H 5

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

```
_$25

Symb
----
IOC$
IO_C
IO_C
IO_D
IO_F
IO_S
KILL

KILL
KILL
LB_E
LB_D
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

LOCK
LOCK
LOCK
LOC_
LOC_
L_CC
L_CC
L_DA
L_DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAKc
MAP_
MAP_

MAP
MARK
MARK
MARK
MARK
MARK
```

```
**FILE**ID**CREHDR
```

```
  CCCCCCCC  RRRRRRRR    EEEEEEEEEE  HH      HH  DDDDDDDD    RRRRRRRR
  CCCCCCCC  RRRRRRRR    EEEEEEEEEE  HH      HH  DDDDDDDD    RRRRRRRR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RRRRRRRR    EEEEEEEE    HHHHHHHHHH  DD      DD  RRRRRRRR
CC          RRRRRRRR    EEEEEEEE    HHHHHHHHHH  DD      DD  RRRRRRRR
CC          RR  RR      EE          HH      HH  DD      DD  RR  RR
CC          RR  RR      EE          HH      HH  DD      DD  RR  RR
CC          RR    RR    EE          HH      HH  DD      DD  RR    RR      ....
CC          RR    RR    EE          HH      HH  DD      DD  RR    RR      ....
  CCCCCCCC  RR      RR  EEEEEEEEEE  HH      HH  DDDDDDDD    RR      RR    ....
  CCCCCCCC  RR      RR  EEEEEEEEEE  HH      HH  DDDDDDDD    RR      RR    ....

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

```
0 MODULE CREHDR (
0                 LANGUAGE (BLISS32),
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  F11ACP Structure Level 2
1 !
1 ! ABSTRACT:
1 !
1 !       This routine creates a new file ID by allocating a file number from the
1 !       index file bitmap. It returns an empty file header, verified for use.
1 !
1 ! ENVIRONMENT:
1 !
1 !       STARLET operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  28-Mar-1977  13:49
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-022 CDS0017         Christian D. Saether    20-Aug-1984
1 !               Force fcb for indexf to be stale always before
1 !               attempting to map vbns.
1 !
1 !       V03-021 CDS0016         Christian D. Saether    13-Aug-1984
1 !               Back off an extra dot in ACG0438.
```

```
 1 !
 1 !        V03-020 ACG0438         Andrew C. Goldstein,    1-Aug-1984  11:55
 1 !                Add cache interlock logic on FID cache; use central
 1 !                dequeue routine.
 1 !
 1 !        V03-019 LMP0278         L. Mark Pilant,         12-Jul-1984  10:58
 1 !                Fix a bug that caused the EXBYTLM error if it was necessary
 1 !                to turn the index file window.
 1 !
 1 !        V03-018 CDS0015         Christian D. Saether    17-Apr-1984
 1 !                Have MAP_IDX check to see whether curr_lckindx is
 1 !                for the index file to avoid releasing it if so.
 1 !
 1 !        V03-017 CDS0014         Christian D. Saether    11-Apr-1984
 1 !                Release allocation lock prior to serializing on
 1 !                new primary header.  This eliminates potential
 1 !                deadlocks when the new primary header is a valid
 1 !                header that someone else is messing with.
 1 !
 1 !        V03-016 CDS0013         Christian D. Saether     1-Apr-1984
 1 !                ACG0409 forgot to rewrite indexf bitmap buffer.  No joke.
 1 !
 1 !        V03-015 ACG0409         Andrew C. Goldstein,    21-Mar-1984  19:40
 1 !                Redesign file ID cacheing algorithm so that file ID's
 1 !                beyond the index file EOF are not cached. Eliminate
 1 !                BASH_HEADERS routine; general code cleanup to remove
 1 !                kernel calls. CHECK_HEADER2 no longer writes USER_STATUS.
 1 !
 1 !        V03-014 ACG0404         Andrew C. Goldstein,    15-Mar-1984  17:37
 1 !                Correct releasing of file sync lock when retrying for a header
 1 !
 1 !        V03-013 CDS0012         Christian D. Saether    23-Feb-1984
 1 !                Eliminate references to FLUSH_LOCK_BASIS.
 1 !
 1 !        V03-012 CDS0011         Christian D. Saether    27-Dec-1983
 1 !                Use BIND_COMMON macro.
 1 !
 1 !        V03-011 CDS0010         Christian D. Saether    12-Dec-1983
 1 !                Start of XQP code is at symbol INITXQP now.
 1 !
 1 !        V03-010 CDS0009         Christian D. Saether     5-Oct-1983
 1 !                Fix bug restoring privileges to the PCB.
 1 !
 1 !        V03-009 CDS0008         Christian D. Saether     3-Oct-1983
 1 !                Save/restore CURR_LCKINDX where necessary rather
 1 !                than PRIM_LCKINDX.
 1 !
 1 !        V03-008 CDS0007         Christian D. Saether    13-Sep-1983
 1 !                Modify interface to allocation serialization.
 1 !
 1 !        V03-007 CDS0006         Christian D. Saether    12-May-1983
 1 !                Serialize header creation.
 1 !
 1 !        V03-006 CDS0005         Christian D. Saether     1-Mar-1983
 1 !                Need BYPASS privilege also.
 1 !
 1 !        V03-005 CDS0004         Christian D. Saether    20-Feb-1983
```

