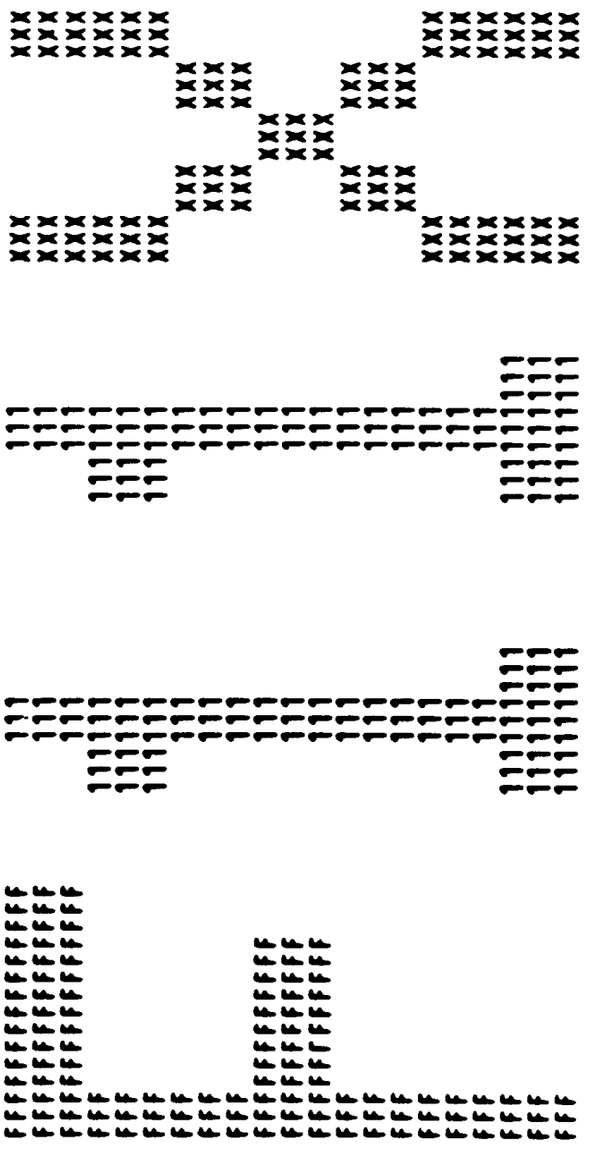

FILEID**CLENUP

```
  CCCCCCCC  LL          EEEEEEEEEE  NN      NN  UU      UU  PPPPPPPP
  CCCCCCCC  LL          EEEEEEEEEE  NN      NN  UU      UU  PPPPPPPP
CC          LL          EE          NN      NN  UU      UU  PP      PP
CC          LL          EE          NN      NN  UU      UU  PP      PP
CC          LL          EE          NNNN    NN  UU      UU  PP      PP
CC          LL          EE          NNNN    NN  UU      UU  PP      PP
CC          LL          EEEEEEEE    NN  NN  NN  UU      UU  PPPPPPPP
CC          LL          EEEEEEEE    NN  NN  NN  UU      UU  PPPPPPPP
CC          LL          EE          NN    NNNN  UU      UU  PP
CC          LL          EE          NN    NNNN  UU      UU  PP
CC          LL          EE          NN      NN  UU      UU  PP
CC          LL          EE          NN      NN  UU      UU  PP
  CCCCCCCC  LLLLLLLLLL  EEEEEEEEEE  NN      NN  UUUUUUUUUU  PP
  CCCCCCCC  LLLLLLLLLL  EEEEEEEEEE  NN      NN  UUUUUUUUUU  PP


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 CLENUP (
;     2          0002  0                 LANGUAGE (BLISS32),
;001  !CDS0023   0003  0                 IDENT = 'V04-002'
;     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 performs the necessary cleanup after an operation
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !
;    41          0041  1 !       STARLET operating system, including privileged system services
;    42          0042  1 !       and internal exec routines.
;    43          0043  1 !
;    44          0044  1 !--
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  6-Jan-1977  23:53
;    48          0048  1 !
;    49          0049  1 ! MODIFIED BY:
;    50          0050  1 !
;001  !CDS0023   0051  1 !       V04-002 CDS0023         Christian D. Saether    15-Nov-1984
;002  !CDS0023   0052  1 !               Clear CLF_FIXFCB in err_cleanup when the file is deleted.
;003  !CDS0023   0053  1 !
;004  !ACG0468   0054  1 !       V04-001 ACG0468         Andrew C. Goldstein,    18-Sep-1984  18:33
;005  !ACG0468   0055  1 !               Allow for quota file write access in last writer check
;006  !ACG0468   0056  1 !
;    51          0057  1 !       V03-034 CDS0022         Christian D. Saether    30-Aug-1984
```

```
;    52          0058  1 ;          Allow for multi-header directory files.
;    53          0059  1 ;          Have error cleanup remove possible bias on primary_fcb
;    54          0060  1 ;          refcnt.
;    55          0061  1 ;
;    56          0062  1 ;      V03-033 CDS0021         Christian D. Saether    23-Aug-1984
;    57          0063  1 ;          Move code that marks FCB stale to a routine in LOCKERS.
;    58          0064  1 ;
;    59          0065  1 ;      V03-032 CDS0020         Christian D. Saether    13-Aug-1984
;    60          0066  1 ;          Add code to mark primary fcb stale clusterwide.
;    61          0067  1 ;
;    62          0068  1 ;      V03-031 CDS0019         Christian D. Saether     7-Aug-1984
;    63          0069  1 ;          Cleanup potential directory index cache block
;    64          0070  1 ;          when deleting a file.
;    65          0071  1 ;
;    66          0072  1 ;      V03-030 CDS0018         Christian D. Saether     1-Aug-1984
;    67          0073  1 ;          Modify test for directory fcb.
;    68          0074  1 ;          Add SET_DIRINDX routine.
;    69          0075  1 ;          Add NUKE_PRIM_FCB routine.
;    70          0076  1 ;          Modify ZERO_IDX routine.
;    71          0077  1 ;
;    72          0078  1 ;      V03-029 ACG0438         Andrew C. Goldstein,    19-Jul-1984  17:55
;    73          0079  1 ;          Add cluster-wide special cache interlock logic.
;    74          0080  1 ;          Condition DELETEACL calls on non-empty ACL.
;    75          0081  1 ;          Use central dequeue routine.
;    76          0082  1 ;
;    77          0083  1 ;      V03-028 CDS0017         Christian D. Saether    25-May-1984
;    78          0084  1 ;          Call KILL_BUFFERS routine to flush cache in
;    79          0085  1 ;          certain situations when not in a cluster.
;    80          0086  1 ;
;    81          0087  1 ;      V03-027 CDS0016         Christian D. Saether     9-May-1984
;    82          0088  1 ;          Release allocation lock prior to calling send_symbiont.
;    83          0089  1 ;
;    84          0090  1 ;      V03-026 CDS0015         Christian D. Saether     4-May-1984
;    85          0091  1 ;          No not map notrunc into nowrite.
;    86          0092  1 ;          Add bugcheck if access lock conversion fails in make_deaccess.
;    87          0093  1 ;
;    88          0094  1 ;      V03-025 CDS0014         Christian D. Saether     3-May-1984
;    89          0095  1 ;          Call CONV_ACCLOCK to remove possible access lock
;    90          0096  1 ;          when deallocating fcb's.
                0097  1 ;
                0098  1 ;      V03-024 CDS0013         Christian D. Saether    19-Apr-1984
;    93          0099  1 ;          Changes to FCB$W_ACNT handling.
;    94          0100  1 ;
;    95          0101  1 ;      V03-023 ACG0415         Andrew C. Goldstein,     5-Apr-1984  21:27
;    96          0102  1 ;          Interface change to ACL_DELETEACL
;    97          0103  1 ;
;    98          0104  1 ;      V03-022 ACG0408         Andrew C. Goldstein,    23-Mar-1984  11:20
;    99          0105  1 ;          Make rest of global storage based
;   100          0106  1 ;
;   101          0107  1 ;      V03-021 CDS0012         Christian D. Saether     9-Mar-1984
;   102          0108  1 ;          Put in bug trap to catch possible double remque of
;   103          0109  1 ;          FCB.
;   104          0110  1 ;
;   105          0111  1 ;      V03-020 CDS0011         Christian D. Saether    23-Feb-1984
;   106          0112  1 ;          Use new WRITE_DIRTY routine to replace FLUSH_BUFFERS.
;   107          0113  1 ;          Remove references to FLUSH_FID.
;   108          0114  1 ;          Replace FLUSH_FID (0) with KILL_CACHE calls.
```

```
;  109     0115  1 !
;  110     0116  1 !       V03-019 CDS0010         Christian D. Saether    27-Dec-1983
;  111     0117  1 !               Use L_NORM linkage.
;  112     0118  1 !               Use BIND_COMMON macro to reduce external declarations.
;  113     0119  1 !
;  114     0120  1 !       V03-018 CDS0009         Christian D. Saether    23-Nov-1983
;  115     0121  1 !               If DIR_FCB is the same as PRIMARY_FCB, do not return
;  116     0122  1 !               the FCB until the end of cleanup (as PRIMARY_FCB, not
;  117     0123  1 !               DIR_FCB).
;  118     0124  1 !               Move cleanup of DIR_FCB until after all i/o is done.
;  119     0125  1 !               Remove REMOVE_FCB routine (kernel call not necessary).
;  120     0126  1 !
;  121     0127  1 !       V03-017 LMP0164         L. Mark Pilant,         10-Oct-1983  15:22
;  122     0128  1 !               Delete the in-core ACL if doing an FCB fixup.
;  123     0129  1 !
;  124     0130  1 !       V03-016 CDS0008         Christian D. Saether     3-Oct-1983
;  125     0131  1 !               Handle CURR_LCKINDX in err_cleanup.  Don't read
;  126     0132  1 !               headers without appropriate serial locks.
;  127     0133  1 !
;  128     0134  1 !       V03-015 CDS0007         Christian D. Saether    14-Sep-1983
;  129     0135  1 !               Take out deqall hack now that RMS does it's own
;  130     0136  1 !               root locks again.
;  131     0137  1 !
;  132     0138  1 !       V03-014 CDS0006         Christian D. Saether    27-Jul-1983
;  133     0139  1 !               Change interface to SEND_SYMBIONT.
;  134     0140  1 !
;  135     0141  1 !       V03-013 LJK0199         Lawrence J. Kenah       27-Apr-1983
;  136     0142  1 !               Do not credit FILCNT when giving back shared window
;  137     0143  1 !
;  138     0144  1 !       V03-012 CDS0006         Christian D. Saether    28-Apr-1983
;  139     0145  1 !               Clear DIR_ENTRY when DIR_FCB is cleared.
;  140     0146  1 !
;  141     0147  1 !       V03-011 CDS0005         Christian D. Saether    21-Apr-1983
;  142     0148  1 !               Change interface to TRUNCATE routine
;  143     0149  1 !
;  144     0150  1 !       V03-010 CDS0004         Christian D. Saether    19-Apr-1983
;  145     0151  1 !               Bug check on unexpected lock manager errors.
;  146     0152  1 !               Clear ACCLKID field in window.
;  147     0153  1 !
;  148     0154  1 !       V03-009 ACG0323         Andrew C. Goldstein,    12-Apr-1983  14:09
;  149     0155  1 !               Add extended file name to back link fixup
;  150     0156  1 !
;  151     0157  1 !       V03-008 STJ3069         Steven T. Jeffreys,     23-Mar-1983
;  152     0158  1 !               Use the ERASE_REQUESTED parameter of RETURN_BLOCKS.
;  153     0159  1 !
;  154     0160  1 !       V03-007 CDS0003         Christian D. Saether     7-Mar-1983
;  155     0161  1 !               Perform a DEQALL if file access lock dequeue fails
;  156     0162  1 !               due to sublocks, then redo the file access dequeue.
;  157     0163  1 !
;  158     0164  1 !       V03-006 LMP0071         L. Mark Pilant,         19-Jan-1983  20:49
;  159     0165  1 !               Correct a problem that caused ACL segments to be left laying
;  160     0166  1 !               around when a directory FCB was flushed.
;  161     0167  1 !
;  162     0168  1 !       V03-005 ACG0308         Andrew C. Goldstein,    14-Jan-1983  15:02
;  163     0169  1 !               Fix FCB linkage consistency problems
;  164     0170  1 !
;  165     0171  1 !       V03-004 CDS0002         Christian D. Saether     3-Jan-1983
```

```
;  166     0172  1 !
;  167     0173  1 !       Always flush header cache until it is restored for xqp.
;  168     0174  1 !
;  169     0175  1 !   V03-003 LMP0059         L. Mark Pilant,         21-Dec-1982  12:23
;  170     0176  1 !       Always create an FCB when accessing a file header.  This
;  171     0177  1 !       eliminates a lot of special case FCB handling.
;  172     0178  1 !
;  173     0179  1 !   V03-002 CDS0001         Christian D. Saether    10-Dec-1982
;  174     0180  1 !       MAKE_DEACCESS dequeues access lock.
;  175     0181  1 !
;  176     0182  1 !   V03-001 LMP0036         L. Mark Pilant,         17-Aug-1982  10:45
;  177     0183  1 !       If the ACL was built using a dummy FCB, dismantle and
;  178     0184  1 !       deallocate the ACL.
;  179     0185  1 !
;  180     0186  1 !   V02-024 ACG0259         Andrew C. Goldstein,    26-Jan-1982  19:12
;  181     0187  1 !       Add mode arg to REMOVE
;  182     0188  1 !
;  183     0189  1 !   V02-023 ACG0247         Andrew C. Goldstein,    23-Dec-1981  20:26
;  184     0190  1 !       Make /NOCACHE flush all caches
;  185     0191  1 !
;  186     0192  1 !   V02-022 ACG0245         Andrew C. Goldstein,    23-Dec-1981  20:26
;  187     0193  1 !       Send spool file to print during cleanup
;  188     0194  1 !
;  189     0195  1 !   V02-021 ACG0244         Andrew C. Goldstein,    23-Dec-1981  20:14
;  190     0196  1 !       Do buffer flush before deallocating control blocks
;  191     0197  1 !
;  192     0198  1 !   V02-020 LMP0003         L. Mark Pilant, 30-Nov-1981  16:40
;  193     0199  1 !       Properly cleanup any cathedral windows.
;  194     0200  1 !
;  195     0201  1 !   V02-019 ACG0208         Andrew C. Goldstein,    11-Nov-1981  17:51
;  196     0202  1 !       Add segmented directory record support
;  197     0203  1 !
;  198     0204  1 !   V02-018 ACG0168         Andrew C. Goldstein,    7-May-1980  18:22
;  199     0205  1 !       Fix last block directory cleanup on delete failure
;  200     0206  1 !
;  201     0207  1 !   V02-017 ACG0167         Andrew C. Goldstein,    16-Apr-1980  19:25
;  202     0208  1 !       Previous revision history moved to F11B.REV
;  203     0209  1 !**
;  204     0210  1
;  205     0211  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;  206     0212  1 REQUIRE 'SRC$:FCPDEF.B32';
;  207     1203  1
;  208     1204  1
;  209     1205  1 FORWARD ROUTINE
;  210     1206  1         CLEANUP             : L_NORM,           ! normal cleanup
;  211     1207  1         ZERO_WINDOWS        : L_NORM,           ! invalidate all windows of file
;  212     1208  1         ZERO_IDX            : L_NORM NOVALUE,   ! initialize directory index
;  213     1209  1         ERR_CLEANUP         : L_NORM,           ! cleanup after error
;  214     1210  1         FLUSH_FIDCACHE      : L_NORM,           ! clean out the file ID cache
;  215     1211  1         MAKE_DEACCESS       : L_NORM,           ! deaccess the file
;  216     1212  1         DEL_EXTFCB          : L_NORM,           ! deallocate extension FCB's
;  217     1213  1         ZERO_CHANNEL        : L_NORM,           ! zero user channel pointer
;  218     1214  1         SET_DIRINDX         : L_JSB_IARG,       ! test for directory index
;  219     1215  1         NUKE_HEAD_FCB       : L_NORM NOVALUE;   ! deallocate primary fcb
```

```
;  221          1216  1 GLOBAL ROUTINE CLEANUP : L_NORM =
;  222          1217  1
;  223          1218  1 !++
;  224          1219  1 !
