              ; 2100
                                     0D  11 0004E        BRB     4$                                          ; 2096
                                10   AC  D5 00050 3$:    TSTL    MARK_USE                                    ; 2102
                                     08  12 00053        BNEQ    4$                                          ;
         02   A6   02   A7    8000   8F  A1 00055        ADDW3   #-32768, 2(QUOTA_CACHE), 2(CACHE_ENTRY)     ; 2105
                        04      0C   AC  E9 0005D 4$:    BLBC    MARK_DIRTY, 5$                              ; 2108
                   0B   A6           02  88 00061        BISB2   #2, 11(CACHE_ENTRY)                         ; 2109
```

```
                                    04 00065 5$:     RET                                                    ; 2111

; Routine Size:  102 bytes,    Routine Base:  $CODE$ + 05B5

; 1121          2112  1
; 1122          2113  1 END
; 1123          2114  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $CODE$                         1563  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                                 Library Statistics
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;  _$255$DUA18:[SYSLIB]LIB.L32;1           18619        82         0        1000        00:01.9



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

; Size:          1557 code + 6 data bytes
; Run Time:        01:32.4
; Elapsed Time:    02:08.7
; Lines/CPU Min:    1373
; Lexemes/CPU-Min: 60255
; Memory Used:  311 pages
; Compilation Complete
```

0442 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

F11BXQP
MAP

ACCESS
LIS

TA78DVDEP
LIS

CLENUP
LIS

CHARGEQ
LIS

F11X