```
1 !                     Call MAP_VBN before checking FILESIZE so that
1 !                     header is checked before deciding to extend
1 !                     index file.
1 !                     Also make READ_IDX_HEADER insensitive to headers that
1 !                     map more than the FCB knows about.
1 !                     Totally punt figuring out what to do with EFBLK
1 !                     for the index file.
1 !
1 !             V03-004 CDS0003         Christian D. Saether    13-Jan-1983
1 !                     Separately save and restore PHD privs.
1 !
1 !             V03-003 CDS0002         Christian D. Saether    28-Dec-1982
1 !                     Give priv around QIO.
1 !
1 !             V03-002 CDS0001         C Saether                3-Aug-1982
1 !                     Change QIOW to QIO with completion AST.
1 !
1 !             V03-001 ACG0273         Andrew C. Goldstein,    23-Mar-1982  10:50
1 !                     Use random file sequence number if old header is junk,
1 !                     use alternate index file header if primary is suspect
1 !
1 !             V02-007 ACG0229         Andrew C. Goldstein,    23-Dec-1981  21:53
1 !                     Count file ID cache hits and misses
1 !
1 !             V02-006 ACG0167         Andrew C. Goldstein,    16-Apr-1980  19:25
1 !                     Previous revision history moved to F11B.REV
1 !**
1
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32';
1 REQUIRE 'SRC$:FCPDEF.B32';
1
1
1 FORWARD ROUTINE
1         CREATE_HEADER   : L_NORM,          ! create file ID and header
1         FILL_FID_CACHE  : L_NORM NOVALUE,  ! load file ID cache from bitmap
1         INIT_FID_CACHE  : L_NORM NOVALUE,  ! initialize file ID cache lock
1         READ_NEW_HEADER : L_NORM,          ! read new file header block
1         HANDLER,                           ! local condition handler
1         READ_IDX_HEADER : L_NORM,          ! read index file header
1         MAP_IDX         : L_NORM;          ! map vbn for index file.
```

```
1 GLOBAL ROUTINE CREATE_HEADER (FILE_ID) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !        This routine creates a new file ID by searching the volume's index
1 !        file bitmap for the first free file number. It also checks that a
1 !        header for the file number is present in the index file. It reads
1 !        the old header and establishes the file sequence number for the
1 !        new one.
1 !
1 ! CALLING SEQUENCE:
1 !        CREATE_HEADER (ARG1)
1 !
1 ! INPUT PARAMETERS:
1 !        NONE
1 !
1 ! IMPLICIT INPUTS:
1 !        CURRENT_VCB: address of volume's VCB
1 !
1 ! OUTPUT PARAMETERS:
1 !        ARG1: address to store file ID of created header
1 !
1 ! IMPLICIT OUTPUTS:
1 !        NEW_FID: file number of header created
1 !        NEW_FID_RVN: RVN of above
1 !
1 ! ROUTINE VALUE:
1 !        address of buffer containing new header
1 !
1 ! SIDE EFFECTS:
1 !        VCB and index file bitmap altered, header block read
1 !
1 !--
1
2 BEGIN
2
2 MAP
2        FILE_ID          : REF BBLOCK;    ! new file ID of header
2
2 LABEL
2        GET_FILE_NUM;                     ! acquire a file number
2
2 LOCAL
2        CACHE_FLUSHED,                    ! flag indicating cluster caches flushed
2        NEW_LCKINDX      : INITIAL (0),
2        TEMP,                             ! temp storage for current lock index
2        VCB              : REF BBLOCK,    ! local copy of VCB address
2        FID_CACHE        : REF BBLOCK,    ! pointer to file ID cache
2        VBN,                              ! relative block number in bitmap
2        BUFFER           : REF BITVECTOR, ! address of index file bitmap buffer
2        ADDRESS          : REF BITVECTOR, ! address of byte in buffer
2        CURRENT_EOF,                      ! current EOF of index file
2        COUNT,                            ! number of index blocks to bash
2        FILE_NUMBER,                      ! file number allocated
2        IDX_FCB          : REF BBLOCK,    ! FCB of index file
```