;  225          1220  1 ! FUNCTIONAL DESCRIPTION:
;  226          1221  1 !
;  227          1222  1 !       This routine performs the cleanup needed after a successfully
;  228          1223  1 !       completed file operation.
;  229          1224  1 !
;  230          1225  1 ! CALLING SEQUENCE:
;  231          1226  1 !       CLEANUP ()
;  232          1227  1 !
;  233          1228  1 ! INPUT PARAMETERS:
;  234          1229  1 !       NONE
;  235          1230  1 !
;  236          1231  1 ! IMPLICIT INPUTS:
;  237          1232  1 !       CLEANUP_FLAGS: indicate specific actions to do
;  238          1233  1 !       PRIMARY_FCB: FCB of file
;  239          1234  1 !       CURRENT_WINDOW: window of file
;  240          1235  1 !       DIR_FCB: FCB of directory
;  241          1236  1 !       CURRENT_VCB: VCB of volume in process
;  242          1237  1 !       IO_PACKET: I/O packet of request
;  243          1238  1 !
;  244          1239  1 ! OUTPUT PARAMETERS:
;  245          1240  1 !       NONE
;  246          1241  1 !
;  247          1242  1 ! IMPLICIT OUTPUTS:
;  248          1243  1 !       NONE
;  249          1244  1 !
;  250          1245  1 ! ROUTINE VALUE:
;  251          1246  1 !       NONE
;  252          1247  1 !
;  253          1248  1 ! SIDE EFFECTS:
;  254          1249  1 !       FCB's and windows deleted when appropriate
;  255          1250  1 !       header written
;  256          1251  1 !       FCB updated
;  257          1252  1 !
;  258          1253  1 !--
;  259          1254  1
;  260          1255  2 BEGIN
;  261          1256  2
;  262          1257  2 LOCAL
;  263          1258  2         CLUSTER,                        ! are we a cluster
;  264          1259  2         QUOTA_CACHE     : REF BBLOCK,   ! address of quota cache
;  265          1260  2         FCB             : REF BBLOCK,   ! local FCB pointer
;  266          1261  2         VCB             : REF BBLOCK,   ! local VCB pointer
;  267          1262  2         RVT             : REF BBLOCK,   ! local RVT pointer
;  268          1263  2         UCB             : REF BBLOCK,   ! local UCB pointer
;  269          1264  2         HEADER          : REF BBLOCK;   ! file header
;  270          1265  2
;  271          1266  2 BIND_COMMON;
;  272          1267  2
;  273          1268  2 DIR_CONTEXT_DEF;
;  274          1269  2
;  275          1270  2 EXTERNAL
;  276          1271  2         CLU$GL_CLUB     : ADDRESSING_MODE (ABSOLUTE);
;  277          1272  2
```

```
;  278          1273  2 EXTERNAL ROUTINE
;  279          1274  2         MAKE_FCB_STALE  : L_NORM NOVALUE, ! mark fcb as stale clusterwide
;  280          1275  2         KILL_BUFFERS    : L_NORM NOVALUE, ! invalidate specified buffers
;  281          1276  2         KILL_CACHE      : L_NORM NOVALUE, ! invalidate all buffers for ucb
;  282          1277  2         WRITE_DIRTY     : L_NORM,         ! write all dirty buffers
;  283          1278  2         SWITCH_VOLUME   : L_NORM,         ! switch to desired volume
;  284          1279  2         FLUSH_QUO_CACHE : L_NORM;         ! flush the quota cache
;  285          1280  2
;  286          1281  2
;  287          1282  2 ! ***** Note: The primary header of the current file is not necessarily
;  288          1283  2 ! resident at this point.
;  289          1284  2 !
;  290          1285  2 ! Switch back to the primary context area if necessary (no normal cleanup
;  291          1286  2 ! is ever necessary on secondary context).
;  292          1287  2 !
;  293          1288  2
;  294          1289  2 IF .CONTEXT_SAVE NEQ 0
;  295          1290  2 THEN
;  296          1291  3     BEGIN
;  297          1292  3     CH$MOVE (CONTEXT_SIZE, CONTEXT_SAVE, CONTEXT_START);
;  298          1293  3     CONTEXT_SAVE = 0;
;  299          1294  2     END;
;  300          1295  2
;  301          1296  2 CLUSTER = 0;
;  302          1297  2 IF .BBLOCK [CURRENT_UCB [UCB$L_DEVCHAR2], DEV$V_CLU]
;  303          1298  2     AND .CLU$GL_CLUB NEQ 0
;  304          1299  2 THEN
;  305          1300  2     CLUSTER = 1;
;  306          1301  2
;  307          1302  2 ! Check the entire volume set to see if the index file or storage map
;  308          1303  2 ! on any volume is write accessed. If so, flush the buffer pool of any
;  309          1304  2 ! of their blocks, and flush the file ID and extent caches as appropriate.
;  310          1305  2 ! Also, if the volume is mounted /NOCACHE, flush the entire buffer cache.
;  311          1306  2 !
;  312          1307  2
;  313          1308  2 RVT = .CURRENT_VCB[VCB$L_RVT];
;  314          1309  2 INCR J FROM 1 TO
;  315          1310  3     BEGIN
;  316          1311  3     IF .RVT EQL .CURRENT_UCB
;  317          1312  4     THEN (UCB = .RVT; 1)
;  318          1313  3     ELSE .RVT[RVT$B_NVOLS]
;  319          1314  3     END
;  320          1315  2 DO
;  321          1316  3     BEGIN
;  322          1317  3     IF .RVT NEQ .CURRENT_UCB
;  323          1318  3     THEN UCB = .VECTOR [RVT[RVT$L_UCBLST], .J-1];
;  324          1319  3     IF .UCB NEQ 0
;  325          1320  3
;  326          1321  3     THEN
;  327          1322  4         BEGIN
;  328          1323  4         VCB = .UCB[UCB$L_VCB];
;  329          1324  4
;  330          1325  4         IF .J EQL 1
;  331          1326  4         THEN
;  332          1327  5             BEGIN
;  333          1328  5
;  334          1329  5 ! If someone has the quota file write accessed (and it is active), flush it
```

```
;  335          1330  5 ! from the buffer pool. (Note that the quota file is located on RVN 1.)
;  336          1331  5 !
;  337          1332  5
;  338          1333  5             QUOTA_CACHE = .VCB[VCB$L_QUOCACHE];
;  339          1334  5             IF .QUOTA_CACHE NEQ 0
;  340          1335  5             THEN
;  341          1336  5                 IF TESTBITSC (QUOTA_CACHE[VCA$V_CACHEFLUSH])
;  342          1337  5                 THEN
;  343          1338  6                     BEGIN
;  344          1339  6                     SWITCH_VOLUME (1);
;  345          1340  6                     FLUSH_QUO_CACHE (); ! may create modified buffers
;  346          1341  5                     END;
;  347          1342  4             END;                    ! of this is RVN 1 (or single volume)
;  348          1343  4
;  349          1344  4 ! If the volume is marked for dismount or nocache, flush out all the
;  350          1345  4 ! caches.
;  351          1346  4 !
;  352          1347  4
;  353          1348  4         IF .BBLOCK [UCB [UCB$L_DEVCHAR], DEV$V_DMT]
;  354          1349  4         OR .VCB[VCB$V_NOCACHE]
;  355          1350  4         THEN
;  356          1351  5             BEGIN
;  357          1352  5             SWITCH_VOLUME (.J);
;  358          1353  5             WRITE_DIRTY (0);
;  359          1354  5             KILL_CACHE (.UCB); ! we cannot use the block cache after this
;  360          1355  4             END;
;  361          1356  3         END;
;  362          1357  2     END;
;  363          1358  2
;  364          1359  2 ! Write modified buffers.  The various cache purges above may have
;  365          1360  2 ! created more dirty buffers than we had at the start of this routine.
;  366          1361  2 ! No more dirty buffers can be created for the remainder of this request.
;  367          1362  2 !
;  368          1363  2
;  369          1364  2 WRITE_DIRTY (0);
;  370          1365  2
;  371          1366  2 ! Invalidate any windows on the file, if requested.
;  372          1367  2 !
;  373          1368  2
;  374          1369  2 IF TESTBITSC (CLEANUP_FLAGS[CLF_INVWINDOW])
;  375          1370  2 THEN KERNEL_CALL (ZERO_WINDOWS, .PRIMARY_FCB);
;  376          1371  2
;  377          1372  2 ! If a directory fcb is left lying about with no use, dispose of it.
;  378          1373  2 ! If the directory file is write accessed, flush the buffer pool of any
;  379          1374  2 ! blocks that might be resident. Also flush the directory index.
;  380          1375  2 ! Cleanup of these fcbs is deferred until all possible errors in the
;  381          1376  2 ! cleanup procedure (i/o errors) have already had an opportunity to happen.
;  382          1377  2 !
;  383          1378  2
;  384          1379  2 IF (FCB = .DIR_FCB) NEQ 0
;  385          1380  2 THEN
;  386          1381  3     BEGIN
;  387          1382  3     IF .FCB [FCB$W_REFCNT] EQL 0
;  388          1383  3     THEN
;  389          1384  4         BEGIN
;  390          1385  4         IF .FCB NEQ .PRIMARY_FCB
;  391          1386  4         THEN
```

```
;  392          1387  4              IF NOT SET_DIRINDX (.FCB)
;  393          1388  4              THEN
;  394          1389  5                  BEGIN
;  395          1390  5                  DEL_EXTFCB (.FCB);
;  396          1391  5                  NUKE_HEAD_FCB (.FCB);
;  397          1392  4                  END;
;  398          1393  4
;  399          1394  4          END
;  400          1395  4
;  401          1396  3      ELSE
;  402          1397  4          BEGIN
;  403          1398  4          IF .FCB [FCB$W_WCNT] NEQ 0
;  404          1399  4          THEN
;  405          1400  5              BEGIN
;  406          1401  5              SWITCH_VOLUME (.FCB [FCB$W_FID_RVN]);
;  407          1402  5              IF NOT .CLUSTER
;  408          1403  5              THEN
;  409          1404  5                  KILL_BUFFERS (1, .FCB [FCB$L_LOCKBASIS]);
;  410          1405  5              ZERO_IDX ();
;  411          1406  4              END;
;  412          1407  3          END;
;  413          1408  3
;  414          1409  3  ! Guarantee that no further attempts will be made to do any directory
;  415          1410  3  ! related cleanup.  This cleanup code was moved beyond the buffer
;  416          1411  3  ! cleanup to avoid the same situation, but clearing the cleanup flags
;  417          1412  3  ! makes sure.
;  418          1413  3  !
;  419          1414  3
;  420          1415  3      CLEANUP_FLAGS [CLF_SUPERSEDE] = 0;
;  421          1416  3      CLEANUP_FLAGS [CLF_REENTER] = 0;
;  422          1417  3      CLEANUP_FLAGS [CLF_REMOVE] = 0;
;  423          1418  3      DIR_FCB = 0;
;  424          1419  3      DIR_ENTRY = 0;
;  425          1420  3
;  426          1421  2      END;
;  427          1422  2
;  428          1423  2  IF (FCB = .PRIMARY_FCB) NEQ 0
;  429          1424  2  THEN
;  430          1425  3      BEGIN
;  431          1426  3
;  432          1427  3  ! Check if the fcb has been modified and if so, and this is a cluster,
;  433          1428  3  ! cause potential fcbs on other nodes to be marked as stale so they
;  434          1429  3  ! will know to rebuild their fcb chains from the file header(s).
;  435          1430  3  !
;  436          1431  3
;  437          1432  3      IF .CLEANUP_FLAGS [CLF_MARKFCBSTALE]
;  438          1433  3          AND .CLUSTER
;  439          1434  3      THEN
;  440          1435  3          MAKE_FCB_STALE (.FCB);
;  441          1436  3
;  442          1437  3  ! If an FCB is left about with no use, dispose of it.
;  443          1438  3  ! Check whether it is a directory fcb first.
;  444          1439  3  !