```
;  214          1203  2             LBN,                            ! LBN of new file header
;  215          1204  2             HEADER          : REF BBLOCK,   ! address of header buffer
;  216          1205  2             STATUS;                         ! value of CHECK_HEADER call
;  217          1206  2
;  218          1207  2     EXTERNAL
;  219          1208  2             PMS$GL_FIDHIT   : ADDRESSING_MODE (GENERAL),
;  220          1209  2                                             ! count of file ID cache hits
;  221          1210  2             PMS$GL_FIDMISS  : ADDRESSING_MODE (GENERAL),
;  222          1211  2                                             ! count of file ID cache misses
;  223          1212  2             EXE$GQ_SYSTIME  : ADDRESSING_MODE (GENERAL);
;  224          1213  2                                             ! system time of day
;  225          1214  2     BIND_COMMON;
;  226          1215  2
;  227          1216  2     EXTERNAL ROUTINE
;  228          1217  2             ALLOCATION_LOCK : L_NORM NOVALUE, ! interlock allocation
;  229          1218  2             ALLOCATION_UNLOCK : L_NORM NOVALUE, ! release allocation lock.
;  230          1219  2             SERIAL_FILE     : L_NORM,       ! serialize file processing
;  231          1220  2             RELEASE_SERIAL_LOCK : L_NORM NOVALUE, ! release processing lock
;  232          1221  2             DEQ_LOCK        : L_NORM,       ! dequeue a lock
;  233          1222  2             READ_BLOCK      : L_NORM,       ! read block from disk
;  234          1223  2             WRITE_BLOCK     : L_NORM,       ! write block to disk
;  235          1224  2             DELETE_FID      : L_NORM,       ! flush file ID cache and release lock
;  236          1225  2             RELEASE_LOCKBASIS : L_NORM,     ! release buffers under specified lock
;  237          1226  2             CACHE_LOCK      : L_NORM,       ! acquire cache sync lock
;  238          1227  2             EXTEND_INDEX    : L_NORM,       ! extend the index file
;  239          1228  2             ERASE_BLOCKS    : L_NORM,       ! erase blocks on disk
;  240          1229  2             CHECKSUM        : L_NORM,       ! compute file header checksum
;  241          1230  2             WRITE_HEADER    : L_NORM,       ! write current file header
;  242          1231  2             RESET_LBN       : L_NORM,       ! change backing LBN of buffer
;  243          1232  2             INVALIDATE      : L_NORM,       ! invalidate a buffer
;  244          1233  2             CREATE_BLOCK    : L_NORM,       ! materialize a block buffer
;  245          1234  2             CHECK_HEADER2   : L_NORM,       ! verify file header
;  246          1235  2             MARK_DIRTY      : L_NORM;       ! mark buffer for write-back
;  247          1236  2
;  248          1237  2     ! Serialize further file header creation processing.
;  249          1238  2     !
;  250          1239  2
;  251          1240  2     ALLOCATION_LOCK ();
;  252          1241  2
;  253          1242  2     ! The outer loop performs retries if blocks in the index file are bad or
;  254          1243  2     ! are valid file headers. A block containing a valid file header is never
;  255          1244  2     ! used to create a new file; it is simply left marked in use for recovery.
;  256          1245  2     ! Bad header blocks are simply left marked in use in the index file bitmap;
;  257          1246  2     ! they will show up in a verify but are otherwise harmless.
;  258          1247  2     !
;  259          1248  2
;  260          1249  2     VCB = .CURRENT_VCB;
;  261          1250  2     FID_CACHE = .BBLOCK [.VCB[VCB$L_CACHE], VCA$L_FIDCACHE];
;  262          1251  2     CACHE_FLUSHED = 0;
;  263          1252  2     WHILE 1 DO
;  264          1253  3         GET_FILE_NUM: BEGIN
;  265          1254  3
;  266          1255  3     ! See if a file number is available in the file number cache. If not,
;  267          1256  3     ! we scan the index file bitmap for the first free (zero) bit. This is done
;  268          1257  3     ! by starting with the block recorded in the VCB and looking at each block
;  269          1258  3     ! with a character scan.
;  270          1259  3     !
```

```
3
3      IF .FID_CACHE[VCA$W_FIDCOUNT] EQL 0
3      THEN
4          BEGIN
4          PMS$GL_FIDMISS = .PMS$GL_FIDMISS + 1;
4          VBN = .VCB[VCB$B_IBMAPVBN];
4
4          IF NOT
5              BEGIN
5              UNTIL .VBN GEQ .VCB[VCB$B_IBMAPSIZE] DO
6                  BEGIN
6                  BUFFER = READ_BLOCK (.VBN + .VCB[VCB$L_IBMAPLBN], 1, INDEX_TYPE);
6                  IF NOT CH$FAIL (ADDRESS = CH$FIND_NOT_CH (512, .BUFFER, 255))
6                  THEN EXITLOOP 0;
6                  VBN = .VBN + 1;
6                  END
5              END
5
5 ! Having found a bitmap block with free files in it, attempt to fill the
5 ! file ID cache. If it refuses to fill, it's because we're at the index
5 ! file EOF.
5 !
5
4          THEN FILL_FID_CACHE (.VCB, .BUFFER, .VBN);
4          IF .FID_CACHE[VCA$W_FIDCOUNT] EQL 0
4          THEN
5              BEGIN
5
5 ! If the index file EOF coincides with the physical end of file, we have to
5 ! extend the index file. Otherwise, we just have to push the EOF. Before
5 ! extending the index file, if we are in a cluster, ask for a cluster-wide
5 ! flush of the file ID caches.
5 !
5
5              IDX_FCB = .VCB[VCB$L_FCBFL];
5              CURRENT_EOF = .IDX_FCB[FCB$L_EFBLK];
5              IF .CURRENT_EOF GEQU .IDX_FCB[FCB$L_FILESIZE]
5              THEN
6                  BEGIN
6                  IF NOT .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
6                  AND NOT .CACHE_FLUSHED
6                  THEN
7                      BEGIN
7                      LOCAL IDX_FILE_ID, LOCK_ID;
7                      DELETE_FID (0);
7                      RELEASE_LOCKBASIS (-1);
7                      ALLOCATION_UNLOCK ();
7                      IDX_FILE_ID = FID$C_INDEXF OR .CURRENT_VCB[VCB$W_RVN] ^ 24;
7                      LOCK_ID = 0;
7                      CACHE_LOCK (.IDX_FILE_ID, LOCK_ID, 1);
7                      ALLOCATION_LOCK ();
7                      DEQ_LOCK (.LOCK_ID);
7                      CACHE_FLUSHED = -1;
7                      LEAVE GET_FILE_NUM;
7                      END
6                  ELSE
6                      EXTEND_INDEX ();
```

```
6                       END
6
6   ! Move the EOF and zero the intervening blocks. Note that this version
6   ! of the file system always sets the index file EOF to be physical end
6   ! of file, because the index file is zeroed on extend. This code is
6   ! present for compatibility with past and future file systems that may
6   ! not zero the index file on extend. Serialize activity on the index
6   ! file header.
6   !
6
5               ELSE
6                   BEGIN
6                   TEMP = .CURR_LCKINDX;
6                   SERIAL_FILE (IDX_FCB [FCB$W_FID]);
6
6                   LBN = MAP_IDX (.CURRENT_EOF+1, COUNT);
6                   ERASE_BLOCKS (.LBN, .COUNT, .IO_CHANNEL);
6                   CURRENT_EOF = .CURRENT_EOF + .COUNT;
6
6                   HEADER = READ_IDX_HEADER ();
6                   BBLOCK [HEADER[FH2$W_RECATTR], FAT$L_EFBLK] = ROT (.CURRENT_EOF+1, 16);
6                   BBLOCK [HEADER[FH2$W_RECATTR], FAT$W_FFBYTE] = 0;
6                   IF .HEADER [FH2$B_IDOFFSET] GEQU ($BYTEOFFSET (FH2$L_HIGHWATER)+4)/2
6                   THEN HEADER [FH2$L_HIGHWATER] = .CURRENT_EOF + 1;
6
6                   CHECKSUM (.HEADER);
6                   WRITE_HEADER ();
6                   IDX_FCB[FCB$L_EFBLK] = .CURRENT_EOF;
6                   RESET_LBN (.HEADER, .VCB[VCB$L_IXHDR2LBN]);
6                   WRITE_BLOCK (.HEADER);
6                   INVALIDATE (.HEADER);
6
6                   RELEASE_SERIAL_LOCK (.CURR_LCKINDX);
6                   CURR_LCKINDX = .TEMP;
5                   END;
5
5   ! Go around the loop to try to allocate a file number again.
5   !
5
5               LEAVE GET_FILE_NUM;
5               END
4           ELSE
4
4   ! We successfully filled the file ID cache from the bitmap.  Write back
4   ! the index file bitmap buffer.
4   !
4
4               WRITE_BLOCK (.BUFFER);
4
4           END
4
4   ! If the file ID cache had entries in it, all we have to do is check one out.
4   !
4
3       ELSE
3           PMS$GL_FIDHIT = .PMS$GL_FIDHIT + 1;
3
```

```
3      FILE_NUMBER = .FID_CACHE[VCA$L_FIDLIST];
3      FID_CACHE[VCA$W_FIDCOUNT] = .FID_CACHE[VCA$W_FIDCOUNT] - 1;
3      CH$MOVE (.FID_CACHE[VCA$W_FIDCOUNT]*4,
3               FID_CACHE[VCA$L_FIDLIST]+4,
3               FID_CACHE[VCA$L_FIDLIST]);
3
3      NEW_FID = .FILE_NUMBER;
3      NEW_FID_RVN = .CURRENT_RVN;          ! record for cleanup
3
3 ! Map the file header. If it fails to map, we have screwed up badly.
3 !
3
3      VBN = .FILE_NUMBER + .VCB[VCB$B_IBMAPSIZE] + .VCB[VCB$W_CLUSTER]*4;
3      LBN = MAP_IDX (.VBN);
3      IF .LBN EQL -1 THEN BUG_CHECK (HDRNOTMAP, FATAL, 'Allocated file header not mapped');
3
3      FILE_ID[FID$W_NUM] = .FILE_NUMBER<0,16>;
3      FILE_ID[FID$B_NMX] = .FILE_NUMBER<16,8>;
3      FILE_ID[FID$B_RVN] = .CURRENT_RVN;
3
3 ! If this is the creation of a new primary header, PRIM_LCKINDX will
3 ! be zero.  In that case, serialize further processing on that header.
3 ! If extension headers are being allocated, the primary lock index has
3 ! already been established.
3 !
3
3      IF .PRIM_LCKINDX EQL 0
3      THEN
4          BEGIN
4
4 ! Release the allocation lock prior to serializing on this file id.
4 ! This could be a valid header that another process is trying to modify
4 ! allocation on, and if so, we would deadlock if the allocation lock
4 ! were not released now.
4 !
4
4          ALLOCATION_UNLOCK ();
4          PRIM_LCKINDX = SERIAL_FILE (.FILE_ID);
4          NEW_LCKINDX = 1;
4          END;
3
3 ! Read the header; then check the block read for resemblence to a file header.
3 !
3
3      HEADER = READ_NEW_HEADER (.LBN);
3
3      IF .HEADER NEQ 0
3      THEN
4          BEGIN
4          FILE_ID[FID$W_SEQ] = .HEADER[FH2$W_FID_SEQ];
4          STATUS = CHECK_HEADER2 (.HEADER, .FILE_ID);
4
4 ! Make the final checks that the block is acceptable as a file header. We do
4 ! not use valid file headers. Also, we skip file numbers with the low 16 bits
4 ! all zero to avoid confusing the old FCS-11. Also skip file numbers in the
4 ! reserved file number range to avoid total confusion if the volume is damaged.
4 !
```