;  445          1440  3
;  446          1441  3      IF .FCB[FCB$W_REFCNT] EQL 0
;  447          1442  3      THEN
;  448          1443  3          IF NOT SET_DIRINDX (.FCB)
```

```
;  449          1444  3          THEN
;  450          1445  4              BEGIN
;  451          1446  4
;  452          1447  4              DEL_EXTFCB (.FCB);
;  453          1448  4
;  454          1449  4              NUKE_HEAD_FCB (.FCB);
;  455          1450  4
;  456          1451  4              PRIMARY_FCB = 0;
;  457          1452  3              END;
;  458          1453  2      END;
;  459          1454  2
;  460          1455  2 RETURN 1;
;  461          1456  2
;  462          1457  1 END;                                    ! end of routine CLEANUP

                                                                .TITLE  CLENUP
                                                                .IDENT  \V04-002\

                                                                .EXTRN  CLU$GL_CLUB, MAKE_FCB_STALE
                                                                .EXTRN  KILL_BUFFERS, KILL_CACHE
                                                                .EXTRN  WRITE_DIRTY, SWITCH_VOLUME
                                                                .EXTRN  FLUSH_QUO_CACHE

                                                                .PSECT  $CODE$,NOWRT,2

                                                    0BFC 00000  .ENTRY  CLEANUP, Save R2,R3,R4,R5,R6,R7,R8,R9,R11    ; 1216
                                          5B    0000G CF  9E 00002  MOVAB  SWITCH_VOLUME, R11                       ;
                                          58    00DC  CA  9E 00007  MOVAB  220(BASE), R8                            ; 1264
                                             36 AA    D5 0000C     TSTL   54(BASE)                                  ; 1289
                                                      08 13 0000F     BEQL   1$                                    ;
                                    6A       36 AA       36 28 00011  MOVC3  #54, 54(BASE), (BASE)                  ; 1292
                                             36 AA       D4 00016     CLRL   54(BASE)                              ; 1293
                                                         59 D4 00019 1$:  CLRL   CLUSTER                          ; 1296
                                          50    94 AA    D0 0001B     MOVL   -108(BASE), R0                        ; 1297
                                          0B    3C A0    E9 0001F     BLBC   60(R0), 2$                            ;
                                             00000000G 9F    D5 00023     TSTL   @#CLU$GL_CLUB                     ; 1298
                                                         03 13 00029     BEQL   2$                                ;
                                          59             01 D0 0002B     MOVL   #1, CLUSTER                       ; 1300
                                          50    98 AA    D0 0002E 2$:    MOVL   -104(BASE), R0                    ; 1308
                                          52    20 A0    D0 00032     MOVL   32(R0), RVT                           ;
                                       94 AA            52 D1 00036     CMPL   RVT, -108(BASE)                   ; 1311
                                                        08 12 0003A     BNEQ   3$                                ;
                                          54            52 D0 0003C     MOVL   RVT, UCB                          ; 1312
                                          57            01 D0 0003F     MOVL   #1, R7                            ;
                                                        04 11 00042     BRB    4$                                ;
                                          57    0B      A2 9A 00044 3$:     MOVZBL 11(RVT), R7                       ; 1313
                                                        53 D4 00048 4$:     CLRL   J                                 ; 1309
                                                        4A 11 0004A     BRB    9$                                ;
                                       94 AA            52 D1 0004C 5$:     CMPL   RVT, -108(BASE)                   ; 1317
                                                        05 13 00050     BEQL   6$                                ;
                                          54    40 A243  D0 00052     MOVL   64(RVT)[J], UCB                   ; 1318
                                                        54 D5 00057 6$:     TSTL   UCB                               ; 1319
                                                        5B 13 00059     BEQL   9$                                ;
                                          55    34       A4 D0 0005B     MOVL   52(UCB), VCB                      ; 1323
                                          01             53 D1 0005F     CMPL   J, #1                             ; 1325
                                                        15 12 00062     BNEQ   7$                                ;
```

```
                    56        5C    A5  D0 00064          MOVL    92(VCB), QUOTA_CACHE                    ;1333
                                    0F  13 00068          BEQL    7$                                      ;1334
                0A        0B    A6  01 E5 0006A          BBCC    #1, 11(QUOTA_CACHE), 7$                 ;1336
                                    01  DD 0006F          PUSHL   #1                                      ;1339
                              6B    01  FB 00071          CALLS   #1, SWITCH_VOLUME
                          0000G CF  00  FB 00074          CALLS   #0, FLUSH_QUO_CACHE                     ;1340
                05        3A    A4  05  E0 00079 7$:      BBS     #5, 58(UCB), 8$                         ;1348
                13        53    A5  01  E1 0007E          BBC     #1, 83(VCB), 9$                         ;1349
                                    53  DD 00083 8$:      PUSHL   J                                       ;1352
                              6B    01  FB 00085          CALLS   #1, SWITCH_VOLUME
                                    7E  D4 00088          CLRL    -(SP)                                   ;1353
                          0000G CF  01  FB 0008A          CALLS   #1, WRITE_DIRTY
                                    54  DD 0008F          PUSHL   UCB                                     ;1354
                          0000G CF  01  FB 00091          CALLS   #1, KILL_CACHE
                B2              53  57  F3 00096 9$:      AOBLEQ  R7, J, 5$                               ;1309
                                    7E  D4 0009A          CLRL    -(SP)                                   ;1364
                          0000G CF  01  FB 0009C          CALLS   #1, WRITE_DIRTY
                08              6A  04  E5 000A1          BBCC    #4, (BASE), 10$                         ;1369
                              08    AA  DD 000A5          PUSHL   8(BASE)                                 ;1370
                          0000V CF  01  FB 000A8          CALLS   #1, ZERO_WINDOWS
                              53  00D0  CA  D0 000AD 10$:  MOVL    208(BASE), FCB                          ;1379
                                    50  13 000B2          BEQL    14$
                                18  A3  B5 000B4          TSTW    24(FCB)                                 ;1382
                                    1F  12 000B7          BNEQ    11$
                              08    AA  53  D1 000B9          CMPL    FCB, 8(BASE)                            ;1385
                                    37  13 000BD          BEQL    13$
                              50        53  D0 000BF          MOVL    FCB, R0                                 ;1387
                                  0000V 30  000C2          BSBW    SET_DIRINDX
                              2E        50  E8 000C5          BLBS    R0, 13$
                                    53  DD 000C8          PUSHL   FCB                                     ;1390
                          0000V CF  01  FB 000CA          CALLS   #1, DEL_EXTFCB
                                    53  DD 000CF          PUSHL   FCB                                     ;1391
                          0000V CF  01  FB 000D1          CALLS   #1, NUKE_HEAD_FCB
                                    1E  11 000D6          BRB     13$                                     ;1382
                                1C  A3  B5 000D8 11$:     TSTW    28(FCB)                                 ;1398
                                    19  13 000DB          BEQL    13$
                              7E    28  A3  3C 000DD          MOVZWL  40(FCB), -(SP)                          ;1401
                              6B        01  FB 000E1          CALLS   #1, SWITCH_VOLUME
                              0A        59  E8 000E4          BLBS    CLUSTER, 12$                            ;1402
                                4C  A7  DD 000E7          PUSHL   76(FCB)                                 ;1404
                                    01  DD 000EA          PUSHL   #1
                          0000G CF  02  FB 000EC          CALLS   #2, KILL_BUFFERS
                          0000V CF  00  FB 000F1 12$:     CALLS   #0, ZERO_IDX                            ;1405
                          6A  00C00020  8F  CA 000F6 13$:     BICL2   #12582944, (BASE)                       ;1417
                                00D0  CA  D4 000FD          CLRL    208(BASE)                               ;1418
                                  08  A8  D4 00101          CLRL    8(R8)                                   ;1419
                              53  08    AA  D0 00104 14$:     MOVL    8(BASE), FCB                            ;1423
                                    2D  13 00108          BEQL    16$
                0A              6A  0E  E1 0010A          BBC     #14, (BASE), 15$                        ;1432
                                07  59  E9 0010E          BLBC    CLUSTER, 15$                            ;1433
                                    53  DD 00111          PUSHL   FCB                                     ;1435
                          0000G CF  01  FB 00113          CALLS   #1, MAKE_FCB_STALE
                                18  A3  B5 00118 15$:     TSTW    24(FCB)                                 ;1441
                                    1A  12 0011B          BNEQ    16$
                              50        53  D0 0011D          MOVL    FCB, R0                                 ;1443
                                  0000V 30  00120          BSBW    SET_DIRINDX
                              11        50  E8 00123          BLBS    R0, 16$
```

```
                                 53  DD 00126          PUSHL   FCB                                    ; 1447
                                 01  FB 00128          CALLS   #1, DEL_EXTFCB                         ;
                                 53  DD 0012D          PUSHL   FCB                                    ; 1449
                                 01  FB 0012F          CALLS   #1, NUKE_HEAD_FCB                      ;
                             08  AA  D4 00134          CLRL    8(BASE)                                ; 1451
                             50  01  D0 00137 16$:      MOVL    #1, R0                                 ; 1455
                                     04 0013A          RET                                            ; 1457

; Routine Size:  315 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  464          1458  1 GLOBAL ROUTINE ZERO_WINDOWS (FCB) : L_NORM =
;  465          1459  1
;  466          1460  1 !++
;  467          1461  1 !
;  468          1462  1 ! FUNCTIONAL DESCRIPTION:
;  469          1463  1 !
;  470          1464  1 !       This routine invalidates all windows currently in use on the
;  471          1465  1 !       indicated FCB. This routine must be executed in kernel mode.
;  472          1466  1 !
;  473          1467  1 ! CALLING SEQUENCE:
;  474          1468  1 !       ZERO_WINDOWS (ARG1)
;  475          1469  1 !
;  476          1470  1 ! INPUT PARAMETERS:
;  477          1471  1 !       ARG1: address of FCB
;  478          1472  1 !
;  479          1473  1 ! IMPLICIT INPUTS:
;  480          1474  1 !       CURRENT_WINDOW: address of caller's window, if any
;  481          1475  1 !
;  482          1476  1 ! OUTPUT PARAMETERS:
;  483          1477  1 !       NONE
;  484          1478  1 !
;  485          1479  1 ! IMPLICIT OUTPUTS:
;  486          1480  1 !       NONE
;  487          1481  1 !
;  488          1482  1 ! ROUTINE VALUE:
;  489          1483  1 !       NONE
;  490          1484  1 !
;  491          1485  1 ! SIDE EFFECTS:
;  492          1486  1 !       all windows marked empty, caller's turned
;  493          1487  1 !
;  494          1488  1 !--
;  495          1489  1
;  496          1490  2 BEGIN
;  497          1491  2
;  498          1492  2 MAP
;  499          1493  2         FCB             : REF BBLOCK;
;  500          1494  2
;  501          1495  2 LOCAL
;  502          1496  2         P               : REF BBLOCK,   ! window pointer
;  503          1497  2         DUMMY,                          ! dummy storage for REMQUE return
;  504          1498  2         WINDOW_SEGMENT  : REF BBLOCK,   ! pointer to window segment
;  505          1499  2         NEXT_SEGMENT    : REF BBLOCK;   ! pointer to window after next one
;  506          1500  2
;  507          1501  2 BASE_REGISTER;
;  508          1502  2
;  509          1503  2 EXTERNAL ROUTINE
;  510          1504  2         DEALLOCATE      : L_NORM;       ! deallocate dynamic memory
;  511          1505  2
;  512          1506  2 ! Loop through the window list off the FCB, zeroing all the retrieval pointer
;  513          1507  2 ! counts. Then turn the user's window to VBN 1 if it exists.
;  514          1508  2 !
;  515          1509  2
;  516          1510  2 P = .FCB[FCB$L_WLFL];
;  517          1511  2
;  518          1512  2 UNTIL .P EQL FCB[FCB$L_WLFL] DO
;  519          1513  3     BEGIN
;  520          1514  3     P[WCB$W_NMAP] = 0;
```

```
;  521          1515  3      WINDOW_SEGMENT = .P[WCB$L_LINK];
;  522          1516  3      UNTIL .WINDOW_SEGMENT EQL 0
;  523          1517  3      DO
;  524          1518  4          BEGIN
;  525          1519  4          NEXT_SEGMENT = .WINDOW_SEGMENT[WCB$L_LINK];
;  526          1520  4          REMQUE (.WINDOW_SEGMENT, DUMMY);
;  527          1521  4          DEALLOCATE (.WINDOW_SEGMENT);
;  528          1522  4          WINDOW_SEGMENT = .NEXT_SEGMENT;
;  529          1523  3          END;
;  530          1524  3      P[WCB$L_LINK] = 0;
;  531          1525  3      P[WCB$V_COMPLETE] = 0;
;  532          1526  3      P = .P[WCB$L_WLFL];
;  533          1527  2      END;
;  534          1528  2
;  535          1529  2  ! ***** Note: When handling of window misses goes into its final form,
;  536          1530  2  ! this routine must also scan the I/O queue on the UCB and look for I/O
;  537          1531  2  ! into the blocks just deallocated. All such requests must be yanked out
;  538          1532  2  ! of the queue and routed to the ACP for error processing.
;  539          1533  2
;  540          1534  2  RETURN 1;
;  541          1535  2
;  542          1536  1  END;                                   ! end of routine ZERO_WINDOWS


                                                        .EXTRN  DEALLOCATE

                                        003C 00000      .ENTRY  ZERO_WINDOWS, Save R2,R3,R4,R5          ; 1458
                                50      04   AC  D0 00C02  MOVL   FCB, R0                             ; 1510
                                52      10   A0  D0 00006  MOVL   16(R0), P                           ;
                    50      04  AC          10  C1 0000A 1$: ADDL3  #16, FCB, R0                      ; 1512
                                50          52  D1 0000F  CMPL   P, R0                               ;
                                            28  13 00012  BEQL   4$                                   ;
                                        16  A2  B4 00014  CLRW   22(P)                               ; 1514
                                53      20  A2  D0 00017  MOVL   32(P), WINDOW_SEGMENT              ; 1515
                                            13  13 0001B 2$: BEQL   3$                               ; 1516
                                54      20  A3  D0 0001D  MOVL   32(WINDOW_SEGMENT), NEXT_SEGMENT   ; 1519
                                55          63  0F 00021  REMQUE (WINDOW_SEGMENT), DUMMY            ; 1520
                                            53  DD 00024  PUSHL  WINDOW_SEGMENT                     ; 1521
                            0000G   CF      01  FB 00026  CALLS  #1, DEALLOCATE                     ;
                                53          54  D0 0002B  MOVL   NEXT_SEGMENT, WINDOW_SEGMENT       ; 1522
                                            EB  11 0002E  BRB    2$                                 ; 1516
                                        20  A2  D4 0003C 3$: CLRL   32(P)                            ; 1524
                                0B      A2      20  8A 00033  BICB2  #32, 11(P)                     ; 1525
                                52              62  D0 00037  MOVL   (P), P                         ; 1526
                                                CE  11 0003A  BRB    1$                             ; 1512
                                50              01  D0 0003C 4$: MOVL   #1, R0                      ; 1534
                                                    04 0003F  RET                                   ; 1536

; Routine Size:  64 bytes,    Routine Base:  $CODE$ + 013B
```

```
;  544          1537  1 GLOBAL ROUTINE ZERO_IDX : L_NORM NOVALUE =
;  545          1538  1
;  546          1539  1 !++
;  547          1540  1 !
;  548          1541  1 ! FUNCTIONAL DESCRIPTION:
;  549          1542  1 !
;  550          1543  1 !       This routine initializes the index in a directory FCB to an unknown
;  551          1544  1 !       state. It will be rebuilt with the next several lookups.
;  552          1545  1 !       It also bumps the sequence count to indicate a change in contents.
;  553          1546  1 !
;  554          1547  1 !
;  555          1548  1 ! CALLING SEQUENCE:
;  556          1549  1 !       ZERO_IDX ()
;  557          1550  1 !
;  558          1551  1 ! INPUT PARAMETERS:
;  559          1552  1 !       NONE
;  560          1553  1 !
;  561          1554  1 ! IMPLICIT INPUTS:
;  562          1555  1 !       DIR_FCB: directory FCB to init
;  563          1556  1 !
;  564          1557  1 ! OUTPUT PARAMETERS:
;  565          1558  1 !       NONE
;  566          1559  1 !
;  567          1560  1 ! IMPLICIT OUTPUTS:
;  568          1561  1 !       NONE
;  569          1562  1 !
;  570          1563  1 ! ROUTINE VALUE:
;  571          1564  1 !       1
;  572          1565  1 !
;  573          1566  1 ! SIDE EFFECTS:
;  574          1567  1 !       directory index zeroed
;  575          1568  1 !
;  576          1569  1 !--
;  577          1570  1
;  578          1571  2 BEGIN
;  579          1572  2
;  580          1573  2 BIND_COMMON;
;  581          1574  2
;  582          1575  2 LOCAL
;  583          1576  2         DIRINDX : REF BBLOCK FIELD (DIRC);
;  584          1577  2
;  585          1578  2 DIR_FCB[FCB$W_DIRSEQ] = .DIR_FCB[FCB$W_DIRSEQ] + 1;
;  586          1579  2
;  587          1580  2 IF (DIRINDX = .DIR_FCB [FCB$L_DIRINDX]) NEQ 0
;  588          1581  2 THEN
;  589          1582  2     DIRINDX [DIRC$W_INUSE] = 0;
;  590          1583  2
;  591          1584  1 END;                                    ! end of routine ZERO_IDX

                                 0000 00000          .ENTRY  ZERO_IDX, Save nothing                  ; 1537
                     50  00D0    CA  D0 00002        MOVL    208(BASE), R0                           ; 1578
                       42    A0  B6 0000?            INCW    66(R0)                                  ;
                     50  00D0    CA  D0 0000A        MOVL    208(BASE), R0                           ; 1580
```

```
                     50      00B0    C0  DD 0000F          MOVL    176(R0), DIRINDX                ;
                                     02  13 00014          BEQL    1$                              ;
                                     60  B4 00016          CLRW    (DIRINDX)                       ; 1582
                                         04 00018 1$:      RET                                     ; 1584

; Routine Size:  25 bytes,    Routine Base:  $CODE$ + 017B
```

```
;  593          1585  1 GLOBAL ROUTINE ERR_CLEANUP : L_NORM =
;  594          1586  1
;  595          1587  1 !++
;  596          1588  1 !
;  597          1589  1 ! FUNCTIONAL DESCRIPTION:
;  598          1590  1 !
;  599          1591  1 !       This routine performs the cleanup needed after a file
;  600          1592  1 !       operation that has terminated in an error.
;  601          1593  1 !
;  602          1594  1 ! CALLING SEQUENCE:
;  603          1595  1 !       ERR_CLEANUP ()
;  604          1596  1 !