```
4
4         IF .FILE_ID[FID$W_NUM] EQL 0
4         THEN
4             WRITE_BLOCK (.HEADER)
4         ELSE
4             IF NOT .STATUS
5             AND NOT (.FILE_ID[FID$B_NMX] EQL 0
5                      AND .FILE_ID[FID$W_NUM] LEQU .CURRENT_VCB[VCB$B_RESFILES])
4             THEN EXITLOOP;
3         END;
3
3 ! If we got this far, i.e., did not exit the loop, we do not want to use
3 ! this file header for some reason.  Before going around another time,
3 ! release the serialization lock if we got one in this routine, and then
3 ! reacquire the allocation lock for another pass around the loop.
3 !
3
3     IF .NEW_LCKINDX
3     THEN
4         BEGIN
4         IF .HEADER NEQ 0
4         THEN INVALIDATE (.HEADER);
4         RELEASE_SERIAL_LOCK (.PRIM_LCKINDX);
4         PRIM_LCKINDX = 0;
4         ALLOCATION_LOCK ();
3         END;
3
2     END;                                    ! end of file number allocation loop
2
2 HEADER_LBN = .LBN;                          ! record LBN of new header
2
2 IF .STATUS EQL 0
2 AND .(.HEADER)<0,32> NEQ 0
2 THEN FILE_ID[FID$W_SEQ] = .EXE$GQ_SYSTIME<16,16>;
2 FILE_ID[FID$W_SEQ] = .FILE_ID[FID$W_SEQ] + 1;
2 CH$MOVE (FID$C_LENGTH, .FILE_ID, HEADER[FH2$W_FID]);
2 HEADER[FH2$B_FID_RVN] = 0;
2
2 MARK_DIRTY (.HEADER);
2 .HEADER
2
1 END;                                        ! end of routine CREATE_HEADER


                                                    .TITLE  CREHDR
                                                    .IDENT  \V04-000\

                                                    .EXTRN  PMS$GL_FIDHIT, PMS$GL_FIDMISS
                                                    .EXTRN  EXE$GQ_SYSTIME, ALLOCATION_LOCK
                                                    .EXTRN  ALLOCATION_UNLOCK
                                                    .EXTRN  SERIAL_FILE, RELEASE_SERIAL_LOCK
                                                    .EXTRN  DEQ_LOCK, READ_BLOCK
                                                    .EXTRN  WRITE_BLOCK, DELETE_FID
                                                    .EXTRN  RELEASE_LOCKBASIS
                                                    .EXTRN  CACHE_LOCK, EXTEND_INDEX
                                                    .EXTRN  ERASE_BLOCKS, CHECKSUM
                                                    .EXTRN  WRITE_HEADER, RESET_LBN
```

```
                                                                        .EXTRN  INVALIDATE, CREATE_BLOCK
                                                                        .EXTRN  CHECK_HEADER2, MARK_DIRTY
                                                                        .EXTRN  BUG$_ADRNOTMAP