;  605          1597  1 ! INPUT PARAMETERS:
;  606          1598  1 !       NONE
;  607          1599  1 !
;  608          1600  1 ! IMPLICIT INPUTS:
;  609          1601  1 !       CLEANUP_FLAGS: indicate specific actions to do
;  610          1602  1 !
;  611          1603  1 ! OUTPUT PARAMETERS:
;  612          1604  1 !       NONE
;  613          1605  1 !
;  614          1606  1 ! IMPLICIT OUTPUTS:
;  615          1607  1 !       NONE
;  616          1608  1 !
;  617          1609  1 ! ROUTINE VALUE:
;  618          1610  1 !       NONE
;  619          1611  1 !
;  620          1612  1 ! SIDE EFFECTS:
;  621          1613  1 !       file deaccessed if necessary
;  622          1614  1 !       channel window pointer cleared
;  623          1615  1 !
;  624          1616  1 !--
;  625          1617  1
;  626          1618  2 BEGIN
;  627          1619  2
;  628          1620  2 BIND_COMMON;
;  629          1621  2
;  630          1622  2 DIR_CONTEXT_DEF;
;  631          1623  2
;  632          1624  2 EXTERNAL ROUTINE
;  633          1625  2         REBLD_PRIM_FCB  : L_NORM NOVALUE, ! rebuild primary fcb from header
;  634          1626  2         BUILD_EXT_FCBS  : L_NORM NOVALUE, ! build extension fcb chain
;  635          1627  2         ALLOCATION_UNLOCK : L_NORM NOVALUE, ! release allocation lock
;  636          1628  2         KILL_DINDX      : L_NORM NOVALUE, ! release directory index block
;  637          1629  2         PMS_END_SUB     : L_NORM,         ! end metering of current subfunction
;  638          1630  2         CLOSE_FILE      : L_NORM,         ! close internal file
;  639          1631  2         DEACC_QFILE     : L_NORM,         ! deaccess the quota file
;  640          1632  2         DEALLOCATE      : L_NORM,         ! deallocate dynamic memory
;  641          1633  2         SEND_SYMBIONT   : L_NORM ADDRESSING_MODE (GENERAL),
;  642          1634  2                                           ! send file to job controller
;  643          1635  2         SWITCH_VOLUME   : L_NORM,         ! switch to desired volume
;  644          1636  2         RESTORE_DIR     : L_NORM,         ! restore directory context
;  645          1637  2         DIR_SCAN        : L_NORM,         ! scan directory file
;  646          1638  2         MAKE_ENTRY      : L_NORM,         ! create new directory entry
;  647          1639  2         REMOVE          : L_NORM,         ! remove a directory entry
;  648          1640  2         READ_BLOCK      : L_NORM,         ! read a disk block
;  649          1641  2         MARK_DIRTY      : L_NORM,         ! mark disk block for write back
```

```
;  650          1642  2          WRITE_BLOCK     : L_NORM,          ! write a disk block
;  651          1643  2          DELETE_FILE     : L_NORM,          ! delete a file
;  652          1644  2          DELETE_FID      : L_NORM,          ! delete a file number
;001 !CDS0023   1645  2          RETURN_BLOCKS   : L_NORM ADDRESSING_MODE (GENERAL), ! return blocks to storage map
;  654-1        1646  2          TRUNCATE        : L_NORM,          ! file truncate routine
;  655          1647  2          INVALIDATE      : L_NORM,          ! invalidate a buffer
;  656          1648  2          READ_HEADER     : L_NORM,          ! read file header
;  657          1649  2          CHECKSUM        : L_NORM,          ! checksum fi e header
;  658          1650  2          REMAP_FILE      : L_NORM;          ! rebuild the windows for a file
;  659          1651  2
;  660          1652  2
;  661          1653  2  ! If a subfunction was being executed, turn off metering now.
;  662          1654  2  !
;  663          1655  2
;  664          1656  2  IF .PMS_SUB_NEST NEQ 0
;  665          1657  2  THEN
;  666          1658  3      BEGIN
;  667          1659  3      PMS_SUB_NEST = 1;
;  668          1660  3      PMS_END_SUB ();
;  669          1661  2      END;
;  670          1662  2
;  671          1663  2  ! We repeat the entire procedure twice if a secondary file operation was
;  672          1664  2  ! in progress (indicated by non-zero saved context).
;  673          1665  2  !
;  674          1666  2
;  675          1667  2  WHILE 1 DO
;  676          1668  3  BEGIN
;  677          1669  3
;  678          1670  3  ! Locals are declared here to prevent their scope from extending around the
;  679          1671  3  ! entire main loop and raising havoc with register assignment.
;  680          1672  3  !
;  681          1673  3
;  682          1674  3  LOCAL
;  683          1675  3          NAME_DESC       : BBLOCK [FND_LENGTH], ! file name descriptor block
;  684          1676  3          HEADER          : REF BBLOCK,      ! address of file header
;  685          1677  3          IDENT_AREA      : REF BBLOCK,      ! ident area of file header
;  686          1678  3          FCB             : REF BBLOCK,      ! FCB pointer
;  687          1679  3          WINDOW_SEGMENT  : REF BBLOCK,      ! address of the next window segment
;  688          1680  3          NEXT_SEGMENT    : REF BBLOCK,      ! address of one beyond the next window
;  689          1681  3          RECADDR         : REF BBLOCK,      ! address of directory record
;  690          1682  3          DIR_FLAGS       : BITVECTOR [32],  ! directory cleanup flags
;  691          1683  3          UNREC_LOCAL,                       ! local copy of UNREC_COUNT
;  692          1684  3          FID_LOCAL,                         ! local copy of NEW_FID
;  693          1685  3          T1,                                ! random temps
;  694          1686  3          T2,
;  695          1687  3          T3;
;  696          1688  3
;  697          1689  3  ! Show that cleanup is in progress.
;  698          1690  3  !
;  699          1691  3
;  700          1692  3  CLEANUP_FLAGS[CLF_CLEANUP] = 1;
;  701          1693  3
;  702          1694  3  ! If the ref count on the primary fcb was biased in fid_to_spec, remove
;  703          1695  3  ! the bias.
;  704          1696  3  !
;  705          1697  3
;  706          1698  3  IF TESTBITSC (CLEANUP_FLAGS [CLF_PFCB_REF_UP])
```

```
;  707          1699  3 THEN
;  708          1700  3     PRIMARY_FCB [FCBSW_REFCNT] = .PRIMARY_FCB [FCBSW_REFCNT] - 1;
;  709          1701  3
;  710          1702  3 ! If an internal file is open, close it first.
;  711          1703  3 !
;  712          1704  3
;  713          1705  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_CLOSEFILE])
;  714          1706  3 THEN CLOSE_FILE (.CURRENT_WINDOW);
;  715          1707  3
;  716          1708  3 ! Invalidate the file ID cache, if necessary.
;  717          1709  3 !
;  718          1710  3
;  719          1711  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_FLUSHFID])
;  720          1712  3 THEN KERNEL_CALL (FLUSH_FIDCACHE);
;  721          1713  3
;  722          1714  3 ! Deaccess the quota file, if we were in the final stages of a quota file
;  723          1715  3 ! enable.
;  724          1716  3 !
;  725          1717  3
;  726          1718  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DEACCQFILE])
;  727          1719  3 THEN KERNEL_CALL (DEACC_QFILE);
;  728          1720  3
;  729          1721  3 ! If there is a file header resident, it probably needs to be checksummed.
;  730          1722  3 !
;  731          1723  3
;  732          1724  3 IF .FILE_HEADER NEQ 0
;  733          1725  3 THEN CHECKSUM (.FILE_HEADER);
;  734          1726  3
;  735          1727  3 ! Clean out the window pointer in the user's channel if necessary.
;  736          1728  3 !
;  737          1729  3
;  738          1730  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_ZCHANNEL])
;  739          1731  3 THEN KERNEL_CALL (ZERO_CHANNEL);
;  740          1732  3
;  741          1733  3 ! If there are unrecorded blocks allocated from the storage map, return them.
;  742          1734  3 !
;  743          1735  3
;  744          1736  3 IF (UNREC_LOCAL = .UNREC_COUNT) NEQ 0
;  745          1737  3 THEN
;  746          1738  4     BEGIN
;  747          1739  4     UNREC_COUNT = 0;
;  748          1740  4     SWITCH_VOLUME (.UNREC_RVN);
;  749          1741  4     RETURN_BLOCKS (.UNREC_LBN, .UNREC_LOCAL, DO_NOT_ERASE);
;  750          1742  3     END;
;  751          1743  3
;  752          1744  3 ! If there is a dangling file ID (from a partial create or header extension),
;  753          1745  3 ! dispose of it.
;  754          1746  3 !
;  755          1747  3
;  756          1748  3 IF (FID_LOCAL = .NEW_FID) NEQ 0
;  757          1749  3 THEN
;  758          1750  4     BEGIN
;  759          1751  4     NEW_FID = 0;
;  760          1752  4     SWITCH_VOLUME (.NEW_FID_RVN);
;  761          1753  4     DELETE_FID (.FID_LOCAL);
;  762          1754  3     END;
;  763          1755  3
```

```
;  764          1756  3 ! Get back the primary file header of the file in process.
;  765          1757  3 !
;  766          1758  3
;  767          1759  3 HEADER = 0;
;  768          1760  3 IF .FILE_HEADER NEQ 0
;  769          1761  3 THEN
;  770          1762  4     BEGIN
;  771          1763  4     FILE_HEADER = 0;
;  772          1764  4     IF (CURR_LCKINDX = .PRIM_LCKINDX) NEQ 0
;  773          1765  4     THEN
;  774          1766  5         HEADER = READ_HEADER ((IF .CURRENT_FIB NEQ 0
;  775          1767  5                                THEN CURRENT_FIB[FIB$W_FID]
;  776          1768  4                                ELSE 0),
;  777          1769  4                               .PRIMARY_FCB);
;  778          1770  3     END;
;  779          1771  3
;  780          1772  3 ! Send the file to the job controller if it is to be spooled.
;  781          1773  3 !
;  782          1774  3
;  783          1775  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DOSPOOL])
;  784          1776  3 THEN
;  785          1777  4     BEGIN
;  786          1778  4
;  787          1779  4 ! Make sure the allocation lock is released before sending it
;  788          1780  4 ! to the symbiont to avoid potential deadlock with the symbiont.
;  789          1781  4 !
;  790          1782  4
;  791          1783  4     ALLOCATION_UNLOCK ();
;  792          1784  4     SEND_SYMBIONT (.HEADER, .PRIMARY_FCB);
;  793          1785  3     END;
;  794          1786  3
;  795          1787  3 ! Deaccess the file if requested.
;  796          1788  3 !
;  797          1789  3
;  798          1790  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DEACCESS])
;  799          1791  3 THEN KERNEL_CALL (MAKE_DEACCESS);
;  800          1792  3
;  801          1793  3 ! Deallocate the window block if called for.
;  802          1794  3 !
;  803          1795  3
;  804          1796  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DELWINDOW])
;  805          1797  3 THEN
;  806          1798  3     IF .CURRENT_WINDOW NEQ 0
;  807          1799  3     THEN
;  808          1800  4         BEGIN
;  809          1801  4         WINDOW_SEGMENT = .CURRENT_WINDOW;
;  810          1802  4         DO
;  811          1803  5             BEGIN
;  812          1804  5             NEXT_SEGMENT = .WINDOW_SEGMENT[WCB$L_LINK];
;  813          1805  5             KERNEL_CALL (DEALLOCATE, .WINDOW_SEGMENT);
;  814          1806  5             WINDOW_SEGMENT = .NEXT_SEGMENT;
;  815          1807  5             END
;  816          1808  4         UNTIL .WINDOW_SEGMENT EQL 0;
;  817          1809  4         CURRENT_WINDOW = 0;
;  818          1810  3         END;
;  819          1811  3
;  820          1812  3 ! Fix the file header back link, if it was modified.
```

```
;  821          1813  3 !
;  822          1814  3
;  823          1815  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_FIXLINK])
;  824          1816  3 THEN IF .HEADER NEQ 0
;  825          1817  3 THEN
;  826          1818  4     BEGIN
;  827          1819  4     CH$MOVE (FID$C_LENGTH, PREV_LINK, HEADER[FH2$W_BACKLINK]);
;  828          1820  4     IDENT_AREA = .HEADER + .HEADER[FH2$B_IDOFFSET]*2;
;  829          1821  4     CH$MOVE (MINU (FILENAME_LENGTH, FI2$S_FILENAME), PREV_INAME,
;  830          1822  4              IDENT_AREA[FI2$T_FILENAME]);
;  831          1823  4     CH$MOVE (MINU (FILENAME_LENGTH-FI2$S_FILENAME, FI2$S_FILENAMEXT),
;  832          1824  4              PREV_INAME+FI2$S_FILENAME,
;  833          1825  4              IDENT_AREA[FI2$T_FILENAMEXT]);
;  834          1826  4     CHECKSUM (.HEADER);
;  835          1827  4     MARK_DIRTY (.HEADER);
;  836          1828  3     END;
;  837          1829  3
;  838          1830  3 ! If a file deletion is called for, do it.  This is either a create that
;  839          1831  3 ! failed later on, or a real delete.
;  840          1832  3 !
;  841          1833  3
;  842          1834  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DELFILE])
;  843          1835  3 THEN IF .HEADER NEQ 0
;  844          1836  3 THEN
;  845          1837  4     BEGIN
;  846          1838  4     IF .PRIMARY_FCB NEQ 0
;  847          1839  4     THEN
;  848          1840  4         IF .PRIMARY_FCB [FCB$L_DIRINDX] NEQ 0
;  849          1841  4         THEN
;  850          1842  4             KILL_DINDX (.PRIMARY_FCB);
;  851          1843  4
;  852          1844  4     CLEANUP_FLAGS[CLF_TRUNCATE] = 0;     ! no truncate necessary after a delete
;001 !CDS0023   1845  4     CLEANUP_FLAGS [CLF_FIXFCB] = 0;      ! no fcb fixup ever necessary after delete
;  853          1846  4     DELETE_FILE (.CURRENT_FIB, .HEADER);
;  854          1847  3     END;
;  855          1848  3
;  856          1849  3 ! If an extend operation failed, truncate the file.
;  857          1850  3 !
;  858          1851  3
;  859          1852  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_TRUNCATE])
;  860          1853  3 THEN IF .HEADER NEQ 0
;  861          1854  3 THEN
;  862          1855  4     BEGIN
;  863          1856  4     T1 = .CURRENT_FIB[FIB$L_EXSZ];       ! save the data returned by EXTEND
;  864          1857  4     T2 = .CURRENT_FIB[FIB$L_EXVBN];      ! so it won't be smashed by TRUNCATE
;  865          1858  4     T3 = .USER_STATUS[1];
;  866          1859  4     CURRENT_FIB[FIB$L_EXSZ] = 0;
;  867          1860  4     TRUNCATE (.CURRENT_FIB, .HEADER, .T2);
;  868          1861  4     HEADER = .FILE_HEADER;                          ! follow buffer shuffling
;  869          1862  4     CURRENT_FIB[FIB$L_EXSZ] = .T1;
;  870          1863  4     CURRENT_FIB[FIB$L_EXVBN] = .T2;
;  871          1864  4     USER_STATUS[1] = .T3;
;  872          1865  4     CLEANUP_FLAGS[CLF_INVWINDOW] = 0;    ! windows were never extended, so no need
;  873          1866  4     CHECKSUM (.HEADER);
;  874          1867  3     END;
;  875          1868  3
;  876          1869  3 ! Various errors leave the file control block screwed up. If needed,
```

```
;  877          1870  3 ! rebuild it and its extensions from scratch.
;  878          1871  3 !