                                                                        .PSECT  $CODE$,NOWRT,2

                                                      0BFC 00000        .ENTRY  CREATE_HEADER, Save R2,R3,R4,R5,R6,R7,R8,-  ; 1146
                                                                                R9,R11                                      ;
                                         5E             2C  C2 00002            SUBL2   #44, SP                             ;
                                                  20    AE  D4 00005            CLRL    NEW_LCKINDX                         ; 1182
                                  0000G  CF             00  FB 00008            CALLS   #0, ALLOCATION_LOCK                 ; 1240
                                         59       98    AA  D0 0000D            MOVL    -104(BASE), VCB                     ; 1249
                                         56       58    B9  D0 00011            MOVL    @88(VCB), FID_CACHE                 ; 1250
                                                  1C    AE  D4 00015            CLRL    CACHE_FLUSHED                       ; 1251
                                                  02    A6  B5 00018 1$:        TSTW    2(FID_CACHE)                        ; 1261
                                                        03  13 0001B            BEQL    2$                                  ;
                                                      014B  31 0001D            BRW     13$                                 ;
                                              00000000G 00  D6 00020 2$:        INCL    PMS$GL_FIDMISS                      ; 1264
                                  18     AE       3A    A9  9A 00026            MOVZBL  58(VCB), VBN                        ; 1265
          18    AE        38    A9       08             00  ED 0002B 3$:        CMPZV   #0, #8, 56(VCB), VBN                ; 1269
                                                        3A  15 00032            BLEQ    6$                                  ;
                                                        03  DD 00034            PUSHL   #3                                  ; 1271
                                                        01  DD 00036            PUSHL   #1                                  ;
                                         50       20    AE  D0 00038            MOVL    VBN, R0                             ;
                                                  30  B940  9F 0003C            PUSHAB  @48(VCB)[R0]                        ;
                                  0000G  CF             03  FB 00040            CALLS   #3, READ_BLOCK                      ;
                                     0C  AE             50  D0 00045            MOVL    R0, BUFFER                          ;
                0C    BE          0200   8F       FF    8F  3B 00049            SKPC    #255, #512, @BUFFER                 ; 1272
                                                        02  12 00051            BNEQ    4$                                  ;
                                                        51  D4 00053            CLRL    R1                                  ;
                                         6E             51  D0 00055 4$:        MOVL    R1, ADDRESS                         ;
                                                        6E  D5 00058            TSTL    ADDRESS                             ;
                                                        05  12 0005A            BNEQ    5$                                  ;
                                                  18    AE  D6 0005C            INCL    VBN                                 ; 1274
                                                        CA  11 0005F            BRB     3$                                  ; 1269
                                                  18    AE  DD 00061 5$:        PUSHL   VBN                                 ; 1283
                                                  10    AE  DD 00064            PUSHL   BUFFER                              ;
                                                        59  DD 00067            PUSHL   VCB                                 ;
                                  0000V  CF             03  FB 00069            CALLS   #3, FILL_FID_CACHE                  ;
                                                  02    A6  B5 0006E 6$:        TSTW    2(FID_CACHE)                        ; 1284
                                                        03  13 00071            BEQL    7$                                  ;
                                                      00EB  31 00073            BRW     12$                                 ;
                                         58             69  D0 00076 7$:        MOVL    (VCB), IDX_FCB                      ; 1294
                                         57       3C    AB  D0 00079            MOVL    60(IDX_FCB), CURRENT_EOF            ; 1295
                                  38     AB             57  D1 0007D            CMPL    CURRENT_EOF, 56(IDX_FCB)            ; 1296
                                                        59  1F 00081            BLSSU   10$                                 ;
                                         50       94    AA  D0 00083            MOVL    -108(BASE), R0                      ; 1299
                                         49       3C    A0  E8 00087            BLBS    60(R0), 8$                          ;
                                         45       1C    AE  E8 0008B            BLBS    CACHE_FLUSHED, 8$                   ; 1300
                                                        7E  D4 0008F            CLRL    -(SP)                               ; 1304
                                  00C0G  CF             01  FB 00091            CALLS   #1, DELETE_FID                      ;
                                         7E             01  CE 00096            MNEGL   #1, -(SP)                           ; 1305
                                  0000G  CF             01  FB 00099            CALLS   #1, RELEASE_LOCKBASIS               ;
                                  0000G  CF             00  FB 0009E            CALLS   #0, ALLOCATION_UNLOCK               ; 1306
                                         50       98    AA  D0 000A3            MOVL    -104(BASE), R0                      ; 1307
                                         50       0E    A0  3C 000A7            MOVZWL  14(R0), R0                          ;
                           50            50             18  78 000AB            ASHL    #24, R0, R0                         ;
```