;  879          1872  3
;  880          1873  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_FIXFCB])
;  881          1874  3 AND .HEADER NEQ 0
;  882          1875  3 THEN
;  883          1876  4     BEGIN
;  884          1877  4
;  885          1878  4     REBLD_PRIM_FCB (.PRIMARY_FCB, .HEADER);
;  886          1879  4
;  887          1880  4     BUILD_EXT_FCBS (.HEADER);
;  888          1881  4
;  889          1882  3     END;
;  890          1883  3
;  891          1884  3 ! Cleanup any cathedral windows which have broken.
;  892          1885  3 !
;  893          1886  3
;  894          1887  3 IF TESTBITSC (CLEANUP_FLAGS[CLF_REMAP]) THEN REMAP_FILE ();
;  895          1888  3
;  896          1889  3 ! Do directory operation cleanups. We could have entered a new file, removed
;  897          1890  3 ! an old one, or both, or done a supersede. A supersede is a replacement of
;  898          1891  3 ! the FID for the same name, type, and version.
;  899          1892  3 !
;  900          1893  3
;  901          1894  3 DIR_FLAGS = .CLEANUP_FLAGS;
;  902          1895  3 CLEANUP_FLAGS[CLF_SUPERSEDE] = 0;
;  903          1896  3 CLEANUP_FLAGS[CLF_REENTER] = 0;
;  904          1897  3 CLEANUP_FLAGS[CLF_REMOVE] = 0;
;  905          1898  3
;  906          1899  3 IF .DIR_FLAGS[CLF_SUPERSEDE]
;  907          1900  3 OR .DIR_FLAGS[CLF_REENTER]
;  908          1901  3 OR .DIR_FLAGS[CLF_REMOVE]
;  909          1902  3 THEN
;  910          1903  4     BEGIN
;  911          1904  4     SWITCH_VOLUME (.CURRENT_FIB[FIB$W_DID_RVN]);
;  912          1905  4
;  913          1906  4 ! Buffer pool thrashing may have kicked out the directory block we need.
;  914          1907  4 ! re-read it and recompute the buffer pointers.
;  915          1908  4 !
;  916          1909  4
;  917          1910  4     IF .DIR_ENTRY NEQ 0
;  918          1911  4     THEN RESTORE_DIR (DIR_CONTEXT);
;  919          1912  4
;  920          1913  4 ! If a directory entry needs to be removed, do so. Pointers are all set
;  921          1914  4 ! up for the REMOVE routine.
;  922          1915  4 !
;  923          1916  4
;  924          1917  4     IF .DIR_FLAGS[CLF_REMOVE]
;  925          1918  4     THEN REMOVE (0);
;  926          1919  4
;  927          1920  4 ! If a directory entry needs to be re-entered, do so. If the entry was
;  928          1921  4 ! removed theough an auto-purge, we need to rescan to the point of
;  929          1922  4 ! removal because a directory shuffle may have invalidated the
;  930          1923  4 ! pointers. Construct a name descriptor from the saved name and version
;  931          1924  4 ! and ca.l the enter routine.
;  932          1925  4 !
;  933          1926  4
```

```
;  934          1927  4      IF .DIR_FLAGS[CLF_REENTER]
;  935          1928  4      THEN
;  936          1929  5          BEGIN
;  937          1930  5          CH$FILL (0, FND_LENGTH, NAME_DESC);
;  938          1931  5          NAME_DESC[FND_COUNT] = .PREV_NAME[0];
;  939          1932  5          NAME_DESC[FND_STRING] = PREV_NAME[1];
;  940          1933  5          NAME_DESC[FND_VERSION] = .PREV_VERSION;
;  941          1934  5          IF .DIR_FLAGS[CLF_SUPERSEDE]
;  942          1935  5          THEN
;  943          1936  6              BEGIN
;  944          1937  6              LAST_ENTRY[0] = 0;
;  945          1938  6              DIR_SCAN (NAME_DESC, 0, 0, 0, 0, 0, -1);
;  946          1939  6              CH$MOVE (FID$C_LENGTH, SUPER_FID, CURRENT_FIB[FIB$W_FID]);
;  947          1940  5              END;
;  948          1941  5          MAKE_ENTRY (NAME_DESC, .CURRENT_FIB);
;  949          1942  5          CLEANUP_FLAGS[CLF_REMOVE] = 0;
;  950          1943  5          WRITE_BLOCK (.DIR_BUFFER);
;  951          1944  4          END;
;  952          1945  4
;  953          1946  4  ! A supersede cleanup consists simply of replacing the superseded file ID
;  954          1947  4  ! in the directory record. Note that the supersede bit could also be set
;  955          1948  4  ! by a create/auto-purge, which also sets the remove and enter bits, and
;  956          1949  4  ! is handled above.
;  957          1950  4  !
;  958          1951  4
;  959          1952  4      IF .DIR_FLAGS[CLF_SUPERSEDE]
;  960          1953  4      AND NOT .DIR_FLAGS[CLF_REENTER]
;  961          1954  4      AND NOT .DIR_FLAGS[CLF_REMOVE]
;  962          1955  4      THEN
;  963          1956  5          BEGIN
;  964          1957  5          DIR_VERSION[DIR$W_VERSION] = .PREV_VERSION;
;  965          1958  5          CH$MOVE (FIB$S_FID, SUPER_FID, DIR_VERSION[DIR$W_FID]);
;  966          1959  5          MARK_DIRTY (.DIR_BUFFER);
;  967          1960  5          END
;  968          1961  5
;  969          1962  5
;  970          1963  3      END;                                    ! end of directory cleanup processing
;  971          1964  3
;  972          1965  3  ! Copy the saved context, if any back into the primary context and repeat
;  973          1966  3  ! the cleanup.
;  974          1967  3  !
;  975          1968  3
;  976          1969  3  IF .CONTEXT_SAVE EQL 0 THEN EXITLOOP;
;  977          1970  3  CH$MOVE (CONTEXT_SIZE, CONTEXT_SAVE, CONTEXT_START);
;  978          1971  3  CONTEXT_SAVE = 0;
;  979          1972  3
;  980          1973  2  END;                                        ! end of major loop
;  981          1974  2
;  982          1975  2  RETURN 1;
;  983          1976  2
;  984          1977  1  END;                                        ! end of routine ERR_CLEANUP


                                                .EXTRN  REBLD_PRIM_FCB, BUILD_EXT_FCBS
                                                .EXTRN  ALLOCATION_UNLOCK
                                                .EXTRN  KILL_DINDX, PMS_END_SUB
```

```
                                        .EXTRN  CLOSE_FILE, DEACC_QFILE
                                        .EXTRN  SEND_SYMBIONT, RESTORE_DIR
                                        .EXTRN  DIR_SCAN, MAKE_ENTRY
                                        .EXTRN  REMOVE, READ_BLOCK
                                        .EXTRN  MARK_DIRTY, QRITE_BLOCK
                                        .EXTRN  DELETE_FILE, DELETE_FID
                                        .EXTRN  RETURN_BLOCKS, TRUNCATE
                                        .EXTRN  INVALIDATE, READ_HEADER
                                        .EXTRN  CHECKSUM, REMAP_FILE

                            0BFC 00000  .ENTRY  ERR_CLEANUP, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 1585
                                                R11
                         5E      10 C2 00002    SUBL2   #16, SP
                              80 AA 9F 00005    PUSHAB  -128(BASE)                          ; 1618
                              08 AA 9F 00008    PUSHAB  8(BASE)
                         59   10 AA 9E 0000B    MOVAB   16(BASE), R9
                         57 00DC CA 9E 0000F    MOVAB   220(BASE), R7
                         5B 01A8 CA 9E 00014    MOVAB   424(BASE), R11
                          0908 CA D5 00019      TSTL    2312(BASE)                          ; 1656
                               0A 13 0001D      BEQL    1$
                  0908 CA      01 D0 0001F      MOVL    #1, 2312(BASE)                      ; 1659
                  0000G CF     00 FB 00024      CALLS   #0, PMS_END_SUB                     ; 1660
                    01 AA      02 88 00029 1$:  BISB2   #2, 1(BASE)                         ; 1692
                07     6A      0F E5 0002D      BBCC    #15, (BASE), 2$                     ; 1698
                         50   00 BE D0 00031    MOVL    @0(SP), R0                          ; 1700
                              18 A0 B7 00035    DECW    24(R0)
                08       6A      18 E5 00038 2$:  BBCC    #24, (BASE), 3$                  ; 1705
                              0C AA DD 0003C    PUSHL   12(BASE)                            ; 1706
                  0000G CF     01 FB 0003F      CALLS   #1, CLOSE_FILE
                05       6A      13 E5 00044 3$:  BBCC    #19, (BASE), 4$                  ; 1711
                  0000V CF     00 FB 00048      CALLS   #0, FLUSH_FIDCACHE                  ; 1712
                05       6A      19 E5 0004D 4$:  BBCC    #25, (BASE), 5$                  ; 1718
                  0000G CF     00 FB 00051      CALLS   #0, DEACC_QFILE                     ; 1719
                              04 AA D5 00056 5$:  TSTL    4(BASE)                          ; 1724
                                 08 13 00059      BEQL    6$
                              04 AA DD 0005B    PUSHL   4(BASE)                             ; 1725
                  0000G CF     01 FB 0005E      CALLS   #1, CHECKSUM
                05       6A      11 E5 00063 6$:  BBCC    #17, (BASE), 7$                  ; 1730
                  0000V CF     00 FB 00067      CALLS   #0, ZERO_CHANNEL                    ; 1731
                         52   28 AA D0 0006C 7$:  MOVL    40(BASE), UNREC_LOCAL            ; 1736
                              19 13 00070       BEQL    8$
                              2R AA D4 00072    CLRL    40(BASE)                            ; 1739
                              2C AA DD 00075    PUSHL   44(BASE)                            ; 1740
                  0000G CF     01 FB 00078      CALLS   #1, SWITCH_VOLUME
                               7E D4 0007D      CLRL    -(SP)                               ; 174'
                               52 DD 0007F      PUSHL   UNREC_LOCAL
                              24 AA DD 00081    PUSHL   36(BASE)
              00000000G   00      03 FB 00084   CALLS   #3, RETURN_BLOCKS
                         52   A8 AA D0 00088 8$:  MOVL    -88(BASE), FID_LOCAL             ; 1748
                              12 13 0008F       BEQL    9$
                              A8 AA D4 00091    CLRL    -88(BASE)                           ; 1751
                              AC AA DD 00094    PUSHL   -84(BASE)                           ; 1752
                  0000G CF     01 FB 00097      CALLS   #1, SWITCH_VOLUME
                               52 DD 0009C      PUSHL   FID_LOCAL                           ; 1753
                  0000G CF     01 FB 0009E      CALLS   #1, DELETE_FID
                               56 D4 000A3 9$:  CLRL    HEADER                              ; 1759
                              04 AA D5 000A5    TSTL    4(BASE)                             ; 1760
```

```
                                        23  13 000A8         BEQL    12$                                     ;
                                 04     AA  D4 000AA         CLRL    4(BASE)                                 ; 1763
               14      AA        18     AA  D0 000AD         MOVL    24(BASE), 20(BASE)                      ; 1764
                                        19  13 000B2         BEQL    12$                                     ;
                                 00     BE  DD 000B4         PUSHL   @0(SP)                                  ; 1769
                                        69  D5 000B7         TSTL    (R9)                                    ; 1766
                                        08  13 000B9         BEQL    10$                                     ;
          50           69               04  C1 000BB         ADDL3   #4, (R9), R0                            ; 1767
                                        50  DD 000BF         PUSHL   R0                                      ;
                                        02  11 000C1         BRB     11$                                     ;
                                        7E  D4 000C3  10$:   CLRL    -(SP)                                   ;
            0000G      CF               02  FB 000C5  11$:   CALLS   #2, READ_HEADER                         ; 1766
                       56               50  D0 000CA         MOVL    R0, HEADER                              ;
          11           6A               02  E5 000CD  12$:   BBCC    #2, (BASE), 13$                         ; 1775
            0000G      CF               00  FB 000D1         CALLS   #0, ALLOCATION_UNLOCK                   ; 1783
                                 00     BE  DD 000D6         PUSHL   @0(SP)                                  ; 1784
                                        56  DD 000D9         PUSHL   HEADER                                  ;
       000000000G      00               02  FB 000DB         CALLS   #2, SEND_SYMBIONT                       ;
          05           6A               10  E5 000E2  13$:   BBCC    #16, (BASE), 14$                        ; 1790
            0000V      CF               00  FB 000E6         CALLS   #0, MAKE_DEACCESS                       ; 1791
          1C           6A               1A  E5 000EB  14$:   BBCC    #26, (BASE), 16$                        ; 1796
                                 0C     AA  D5 000EF         TSTL    12(BASE)                                ; 1798
                                        17  13 000F2         BEQL    16$                                     ;
                       52        0C     AA  D0 000F4         MOVL    12(BASE), WINDOW_SEGMENT                ; 1801
                       53        20     A2  D0 000F8  15$:   MOVL    32(WINDOW_SEGMENT), NEXT_SEGMENT        ; 1804
                                        52  DD 000FC         PUSHL   WINDOW_SEGMENT                          ; 1805
            0000G      CF               01  FB 000FE         CALLS   #1, DEALLOCATE                          ;
                       52               53  D0 00103         MOVL    NEXT_SEGMENT, WINDOW_SEGMENT            ; 1806
                                        F0  12 00106         BNEQ    15$                                     ; 1808
                                 0C     AA  D4 00108         CLRL    12(BASE)                                ; 1809
          29           6A               1E  E5 0010B  16$:   BBCC    #30, (BASE), 17$                        ; 1815
                                        56  D5 0010F         TSTL    HEADER                                  ; 1816
                                        25  13 00111         BEQL    17$                                     ;
  42      A6     30    AA               06  28 00113         MOVC3   #6, 48(BASE), 66(HEADER)                ; 1819
                       50               66  9A 00119         MOVZBL  (HEADER), R0                            ; 1820
                       58             6640  3E 0011C         MOVAW   (HEADER)[R0], IDENT_AREA                ;
          68           6B               14  28 00120         MOVC3   #20, (R11), (IDENT_AREA)                ; 1822
  36      A8     14    AB               3C  28 00124         MOVC3   #60, 20(R11), 54(IDENT_AREA)            ; 1825
                                        56  DD 0012A         PUSHL   HEADER                                  ; 1826
            0000G      CF               01  FB 0012C         CALLS   #1, CHECKSUM                            ;
                                        56  DD 00131         PUSHL   HEADER                                  ; 1827
            0000G      CF               01  FB 00133         CALLS   #1, MAPK_DIRTY                          ;
          27           6A               15  E5 00138  17$:   BBCC    #21, (BASE), 19$                        ; 1834
                                        56  D5 0013C         TSTL    HEADER                                  ; 1835
                                        23  13 0013E         BEQL    19$                                     ;
                       50        00     BE  D0 00140         MOVL    @0(SP), R0                              ; 1838
                                        0D  13 00144         BEQL    18$                                     ;
                               00B0     C0  D5 00146         TSTL    176(R0)                                 ; 1840
                                        07  13 0014A         BEQL    18$                                     ;
                                        50  DD 0014C         PUSHL   R0                                      ; 1842
            0000G      CF               01  FB 0014E         CALLS   #1, KILL_DINDX                          ;
                       6A 00040002      8F  CA 00153  18$:   BICL2   #262146, (BASE)                         ; 1844