```
                   50            01  88 000AF        BISB2   #1, IDX_FILE_ID                              ;
                           24    AE  D4 000B2        CLRL    LOCK_ID                                      ; 1308
                                 01  DD 000B5        PUSHL   #1                                           ; 1309
                           28    AE  9F 000B7        PUSHAB  LOCK_ID                                      ;
                                 50  DD 000BA        PUSHL   IDX_FILE_ID                                  ;
               0000G   CF        03  FB 000BC        CALLS   #3, CACHE_LOCK                               ;
               0000G   CF        00  FB 000C1        CALLS   #0, ALLOCATION_LOCK                          ; 1310
                           24    AE  DD 000C6        PUSHL   LOCK_ID                                      ; 1311
               0000G   CF        01  FB 000C9        CALLS   #1, DEQ_LOCK                                 ;
                  1C   AE        01  CE 000CE        MNEGL   #1, CACHE_FLUSHED                            ; 1312
                                 05  11 000D2        BRB     9$                                           ; 1313
               0000G   CF        00  FB 000D4 8$:    CALLS   #0, EXTEND_INDEX                             ; 1316
                               FF3C  31 000D9 9$:    BRW     1$                                           ; 1296
                  04   AE  14    AA  D0 000DC 10$:   MOVL    20(BASE), TEMP                               ; 1329
                           24    AB  9F 000E1        PUSHAB  36(IDX_FCB)                                  ; 1330
               0000G   CF        01  FB 000E4        CALLS   #1, SERIAL_FILE                              ;
                           28    AE  9F 000E9        PUSHAB  COUNT                                        ; 1332
                           01    A7  9F 000EC        PUSHAB  1(CURRENT_EOF)                               ;
               0000V   CF        02  FB 000EF        CALLS   #2, MAP_IDX                                  ;
                  10   AE        50  D0 000F4        MOVL    R0, LBN                                      ;
                         FF78    CA  DD 000F8        PUSHL   -136(BASE)                                   ; 1333
                           2C    AE  DD 000FC        PUSHL   COUNT                                        ;
                           18    AE  DD 000FF        PUSHL   LBN                                          ;
               0000G   CF        03  FB 00102        CALLS   #3, ERASE_BLOCKS                             ;
                       57  28    AE  C0 00107        ADDL2   COUNT, CURRENT_EOF                           ; 1334
               0000V   CF        00  FB 0010B        CALLS   #0, READ_IDX_HEADER                          ; 1336
                       58        50  D0 00110        MOVL    R0, HEADER                                   ;
                       50  01    A7  9E 00113        MOVAB   1(R7), R0                                    ; 1337
   1C    A8            50        10  9C 00117        ROTL    #16, R0, 28(HEADER)                          ;
                           20    A8  B4 0011C        CLRW    32(HEADER)                                   ; 1338
                       28        68  91 0011F        CMPB    (HEADER), #40                                ; 1339
                                 05  1F 00122        BLSSU   11$                                          ;
                  4C   A8  01    A7  9E 00124        MOVAB   1(R7), 76(HEADER)                            ; 1340
                                 58  DD 00129 11$:   PUSHL   HEADER                                       ; 1342
               0000G   CF        01  FB 0012B        CALLS   #1, CHECKSUM                                 ;
               0000G   CF        00  FB 00130        CALLS   #0, WRITE_HEADER                             ; 1343
                  3C   AB        57  D0 00135        MOVL    CURRENT_EOF, 60(IDX_FCB)                     ; 1344
                           2C    A9  DD 00139        PUSHL   44(VCB)                                      ; 1345
                                 58  DD 0013C        PUSHL   HEADER                                       ;
               0000G   CF        02  FB 0013E        CALLS   #2, RESET_LBN                                ;
                                 58  DD 00143        PUSHL   HEADER                                       ; 1346
               0000G   CF        01  FB 00145        CALLS   #1, WRITE_BLOCK                              ;
                                 58  DD 0014A        PUSHL   HEADER                                       ; 1347
               0000G   CF        01  FB 0014C        CALLS   #1, INVALIDATE                               ;
                           14    AA  DD 00151        PUSHL   20(BASE)                                     ; 1349
               0000G   CF        01  FB 00154        CALLS   #1, RELEASE_SERIAL_LOCK                      ;
                  14   AA  04    AE  D0 00159        MOVL    TEMP, 20(BASE)                               ; 1350
                               FEB7  31 0015E        BRW     1$                                           ; 1356
                           0C    AE  DD 00161 12$:   PUSHL   BUFFER                                       ; 1364
               0000G   CF        01  FB 00164        CALLS   #1, WRITE_BLOCK                              ;
                                 06  11 00169        BRB     14$                                          ; 1261
                     0000000G    00  D6 0016B 13$:   INCL    PMS$GL_FIDHIT                                ; 1372
                  14   AE  24    A6  D0 00171 14$:   MOVL    36(FID_CACHE), FILE_NUMBER                   ; 1374
                           02    A6  B7 00176        DECW    2(FID_CACHE)                                 ; 1375
                       50  02    A6  3C 00179        MOVZWL  2(FID_CACHE), R0                             ; 1376
                       50        04  C4 0017D        MULL2   #4, R0                                       ;
   24    A6       28   A6        50  28 00180        MOVC3   R0, 40(FID_CACHE), 36(FID_CACHE)             ; 1378
```