                                        56  DD 0015A         PUSHL   HEADER                                  ; 1846
                                        69  DD 0015C         PUSHL   (R9)                                    ;
            0000G      CF               02  FB 0015E         CALLS   #2, DELETE_FILE                         ;
          4F           6A               12  E5 00163  19$:   BBCC    #18, (BASE), 20$                        ; 1852
                                        56  D5 00167         TSTL    HEADER                                  ; 1853
```

```
                                  4B  13 00169          BEQL     20$                               :
                    50            69  D0 0016B          MOVL     (R9), R0                          : 1856
                    54       18   A0  D0 0016E          MOVL     24(R0), T1                        :
                    50            69  D0 00172          MOVL     (R9), R0                          : 1857
                    52       1C   A0  D0 00175          MOVL     28(R0), T2                        :
      50       04   AE            04  C1 00179          ADDL3    #4, 4(SP), R0                     : 1858
                    53            60  D0 0017E          MOVL     (R0), T3                          :
                    50            69  D0 00181          MOVL     (R9), R0                          : 1859
                             18   A0  D4 00184          CLRL     24(R0)                            :
                                  52  DD 00187          PUSHL    T2                                : 1860
                                  56  DD 00189          PUSHL    HEADER                            :
                                  69  DD 0018B          PUSHL    (R9)                              :
            0000G   CF            03  FB 0018D          CALLS    #3, TRUNCATE                      :
                    56       04   AA  D0 00192          MOVL     4(BASE), HEADER                   : 1861
                    50            69  D0 00196          MOVL     (R9), R0                          : 1862
               18   A0            54  D0 00199          MOVL     T1, 24(R0)                        :
                    50            69  D0 0019D          MOVL     (R9), R0                          : 1863
               1C   A0            52  D0 001A0          MOVL     T2, 28(R0)                        :
      50       04   AE            04  C1 001A4          ADDL3    #4, 4(SP), R0                     : 1864
                    60            53  D0 001A9          MOVL     T3, (R0)                          :
                    6A            10  8A 001AC          BICB2    #16, (BASE)                       : 1865
                                  56  DD 001AF          PUSHL    HEADER                            : 1866
            0000G   CF            01  FB 001B1          CALLS    #1, CHECKSUM                      :
      15            6A            01  E5 001B6 20$:     BBCC     #1, (BASE), 21$                   : 1873
                                  56  D5 001BA          TSTL     HEADER                            : 1874
                                  11  13 001BC          BEQL     21$                               :
                                  56  DD 001BE          PUSHL    HEADER                            : 1878
                             04   BE  DD 001C0          PUSHL    @4(SP)                            :
            0000G   CF            02  FB 001C3          CALLS    #2, REBLD_PRIM_FCB                :
                                  56  DD 001C8          PUSHL    HEADER                            : 1880
            0000G   CF            01  FB 001CA          CALLS    #1, BUILD_EXT_FCBS                :
      05            6A            1F  E5 001CF 21$:     BBCC     #31, (BASE), 22$                  : 1887
            0000G   CF            00  FB 001D3          CALLS    #0, REMAP_FILE                    :
                    58            6A  D0 001D8 22$:     MOVL     (BASE), DIR_FLAGS                 : 1894
                    6A  00C00020  8F  CA 001DB          BICL2    #12582944, (BASE)                 : 1897
      08            58            05  E0 001E2          BBS      #5, DIR_FLAGS, 23$                : 1899
      07            58            17  E0 001E6          BBS      #23, DIR_FLAGS, 23$               : 1900
      03            58            16  E0 001EA          BBS      #22, DIR_FLAGS, 23$               : 1901
                                009E  31 001EE          BRW      28$                               :
                    50            69  D0 001F1 23$:     MOVL     (R9), R0                          : 1904
                    7E       0E   A0  3C 001F4          MOVZWL   14(R0), -(SP)                     :
            0000G   CF            01  FB 001F8          CALLS    #1, SWITCH_VOLUME                 :
                             08   A7  D5 001FD          TSTL     8(R7)                             : 1910
                                  07  13 00200          BEQL     24$                               :
                                  57  DD 00202          PUSHL    R7                                : 1911
            0000G   CF            01  FB 00204          CALLS    #1, RESTORE_DIR                   :
      07            58            16  E1 00209 24$:     BBC      #22, DIR_FLAGS, 25$               : 1917
                                  7E  D4 0020D          CLRL     -(SP)                             : 1918
            0000G   CF            01  FB 0020F          CALLS    #1, REMOVE                        :
      52            58            17  E1 00214 25$:     BBC      #23, DIR_FLAGS, 27$               : 1927
10    00            6E            00  2C 00218          MOVC5    #0, (SP), #0, #16, NAME_DESC      : 1930
                             08   AE     0021D                                                     :
               0C   AE     0156   CA  9A 0021F          MOVZBL   342(BASE), NAME_DESC+4            : 1931
               10   AE     0157   CA  9E 00225          MOVAB    343(BASE), NAME_DESC+8            : 1932
               14   AE     0152   CA  B0 0022B          MOVW     338(BASE), NAME_DESC+12           : 1933
      1E            58            05  E1 00231          BBC      #5, DIR_FLAGS, 26$                : 1934
                             1C   A7  94 00235          CLRB     28(R7)                            : 1937
```

```
                    7E          01  CE 00238         MNEGL   #1, -(SP)                        ; 1938
                                7E  7C 0023B         CLRQ    -(SP)                            ;
                                7E  7C 0023D         CLRQ    -(SP)                            ;
                                7E  D4 0023F         CLRL    -(SP)                            ;
                           20   AE  9F 00241         PUSHAB  NAME_DESC                        ;
              0000G  CF         07  FB 00244         CALLS   #7, DIR_SCAN                     ;
                     50         69  D0 00249         MOVL    (R9), R0                         ; 1939
  04   A0     01FE   CA         06  28 0024C         MOVC3   #6, 510(BASE), 4(R0)             ;
                                69  DD 00253 26$:    PUSHL   (R9)                             ; 1941
                           0C   AE  9F 00255         PUSHAB  NAME_DESC                        ;
              0000G  CF         02  FB 00258         CALLS   #2, MAKE_ENTRY                   ;
                 02  AA    40   8F  8A 0025D         BICB2   #64, 2(BASE)                     ; 1942
                           04   A7  DD 00262         PUSHL   4(R7)                            ; 1943
              0000G  CF         01  FB 00265         CALLS   #1, WRITE_BLOCK                  ;
       21            58         05  E1 0026A 27$:    BBC     #5, DIR_FLAGS, 28$               ; 1952
       1D            58         17  E0 0026E         BBS     #23, DIR_FLAGS, 28$              ; 1953
       19            58         16  E0 00272         BBS     #22, DIR_FLAGS, 28$              ; 1954
                 0C  B7  0152   CA  B0 00276         MOVW    338(BASE), @12(R7)               ; 1957
                     50    0C   A7  D0 0027C         MOVL    12(R7), R0                       ; 1958
  02   A0     01FE   CA         06  28 00280         MOVC3   #6, 510(BASE), 2(R0)             ;
                           04   A7  DD 00287         PUSHL   4(R7)                            ; 1959
              0000G  CF         01  FB 0028A         CALLS   #1, MARK_DIRTY                   ;
                           36   AA  D5 0028F 28$:    TSTL    54(BASE)                         ; 1969
                                0B  13 00292         BEQL    29$                              ;
       6A       36   AA         36  28 00294         MOVC3   #54, 54(BASE), (BASE)            ; 1970
                           36   AA  D4 00299         CLRL    54(BASE)                         ; 1971
                              FD8A  31 0029C         BRW     1$                               ; 1667
                     50         01  D0 0029F 29$:    MOVL    #1, R0                           ; 1975
                                    04 002A2         RET                                      ; 1977

; Routine Size:  675 bytes,    Routine Base:  $CODE$ + 0194
```

```
;  986          1978  1 ROUTINE FLUSH_FIDCACHE : L_NORM =
;  987          1979  1
;  988          1980  1 !++
;  989          1981  1 !
;  990          1982  1 ! FUNCTIONAL DESCRIPTION:
;  991          1983  1 !
;  992          1984  1 !       This routine empties the file ID cache by zeroing the entry count.
;  993          1985  1 !       It must be called in kernel mode.
;  994          1986  1 !
;  995          1987  1 !
;  996          1988  1 ! CALLING SEQUENCE:
;  997          1989  1 !       FLUSH_FIDCACHE ()
;  998          1990  1 !
;  999          1991  1 ! INPUT PARAMETERS:
; 1000          1992  1 !       NONE
; 1001          1993  1 !
; 1002          1994  1 ! IMPLICIT INPUTS:
; 1003          1995  1 !       CURRENT_VCB: VCB of volume
; 1004          1996  1 !
; 1005          1997  1 ! OUTPUT PARAMETERS:
; 1006          1998  1 !       NONE
; 1007          1999  1 !
; 1008          2000  1 ! IMPLICIT OUTPUTS:
; 1009          2001  1 !       NONE
; 1010          2002  1 !
; 1011          2003  1 ! ROUTINE VALUE:
; 1012          2004  1 !       1
; 1013          2005  1 !
; 1014          2006  1 ! SIDE EFFECTS:
; 1015          2007  1 !       file ID cache cleared
; 1016          2008  1 !
; 1017          2009  1 !--
; 1018          2010  1
; 1019          2011  2 BEGIN
; 1020          2012  2
; 1021          2013  2 BIND_COMMON;
; 1022          2014  2
; 1023          2015  2 LOCAL
; 1024          2016  2         FID_CACHE       : REF BBLOCK;   ! file ID cache
; 1025          2017  2
; 1026          2018  2
; 1027          2019  2 FID_CACHE = .BBLOCK [.CURRENT_VCB[VCB$L_CACHE], VCA$L_FIDCACHE];
; 1028          2020  2 FID_CACHE[VCA$W_FIDCOUNT] = 0;
; 1029          2021  2
; 1030          2022  2 1
; 1031          2023  1 END;                                    ! end of routine FLUSH_FIDCACHE


                                0000 00000 FLUSH_FIDCACHE:
                                                .WORD   Save nothing                    ; 1978
                    50    98    AA   D0 00002         MOVL    -104(BASE), R0          ; 2019
                    50    58    B0   D0 00006         MOVL    @88(R0), FID_CACHE      ;
                          02    A0   B4 0000A         CLRW    2(FID_CACHE)            ; 2020
                    50          01   D0 0000D         MOVL    #1, R0                  ; 2023
```

```
                                04 00010        RET                                             ;

; Routine Size:  17 bytes,    Routine Base:  $CODE$ + 0437
```

```
;  1033          2024  1 ROUTINE MAKE_DEACCESS : L_NORM =
;  1034          2025  1
;  1035          2026  1 !++
;  1036          2027  1 !
;  1037          2028  1 ! FUNCTIONAL DESCRIPTION:
;  1038          2029  1 !
;  1039          2030  1 !       This routine performs the machinery for deaccessing a file.
;  1040          2031  1 !
;  1041          2032  1 ! CALLING SEQUENCE:
;  1042          2033  1 !       MAKE_DEACCESS ()
;  1043          2034  1 !
;  1044          2035  1 ! INPUT PARAMETERS:
;  1045          2036  1 !       NONE
;  1046          2037  1 !
;  1047          2038  1 ! IMPLICIT INPUTS:
;  1048          2039  1 !       PRIMARY_FCB: FCB of file
;  1049          2040  1 !       CURRENT_WINDOW: window of file
;  1050          2041  1 !       CURRENT_VCB: VCB of volume in process
;  1051          2042  1 !
;  1052          2043  1 ! OUTPUT PARAMETERS:
;  1053          2044  1 !       NONE
;  1054          2045  1 !
;  1055          2046  1 ! IMPLICIT OUTPUTS:
;  1056          2047  1 !       NONE
;  1057          2048  1 !
;  1058          2049  1 ! ROUTINE VALUE:
;  1059          2050  1 !       NONE
;  1060          2051  1 !
;  1061          2052  1 ! SIDE EFFECTS:
;  1062          2053  1 !       file deaccessed
;  1063          2054  1 !
;  1064          2055  1 !--
;  1065          2056  1
;  1066          2057  2 BEGIN
;  1067          2058  2
;  1068          2059  2 BIND_COMMON;
;  1069          2060  2
;  1070          2061  2 LOCAL
;  1071          2062  2         FCB             : REF BBLOCK,   ! local for primary fcb.
;  1072          2063  2         LCKMODE,                        ! lock mode for access lock.
;  1073          2064  2         WINDOW_SEGMENT  : REF BBLOCK,   ! address of the next window segment
;  1074          2065  2         DUMMY;                          ! dummy local to receive REMQUE
;  1075          2066  2
;  1076          2067  2 EXTERNAL
;  1077          2068  2         PMS$GL_OPEN     : ADDRESSING_MODE (ABSOLUTE);
;  1078          2069  2                                         ! system count of currently open files
;  1079          2070  2
;  1080          2071  2 EXTERNAL ROUTINE
;  1081          2072  2         DEQ_LOCK        : L_NORM,       ! dequeue a lock
;  1082          2073  2         CONV_ACCLOCK    : L_NORM,       ! Convert file access lock.
;  1083          2074  2         LOCK_MODE       : L_JSB_1ARG;   ! Calculate access lock mode.
;  1084          2075  2
;  1085          2076  2 FCB = .PRIMARY_FCB;
;  1086          2077  2
;  1087          2078  2 ! Unlink the window from the FCB. Clear the applicable access conditions
;  1088          2079  2 ! in the FCB.
;  1089          2080  2 !
```

```
;  1090          2081  2
;  1091          2082  2 WINDOW_SEGMENT = .CURRENT_WINDOW;
;  1092          2083  2 DO
;  1093          2084  3     BEGIN
;  1094          2085  3     IF .WINDOW_SEGMENT[WCB$L_WLFL] NEQ 0 THEN REMQUE (.WINDOW_SEGMENT, DUMMY);
;  1095          2086  3     WINDOW_SEGMENT = .WINDOW_SEGMENT[WCB$L_LINK];
;  1096          2087  3     END
;  1097          2088  2 UNTIL .WINDOW_SEGMENT EQL 0;
;  1098          2089  2
;  1099          2090  2 IF NOT .CURRENT_WINDOW [WCB$V_NOACCLOCK]
;  1100          2091  2 THEN
;  1101          2092  3     BEGIN
;  1102          2093  3     IF .CURRENT_WINDOW[WCB$V_NOREAD]
;  1103          2094  3     THEN FCB[FCB$V_EXCL] = 0;
;  1104          2095  3
;  1105          2096  3     IF .CURRENT_WINDOW[WCB$V_NOTRUNC]
;  1106          2097  3     THEN FCB[FCB$W_TCNT] = .FCB[FCB$W_TCNT] - 1;
;  1107          2098  3
;  1108          2099  3     IF .CURRENT_WINDOW[WCB$V_NOWRITE]
;  1109          2100  3     THEN FCB[FCB$W_LCNT] = .FCB[FCB$W_LCNT] - 1;
;  1110          2101  3
;  1111          2102  3     FCB [FCB$W_ACNT] = .FCB [FCB$W_ACNT] - 1;
;  1112          2103  3
;  1113          2104  2     END;                          ! of normal (not NOLOCK) deaccess.
;  1114          2105  2
;  1115          2106  2 FCB[FCB$W_REFCNT] = .FCB[FCB$W_REFCNT] - 1;
;  1116          2107  2
;  1117          2108  2 ! For a write access, bump down the writer count. If this is the
;  1118          2109  2 ! last write, and the file is the index file or the storage map, clear
;  1119          2110  2 ! the appropriate flag in the VCB. If there's a cache lock being held
;  1120          2111  2 ! for this file, release it.
;  1121          2112  2 !
;  1122          2113  2
;  1123          2114  2 IF .CURRENT_WINDOW[WCB$V_WRITE]
;  1124          2115  2 THEN
;  1125          2116  3     BEGIN
;  1126          2117  3
;  1127          2118  3     IF NOT .CURRENT_WINDOW [WCB$V_NOACCLOCK]
;  1128          2119  3     THEN
;  1129          2120  3         FCB[FCB$W_WCNT] = .FCB[FCB$W_WCNT] - 1;
;  1130          2121  3
;  1131          2122  3     IF .FCB[FCB$W_WCNT] EQL 0
;001  !ACG0468    2123  4     OR (.FCB[FCB$W_WCNT] LEQU 1 AND .FCB EQL .CURRENT_VCB[VCB$L_QUOTAFCB])
;002  !ACG0468    2124  4     OR (.FCB [FCB$W_REFCNT] EQL 0 AND .CURRENT_WINDOW [WCB$V_WRITE])
;  1133-1        2125  3     THEN
;  1134          2126  4         BEGIN
;  1135          2127  4         IF .FCB[FCB$B_FID_NMX] EQL 0
;  1136          2128  4         THEN
;  1137          2129  5             BEGIN
;  1138          2130  5             IF .FCB[FCB$W_FID_NUM] EQL 1
;  1139          2131  5             THEN CURRENT_VCB[VCB$V_WRITE_IF] = 0;
;  1140          2132  5             IF .FCB[FCB$W_FID_NUM] EQL 2
;  1141          2133  5             THEN CURRENT_VCB[VCB$V_WRITE_SM] = 0;
;  1142          2134  4             END;
;  1143          2135  4         IF .FCB[FCB$L_CACHELKID] NEQ 0
;  1144          2136  4         THEN
;  1145          2137  5             BEGIN
```

```
;  1146     2138  5              DEQ_LOCK (.FCB[FCB$L_CACHELKID]);
;  1147     2139  5              FCB[FCB$L_CACHELKID] = 0;
;  1148     2140  4              END;
;  1149     2141  3          END;
;  1150     2142  2      END;
;  1151     2143  2
;  1152     2144  2  ! Recalculate the lock mode of the access lock for this fcb.
;  1153     2145  2  !
;  1154     2146  2
;  1155     2147  2  IF .FCB [FCB$W_ACNT] EQL 0
;  1156     2148  2  THEN
;  1157     2149  2      LCKMODE = LCK$K_NLMODE
;  1158     2150  2  ELSE
;  1159     2151  3      BEGIN
;  1160     2152  3      LOCAL
;  1161     2153  3          ACCTL;
;  1162     2154  3
;  1163     2155  3      ACCTL = 0;
;  1164     2156  3      IF .FCB [FCB$W_WCNT] NEQ 0
;  1165     2157  3      THEN ACCTL = .ACCTL + FIB$M_WRITE;
;  1166     2158  3      IF .FCB [FCB$W_LCNT] NEQ 0
;  1167     2159  3      THEN ACCTL = .ACCTL + FI3$M_NOWRITE;
;  1168     2160  3
;  1169     2161  3      LCKMODE = LOCK_MODE (.ACCTL);
;  1170     2162  3
;  1171     2163  2      END;
;  1172     2164  2
;  1173     2165  2  ! If the new access lock mode lock for this fcb is different (lower)
;  1174     2166  2  ! than the curren' lock, convert it.  The conversion routine will also
;  1175     2167  2  ! dequeue the lock if this is the last reference.
;  1176     2168  2  !
;  1177     2169  2
;  1178     2170  2  IF .LCKMODE<0,8> NEQ .FCB [FCB$B_ACCLKMODE]
;  1179     2171  2      OR .FCB [FCB$W_REFCNT] EQL 0
;  1180     2172  2  THEN
;  1181     2173  2      IF NOT CONV_ACCLOCK (.LCKMODE, .FCB)
;  1182     2174  2      THEN
;  1183     2175  2          BUG_CHECK (XQPERR, 'deaccess conversion failed');
;  1184     2176  2
;  1185     2177  2  ! Note: We now have a file control block with a possible zero access count
;  1186     2178  2  ! in the FCB list. This gets dealt with by the general cleanup.
;  1187     2179  2  !
;  1188     2180  2
;  1189     2181  2  PMS$GL_OPEN = .PMS$GL_OPEN - 1;          ! bump down count of open files
;  1190     2182  2  CURRENT_VCB[VCB$W_TRANS] = .CURRENT_VCB[VCB$W_TRANS] - 1;
;  1191     2183  2
;  1192     2184  2  RETURN 1;
;  1193     2185  2
;  1194     2186  1  END;                                    ! end of routine MAKE_DEACCESS


                                                            .EXTRN  PMS$GL_OPEN, DEQ_LOCK
                                                            .EXTRN  CONV_ACCLOCK, LOCK_MODE
                                                            .EXTRN  BUG$_XQPERR

                                                 000C 00000 MAKE_DEACCESS:
```

```
                                                   .WORD   Save R2,R3                          ; 2024
           51   0C   AA  9E  00002          MOVAB   12(BASE), R1                        ; 2057
           52   08   AA  D0  00006          MOVL    8(BASE), FCB                        ; 2076
           50        61  D0  0000A          MOVL    (R1), WINDOW_SEGMENT                ; 2082
                     60  D5  0000D  1$:     TSTL    (WINDOW_SEGMENT)                    ; 2085
                     03  13  0000F          BEQL    2$                                  ;
           53        60  0F  00011          REMQUE  (WINDOW_SEGMENT), DUMMY             ;
           50   20   A0  D0  00014  2$:     MOVL    32(WINDOW_SEGMENT), WINDOW_SEGMENT  ; 2086
                     F3  12  00018          BNEQ    1$                                  ; 2088
           50        61  D0  0001A          MOVL    (R1), R0                            ; 2090
 21   14   A0        02  E0  0001C          BBS     #2, 20(R0), 6$                      ;
 04   15   A0        02  E1  00022          BBC     #2, 21(R0), 3$                      ; 2093
      22   A2        08  8A  00027          BICB2   #8, 34(FCB)                         ; 2094
           50        61  D0  0002B  3$:     MOVL    (R1), R0                            ; 2096
 03   15   A0        03  E1  0002E          BBC     #3, 21(R0), 4$                      ;
                20   A2  B7  00033          DECW    32(FCB)                             ; 2097
           50        61  D0  00036  4$:     MOVL    (R1), R0                            ; 2099
           03   14   A0  E9  00039          BLBC    20(R0), 5$                          ;
                1E   A2  B7  0003D          DECW    30(FCB)                             ; 2100
                1A   A2  B7  00040  5$:     DECW    26(FCB)                             ; 2102
                18   A2  B7  00043  6$:     DECW    24(FCB)                             ; 2106
           50        61  D0  00046          MOVL    (R1), R0                            ; 2114
 5B   0B   A0        01  E1  00049          BBC     #1, 11(R0), 12$                     ;
 03   14   A0        02  E0  0004E          BBS     #2, 20(R0), 7$                      ; 2118
                1C   A2  B7  00053          DECW    28(FCB)                             ; 2120
                1C   A2  B5  00056  7$:     TSTW    28(FCB)                             ; 2122
                     1D  13  00059          BEQL    9$                                  ;
           01   1C   A2  B1  0005B          CMPW    28(FCB), #1                         ; 2123
                     0A  1A  0005F          BGTRU   8$                                  ;
           50   98   AA  D0  00061          MOVL    -104(BASE), R0                      ;
      54   A0        52  D1  00065          CMPL    FCB, 84(R0)                         ;
                     0D  13  00069          BEQL    9$                                  ;
                18   A2  B5  0006B  8$:     TSTW    24(FCB)                             ; 2124
                     39  12  0006E          BNEQ    12$                                 ;
           50        61  D0  00070          MOVL    (R1), R0                            ;
 31   0B   A0        01  E1  00073          BBC     #1, 11(R0), 12$                     ;
                29   A2  95  00078  9$:     TSTB    41(FCB)                             ; 2127
                     1C  12  0007B          BNEQ    11$                                 ;
           01   24   A2  B1  0007D          CMPW    36(FCB), #1                         ; 2130
                     08  12  00081          BNEQ    10$                                 ;
           50   98   AA  D0  00083          MOVL    -104(BASE), R0                      ; 2131
      0B   A0        01  8A  00087          BICB2   #1, 11(R0)                          ;
           02   24   A2  B1  0008B  10$:    CMPW    36(FCB), #2                         ; 2132
                     08  12  0008F          BNEQ    11$                                 ;
           50   98   AA  D0  00091          MOVL    -104(BASE), R0                      ; 2133
      0B   A0        02  8A  00095          BICB2   #2, 11(R0)                          ;
                54   A2  D5  00099  11$:    TSTL    84(FCB)                             ; 2135
                     0B  13  0009C          BEQL    12$                                 ;
                54   A2  DD  0009E          PUSHL   84(FCB)                             ; 2138
   0000G   CF        01  FB  000A1          CALLS   #1, DEQ_LOCK                        ;
                54   A2  D4  000A6          CLRL    84(FCB)                             ; 2139
                1A   A2  B5  000A9  12$:    TSTW    26(FCB)                             ; 2147
                     04  12  000AC          BNEQ    13$                                 ;
                     51  D4  000AE          CLRL    LCKMODE                             ; 2149
                     19  11  000B0          BRB     16$                                 ;
                     50  D4  000B2  13$:    CLRL    ACCTL                               ; 2155
                1C   A2  B5  000B4          TSTW    28(FCB)                             ; 2156
```

```
                        05  13 000B7          BEQL    14$                          ;
           50     0100  C0  9E 000B9          MOVAB   256(R0), ACCTL               ; 2157
                    1E  A2  B5 000BE  14$:    TSTW    30(FCB)                      ; 2158
                        02  13 000C1          BEQL    15$                          ;
                        50  D6 000C3          INCL    ACCTL                        ; 2159
                     0000G  30 000C5  15$:    BSBW    LOCK_MODE                    ; 2161
           51           50  D0 000C8          MOVL    R0, LCKMODE                  ;
      0B   A2           51  91 000CB  16$:    CMPB    LCKMODE, 11(FCB)             ; 2170
                        05  12 000CF          BNEQ    17$                          ;
                    18  A2  B5 000D1          TSTW    24(FCB)                      ; 2171
                        0E  12 000D4          BNEQ    18$                          ;
                        06  BB 000D6  17$:    PUSHR   #^M<R1,R2>                   ; 2173
   0000G   CF           02  FB 000D8          CALLS   #2, CONV_ACCLOCK             ;
           04           50  E8 000DD          BLBS    R0, 18$                      ;
                          FEFF 000E0          BUGW                                 ; 2175
                         0000* 000E2          .WORD   <BUG$_XQPERR!4>              ;
              00000000G 9F  D7 000E4  18$:    DECL    @#PMS$GL_OPEN                ; 2181
           50       98  AA  D0 000EA          MOVL    -104(BASE), R0               ; 2182
                    0C  A0  B7 000EE          DECW    12(R0)                       ;
           50           01  D0 000F1          MOVL    #1, R0                       ; 2184
                            04 000F4          RET                                  ; 2186

; Routine Size:  245 bytes,    Routine Base:  $CODE$ + 0448
```

```
;  1196          2187  1 GLOBAL ROUTINE DEL_EXTFCB (START_FCB) : L_NORM =
;  1197          2188  1
;  1198          2189  1 !++
;  1199          2190  1 !
;  1200          2191  1 ! FUNCTIONAL DESCRIPTION:
;  1201          2192  1 !
;  1202          2193  1 !       This routine removes and deallocates all extension FCB's, if any,
;  1203          2194  1 !       linked to the indicated FCB.
;  1204          2195  1 !
;  1205          2196  1 ! CALLING SEQUENCE:
;  1206          2197  1 !       DEL_EXTFCB (ARG1)
;  1207          2198  1 !
;  1208          2199  1 ! INPUT PARAMETERS:
;  1209          2200  1 !       ARG1: address of primary FCB or 0
;  1210          2201  1 !
;  1211          2202  1 ! IMPLICIT INPUTS:
;  1212          2203  1 !       NONE
;  1213          2204  1 !
;  1214          2205  1 ! OUTPUT PARAMETERS:
;  1215          2206  1 !       NONE
;  1216          2207  1 !
;  1217          2208  1 ! IMPLICIT OUTPUTS:
;  1218          2209  1 !       NONE
;  1219          2210  1 !
;  1220          2211  1 ! ROUTINE VALUE:
;  1221          2212  1 !       NONE
;  1222          2213  1 !
;  1223          2214  1 ! SIDE EFFECTS:
;  1224          2215  1 !       FCB's deallocated
;  1225          2216  1 !
;  1226          2217  1 !--
;  1227          2218  1
;  1228          2219  2 BEGIN
;  1229          2220  2
;  1230          2221  2 MAP
;  1231          2222  2         START_FCB       : REF BBLOCK;   ! FCB argument
;  1232          2223  2
;  1233          2224  2 LOCAL
;  1234          2225  2         FCB             : REF BBLOCK,   ! running FCB pointer
;  1235          2226  2         NEXT_FCB        : REF BBLOCK,   ! next extension FCB
;  1236          2227  2         P               : REF BBLOCK,   ! pointer to chase for VCB
;  1237          2228  2         DUMMY;                          ! dummy local to receive REMQUE
;  1238          2229  2
;  1239          2230  2 BASE_REGISTER;
;  1240          2231  2
;  1241          2232  2 EXTERNAL ROUTINE
;  1242          2233  2         DEALLOCATE      : L_NORM;       ! deallocate dynamic memory
;  1243          2234  2
;  1244          2235  2 ! Checking for null pointers, find the first extension FCB. Follow the extension
;  1245          2236  2 ! list and remove and deallocate the extension FCB's, cleaning out the pointers
;  1246          2237  2 ! on the way. For each FCB removed, we must find the VCB (by chasing around the
;  1247          2238  2 ! FCB list) and decrement the transaction count.
;  1248          2239  2 !
;  1249          2240  2
;  1250          2241  2 IF .START_FCB EQL 0 THEN RETURN 1;
;  1251          2242  2 FCB = .START_FCB[FCB$L_EXFCB];
;  1252          2243  2 START_FCB[FCB$L_EXFCB] = 0;
```

```
;  1253          2244  2 UNTIL .FCB EQL 0 DO
;  1254          2245  3     BEGIN
;  1255          2246  3     NEXT_FCB = .FCB[FCB$L_EXFCB];
;  1256          2247  3
;  1257          2248  3     P = .FCB[FCB$L_FCBFL];
;  1258          2249  3     UNTIL .P[VCB$B_TYPE] EQL DYN$C_VCB
;  1259          2250  3     DO P = .P[FCB$L_FCBFL];
;  1260          2251  3     P[VCB$W_TRANS] = .P[VCB$W_TRANS] - 1;
;  1261          2252  3
;  1262          2253  3     FCB[FCB$L_EXFCB] = 0;
;  1263          2254  3     IF .FCB [FCB$B_TYPE] NEQ DYN$C_FCB
;  1264          2255  3     THEN
;  1265          2256  3         BUG_CHECK (NOTFCBFCB, 'not fcb');
;  1266          2257  3     REMQUE (.FCB, DUMMY);
;  1267          2258  3     DEALLOCATE (.FCB);
;  1268          2259  3     FCB = .NEXT_FCB;
;  1269          2260  2     END;
;  1270          2261  2
;  1271          2262  2 RETURN 1;
;  1272          2263  2
;  1273          2264  1 END;                                    ! end of routine DEL_EXTFCB


                                                        .EXTRN  BUG$_NOTFCBFCB

                                        003C 00000      .ENTRY  DEL_EXTFCB, Save R2,R3,R4,R5            ; 2187
                              50    04  AC  D0 00002      MOVL    START_FCB, R0                           ; 2241
                                        3C  13 00006      BEQL    5$                                      ;
                              53    0C  A0  D0 00008      MOVL    12(R0), FCB                             ; 2242
                                    0C  A0  D4 0000C      CLRL    12(R0)                                  ; 2243
                                        53  D5 0000F 1$:  TSTL    FCB                                     ; 2244
                                        31  13 00011      BEQL    5$                                      ;
                              54    0C  A3  D0 00013      MOVL    12(FCB), NEXT_FCB                       ; 2246
                              52        63  D0 00017      MOVL    (FCB), P                                ; 2248
                              11    0A  A2  91 0001A 2$:  CMPB    10(P), #17                              ; 2249
                                        05  13 0001E      BEQL    3$                                      ;
                              52        62  D0 00020      MOVL    (P), P                                  ; 2250
                                        F5  11 00023      BRB     2$                                      ;
                                    0C  A2  B7 00025 3$:  DECW    12(P)                                   ; 2251
                                    0C  A3  D4 00028      CLRL    12(FCB)                                 ; 2253
                              07    0A  A3  91 0002B      CMPB    10(FCB), #7                             ; 2254
                                        04  13 0002F      BEQL    4$                                      ;
                                      FEFF  00031      BUGW                                            ; 2256
                                     0000*  00033      .WORD   <BUG$_NOTFCBFCB!4>                       ;
                              55        63  0F 00035 4$:  REMQUE  (FCB), DUMMY                          ; 2257
                                        53  DD 00038      PUSHL   FCB                                     ; 2258
                        0000G CF        01  FB 0003A      CALLS   #1, DEALLOCATE                          ;
                              53        54  D0 0003F      MOVL    NEXT_FCB, FCB                           ; 2259
                                        CB  11 00042      BRB     1$                                      ; 2244
                              50        01  D0 00044 5$:  MOVL    #1, R0                                  ; 2262
                                            04 00047      RET                                             ; 2264

; Routine Size:  72 bytes,    Routine Base:  $CODE$ + 053D
```

```
;1275     2265  1 ROUTINE ZERO_CHANNEL : L_NORM =
;1276     2266  1
;1277     2267  1 !++
;1278     2268  1 !
;1279     2269  1 ! FUNCTIONAL DESCRIPTION:
;1280     2270  1 !
;1281     2271  1 !       This routine zeroes out the window pointer being returned to
;1282     2272  1 !       the user for his channel control block. It also credits one to the
;1283     2273  1 !       user's open file quota, except for the case of a shared window.
;1284     2274  1 !       This routine must be executed in kernel mode.
;1285     2275  1 !
;1286     2276  1 ! CALLING SEQUENCE:
;1287     2277  1 !       ZERO_CHANNEL ()
;1288     2278  1 !
;1289     2279  1 ! INPUT PARAMETERS:
;1290     2280  1 !       NONE
;1291     2281  1 !
;1292     2282  1 ! IMPLICIT INPUTS:
;1293     2283  1 !       IO_PACKET: I/O packet of request
;1294     2284  1 !
;1295     2285  1 ! OUTPUT PARAMETERS:
;1296     2286  1 !       NONE
;1297     2287  1 !
;1298     2288  1 ! IMPLICIT OUTPUTS:
;1299     2289  1 !       NONE
;1300     2290  1 !
;1301     2291  1 ! ROUTINE VALUE:
;1302     2292  1 !       NONE
;1303     2293  1 !
;1304     2294  1 ! SIDE EFFECTS:
;1305     2295  1 !       channel window pointer cleared, file quota bumped unless shared window
;1306     2296  1 !
;1307     2297  1 !--
;1308     2298  1
;1309     2299  2 BEGIN
;1310     2300  2
;1311     2301  2 LOCAL
;1312     2302  2         ABD             : REF BBLOCKVECTOR [,ABD$C_LENGTH],
;1313     2303  2                                           ! buffer descriptors
;1314     2304  2         JIB             : REF BBLOCK,     ! Job information block address
;1315     2305  2         PCB             : REF BBLOCK;     ! address of user process control block
;1316     2306  2
;1317     2307  2 EXTERNAL
;1318     2308  2         SCH$GL_PCBVEC   : REF VECTOR ADDRESSING_MODE (ABSOLUTE);
;1319     2309  2                                           ! system PCB vector
;1320     2310  2
;1321     2311  2 BIND_COMMON;
;1322     2312  2
;1323     2313  2                                           ! pointer to buffer descriptors
;1324     2314  2 ABD = .BBLOCK [.IO_PACKET[IRP$L_SVAPTE], AIB$L_DESCRIPT];
;1325     2315  2 ABD[ABD$C_WINDOW, ABD$W_COUNT] = 4;
;1326     2316  2 .ABD[ABD$C_WINDOW, ABD$W_TEXT] + ABD[ABD$C_WINDOW, ABD$W_TEXT] + 1 = 0;
;1327     2317  2
;1328     2318  2 IF
;1329     2319  3     BEGIN
;1330     2320  3
;1331     2321  3     ! The FILCNT quota is credited if a WCB has not yet been allocated or
```

```
;  1332          2322  3      ! if the SHRWCB bit is not set in the WCB.
;  1333          2323  3
;  1334          2324  3      IF .CURRENT_WINDOW EQL 0
;  1335          2325  3      THEN 1
;  1336          2326  3      ELSE NOT .CURRENT_WINDOW[WCB$V_SHRWCB]
;  1337          2327  3      END
;  1338          2328  2  THEN
;  1339          2329  3      BEGIN
;  1340          2330  3      PCB = .SCH$GL_PCBVEC[.(IO_PACKET[IRP$L_PID])<0,16>];
;  1341          2331  3      JIB = .PCB[PCB$L_JIB];
;  1342          2332  3      JIB[JIB$W_FILCNT] = .JIB[JIB$W_FILCNT] + 1;
;  1343          2333  2      END;
;  1344          2334  2
;  1345          2335  2  RETURN 1;
;  1346          2336  2
;  1347          2337  1  END;                                   ! end of routine ZERO_CHANNEL


                                                        .EXTRN  SCH$GL_PCBVEC

                                0000 00000 ZERO_CHANNEL:
                                                        .WORD   Save nothing                    ;  2265
                     50      90   AA   D0 00002        MOVL    -112(BASE), R0                  ;  2314
                     51      2C   B0   D0 00006        MOVL    @44(R0), ABD
                  02 A1           04   B0 0000A        MOVW    #4, 2(ABD)                      ;  2315
                     50           61   3C 0000E        MOVZWL  (ABD), R0                       ;  2316
                              01 A140   9F 00011        PUSHAB  1(ABD)[R0]
                                 9E     D4 00015        CLRL    @(SP)+
                        50       0C    AA   D0 00017    MOVL    12(BASE), R0                    ;  2324
                                 05    13   00018      BEQL    1$
                  1D       0B    A0          03   E0 0001D    BBS     #3, 11(R0), 2$          ;  2326
                     51 00000000G   CF   D0 00022 1$:  MOVL    @#SCH$GL_PCBVEC, R1             ;  2330
                     50       90    A1   D0 00029    MOVL    -112(BASE), R0
                     50             0?   C0 0002D    ADDL2   #12, R0
                     50             60   3C 00030    MOVZWL  (R0), R0
                     50           6140   D0 00033    MOVL    (R1)[R0], PCB
                     50     0080    C0   D0 00037    MOVL    128(PCB), JIB                   ;  2331
                              30    A0   B6 0003C    INCW    48(JIB)                         ;  2332
                     50             01   D0 0003F 2$:  MOVL    #1, R0                      ;  2335
                                          04 00042    RET                                     ;  2337

; Routine Size:  67 bytes,    Routine Base:  $CODE$ + 0585
```

```
;  1349          2338  1 GLOBAL ROUTINE NUKE_HEAD_FCB (FCB) : L_NORM NOVALUE =
;  1350          2339  1
;  1351          2340  1 !++
;  1352          2341  1 !
;  1353          2342  1 ! Functional Description:
;  1354          2343  1 !
;  1355          2344  1 ! Given an fcb already stripped of possible extension fcbs,
;  1356          2345  1 ! and which has a refcnt of 0 (assumed), clean up the things
;  1357          2346  1 ! that need cleaning up, remove it from the fcb list (we assume
;  1358          2347  1 ! that is where it is), and deallocate it.
;  1359          2348  1 !
;  1360          2349  1 !--
;  1361          2350  1
;  1362          2351  2 BEGIN
;  1363          2352  2
;  1364          2353  2 MAP
;  1365          2354  2         FCB     : REF BBLOCK;
;  1366          2355  2
;  1367          2356  2 BASE_REGISTER;
;  1368          2357  2
;  1369          2358  2 EXTERNAL ROUTINE
;  1370          2359  2         ACL_DELETEACL,
;  1371          2360  2         CONV_ACCLOCK    : L_NORM,
;  1372          2361  2         DEALLOCATE      : L_NORM;
;  1373          2362  2
;  1374          2363  2 LOCAL
;  1375          2364  2         DUMMY;
;  1376          2365  2
;  1377          2366  2 IF .FCB [FCB$B_TYPE] NEQ DYN$C_FCB
;  1378          2367  2 THEN
;  1379          2368  2     BUG_CHECK (NOTFCBFCB, 'not fcb');
;  1380          2369  2
;  1381          2370  2 REMQUE (.FCB, DUMMY);
;  1382          2371  2
;  1383          2372  2 IF .BBLOCK [FCB [FCB$R_ORB], ORB$V_ACL_QUEUE]
;  1384          2373  2 THEN
;  1385          2374  2     ACL_DELETEACL (FCB [FCB$L_ACLFL], 0);
;  1386          2375  2
;  1387          2376  2 IF NOT CONV_ACCLOCK (0, .FCB)
;  1388          2377  2 THEN
;  1389          2378  2     BUG_CHECK (XQPERR, 'Unexpected lock manager status');
;  1390          2379  2
;  1391          2380  2 DEALLOCATE (.FCB);
;  1392          2381  2
;  1393          2382  1 END;                 ! of routine NUKE_HEAD_FCB


                                                        .EXTRN  ACL_DELETEACL

                                  0000 00000            .ENTRY  NUKE_HEAD_FCB, Save nothing             ; 2338
                50        04    AC  D0 00002            MOVL    FCB, R0                                 ; 2366
                07        0A    A0  91 00006            CMPB    10(R0), #7
                                  04  13 0000A            BEQL    1$
                                    FEFF 0000C            BUGW                                            ; 2368
                                   0000* 0000E            .WORD   <BUG$_NOTFCBFCB!4>
                50        04    BC  0F 00010 1$:        REMQUE  @FCB, DUMMY                             ; 2370
```

```
                50        04     AC  D0 00014          MOVL    FCB, R0                          ; 2372
          10      63      A0            01  E1 00018    BBC     #1, 99(R0), 2$
                                        7E  D4 0001D    CLRL    -(SP)                            ; 2374
          7E      04    AC 00000080     8F  C1 0001F    ADDL3   #128, FCB, -(SP)
                0000G   CF              02  FB 00028    CALLS   #2, ACL_DELETEACL
                                  04    AC  DD 0002D 2$:  PUSHL   FCB                            ; 2376
                                        7E  D4 00030    CLRL    -(SP)
                0000G   CF              02  FB 00032    CALLS   #2, CONV_ACCLOCK
                        04              50  E8 00037    BLBS    R0, 3$
                                          FEFF 0003A    BUGW                                     ; 2378
                                         0000* 0003C    .WORD   <BUG$_XQPERR'4>
                                  04    AC  DD 0003E 3$:  PUSHL   FCB                            ; 2380
                0000G   CF              01  FB 00041    CALLS   #1, DEALLOCATE
                                        04  00046      RET                                      ; 2382

; Routine Size:  71 bytes,    Routine Base:  $CODE$ + 05CB
```

```
;  1395          2383  1 LOCK_CODE;
;  1396          2384  1 GLOBAL ROUTINE SET_DIRINDX (FCB) : L_JSB_1ARG =
;  1397          2385  1
;  1398          2386  1 !++
;  1399          2387  1 !
;  1400          2388  1 ! Functional Description:
;  1401          2389  1 !
;  1402          2390  1 ! This routine tests for the presence of a directory index, and
;  1403          2391  1 ! set the FCB$V_DIR flag accordingly at SCHED ipl, so at to
;  1404          2392  1 ! interlock with the directory index handling routine which
;  1405          2393  1 ! may be trying to toss it out, and the search_fcb routine,
;  1406          2394  1 ! which also runs at sched ipl.
;  1407          2395  1 !
;  1408          2396  1 ! ROUTINE VALUE:
;  1409          2397  1 !       true - if this now a directory fcb eligible for replacement
;  1410          2398  1 !       false - otherwise
;  1411          2399  1 !
;  1412          2400  1 !--
;  1413          2401  1
;  1414          2402  2 BEGIN
;  1415          2403  2
;  1416          2404  2 MAP
;  1417          2405  2         FCB     : REF BBLOCK;
;  1418          2406  2
;  1419          2407  2 LOCAL
;  1420          2408  2         STATUS  : INITIAL (0);
;  1421          2409  2
;  1422          2410  2 SET_IPL (IPL$_SCHED);
;  1423          2411  2
;  1424          2412  2 IF .FCB [FCB$L_DIRINDX] NEQ 0
;  1425          2413  2 THEN
;  1426          2414  3     BEGIN
;  1427          2415  3     FCB [FCB$V_DIR] = 1;
;  1428          2416  3     STATUS = .STATUS + 1;
;  1429          2417  2     END;
;  1430          2418  2
;  1431          2419  2 SET_IPL (0);
;  1432          2420  2
;  1433          2421  2 .STATUS
;  1434          2422  2
;  1435          2423  1 END;             ! of routine SET_DIRINDX


                                                        .PSECT  $LOCKEDC1$,NOWRT,2

                                        51  D4 00000 SET_DIRINDX::
                                                      CLRL    STATUS                  ; 2402
                                    12              03  DA 00002                     MTPR    #3, #18                 ; 2410
                                            0080    C0  D5 00005                     TSTL    176(FCB)                ; 2412
                                                    06  13 00009                     BEQL    1$
                                22  A0              01  88 0000B                     BISB2   #1, 34(FCB)             ; 2415
                                                    51  D6 0000F                     INCL    STATUS                  ; 2416
                                    12              00  DA 00011 1$:                 MTPR    #0, #18                 ; 2419
                                    50              51  D0 00014                     MOVL    STATUS, R0              ; 2423
```

```
                                        05 00017        RSB                                     ;

; Routine Size:  24 bytes,    Routine Base:  $LOCKEDC1$ + 0000

; 1436          2424  1
; 1437          2425  1 ! Note that just prior to the SET_DIRINDX routine the psects were
; 1438          2426  1 ! changed to the locked psect because the SET_DIRINDX routine must
; 1439          2427  1 ! be locked.  Any routines added at this point will be locked also,
; 1440          2428  1 ! so unless they need to be locked, put them prior to SET_DIRINDX.
; 1441          2429  1 !
; 1442          2430  1
; 1443          2431  1 END
; 1444          2432  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $CODE$                         1551  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $LOCKEDC1$                       24  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL    EL,  CON,NOPIC,ALIGN(2)



;                               Library Statistics
;
;                                  -------- Symbols --------      Pages       Processing
;       File                       Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA18:[SYSLIB]LIB.L32;1    18619        96         0        1000        00:02.0



;                                       COMMAND QUALIFIERS

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

; Size:          1575 code + 0 data bytes
; Run Time:         01:21.6
; Elapsed Time:     02:08.8
; Lines/CPU Min:     1787
; Lexemes/CPU-Min:  53123
; Memory Used:   372 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

0443 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

DELBAD
LIS

QUOTAUTIL
LIS

CREHDR
LIS

RDBLOK
LIS

MAKACC
LIS

CREATE
LIS

DEACCS
LIS

EXTHDR
LIS