```
                A8   AA      14   AE  D0 00186         MOVL    FILE_NUMBER, -88(BASE)                 ; 1380
                AC   AA      A0   AA  D0 0018B         MOVL    -96(BASE), -84(BASE)                   ; 1381
                     50      38   A9  9A 00190         MOVZBL  56(VCB), R0                            ; 1386
   51                50      14   AE  C1 00194         ADDL3   FILE_NUMBER, R0, R1                    ;
                     50      3C   A9  3C 00199         MOVZWL  60(VCB), R0                            ;
                18   AE         6140  DE 0019D         MOVAL   (R1)[R0], VBN                          ;
                             18   AE  DD 001A2         PUSHL   VBN                                    ; 1387
             0000V   CF           01  FB 001A5         CALLS   #1, MAP_IDX                            ;
                10   AE           50  D0 001AA         MOVL    R0, LBN                                ;
          FFFFFFFF   8F      10   AE  D1 001AE         CMPL    LBN, #-1                               ; 1388
                                  04  12 001B6         BNEQ    15$                                    ;
                                   FEFF 001B8          BUGW                                           ;
                                  0000* 001BA          .WORD   <BUG$_HDRNOTMAP!4>                     ;
                04   BC      14   AE  B0 001BC 15$:    MOVW    FILE_NUMBER, @FILE_ID                  ; 1390
                     50      04   AC  D0 001C1         MOVL    FILE_ID, R0                            ; 1391
                05   A0      16   AE  90 001C5         MOVB    FILE_NUMBER+2, 5(R0)                   ;
                     50      04   AC  D0 001CA         MOVL    FILE_ID, R0                            ; 1392
                04   A0      A0   AA  90 001CE         MOVB    -96(BASE), 4(R0)                       ;
                             18   AA  D5 001D3         TSTL    24(BASE)                               ; 1400
                                  15  12 001D6         BNEQ    16$                                    ;
             0000G   CF           00  FB 001D8         CALLS   #0, ALLOCATION_UNLOCK                  ; 1410
                             04   AC  DD 001DD         PUSHL   FILE_ID                                ; 1411
             0000G   CF           01  FB 001E0         CALLS   #1, SERIAL_FILE                        ;
                18   AA           50  D0 001E5         MOVL    R0, 24(BASE)                           ;
                20   AE           01  D0 001E9         MOVL    #1, NEW_LCKINDX                        ; 1412
                             10   AE  DD 001ED 16$:    PUSHL   LBN                                    ; 1418
             0000V   CF           01  FB 001F0         CALLS   #1, READ_NEW_HEADER                    ;
                     58           50  D0 001F5         MOVL    R0, HEADER                             ;
                                  3E  13 001F8         BEQL    18$                                    ; 1420
                     50      04   AC  D0 001FA         MOVL    FILE_ID, R0                            ; 1423
                02   A0      0A   A8  B0 001FE         MOVW    10(HEADER), 2(R0)                      ;
                             04   AC  DD 00203         PUSHL   FILE_ID                                ; 1424
                                  58  DD 00206         PUSHL   HEADER                                 ;
             0000G   CF           02  FB 00208         CALLS   #2, CHECK_HEADER                       ;
                08   AE           50  D0 0020D         MOVL    R0, STATUS                             ;
                     52      04   AC  D0 00211         MOVL    FILE_ID, R2                            ; 1432
                                  62  B5 00215         TSTW    (R2)                                   ;
                                  09  12 00217         BNEQ    17$                                    ;
                                  58  DD 00219         PUSHL   HEADER                                 ; 1434
             0000G   CF           01  FB 0021B         CALLS   #1, WRITE_BLOCK                        ;
                                  16  11 00220         BRB     18$                                    ;
                     12      08   AE  E8 00222 17$:    BLBS    STATUS, 18$                            ; 1436
                             05   A2  95 00226         TSTB    5(R2)                                  ; 1437
                                  2F  12 00229         BNEQ    21$                                    ;
                     50      98   AA  D0 0022B         MOVL    -104(BASE), R0                         ; 1438
                     51      4F   A0  9A 0022F         MOVZBL  79(R0), R1                             ;
                     62           51  B1 00233         CMPW    R1, (R2)                               ;
                                  22  1F 00236         BLSSU   21$                                    ;
                     1B      20   AE  E9 00238 18$:    BLBC    NEW_LCKINDX, 20$                       ; 1448
                                  58  D5 0023C         TSTL    HEADER                                 ; 1451
                                  07  13 0023E         BEQL    19$                                    ;
                                  58  DD 00240         PUSHL   HEADER                                 ; 1452
             0000G   CF           01  FB 00242         CALLS   #1, INVALIDATE                         ;
                             18   AA  DD 00247 19$:    PUSHL   24(BASE)                               ; 1453
             0000G   CF           01  FB 0024A         CALLS   #1, RELEASE_SERIAL_LOCK                ;
                             18   AA  D4 0024F         CLRL    24(BASE)                               ; 1454
             0000G   CF           00  FB 00252         CALLS   #0, ALLOCATION_LOCK                    ; 1455
```

```
                                    FDBE  31 00257 20$:    BRW     1$                                        ; 1252
                    B0   AA      10   AE  D0 0025A 21$:    MOVL    LBN, -80(BASE)                            ; 1460
                                 08   AE  D5 0025F         TSTL    STATUS                                    ; 1462
                                      10  12 00262         BNEQ    22$                                       ;
                                      68  D5 00264         TSTL    (HEADER)                                  ; 1463
                                      0C  13 00266         BEQL    22$                                       ;
                         50      04   AC  D0 00268         MOVL    FILE_ID, R0                               ; 1464
                    02   A0 00000000G 00  B0 0026C         MOVW    EXE$GQ_SYSTIME+2, 2(R0)                   ;
                         50      04   AC  D0 00274 22$:    MOVL    FILE_ID, R0                               ; 1465
                                 02   A0  B6 00278         INCW    2(R0)                                     ;
          08   A8        04   BC      06  28 0027B         MOVC3   #6, @FILE_ID, 8(HEADER)                   ; 1466
                                 0C   A8  94 00281         CLRB    12(HEADER)                                ; 1467
                                      58  DD 00284         PUSHL   HEADER                                    ; 1469
                  0000G  CF           01  FB 00286         CALLS   #1, MARK_DIRTY                            ;
                         50           58  D0 0028B         MOVL    HEADER, R0                                ; 1472
                                          04 0028E         RET                                               ;

; Routine Size:  655 bytes,    Routine Base:  $CODE$ + 0000
```

```
ROUTINE FILL_FID_CACHE (VCB, BUFFER, VBN) : L_NORM NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine refills the cache from the supplied bitmap buffer.
!       It will not fill the cache with file ID's that represent
!       headers past the current index file EOF.
!
!
! CALLING SEQUENCE:
!       FILL_FID_CACHE (ARG1, ARG2, ARG3)
!
! INPUT PARAMETERS:
!       ARG1: address of volume VCB
!       ARG2: address of bitmap buffer
!       ARG3: relative block number in bitmap
!
! IMPLICIT INPUTS:
!       NONE
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! ROUTINE VALUE:
!       NONE
!
! SIDE EFFECTS:
!       file ID cache modified
!
!--

BEGIN

MAP
        VCB             : REF BBLOCK,   ! local copy of VCB address
        BUFFER          : REF BITVECTOR; ! address of index file bitmap buffer

LOCAL
        CACHE           : REF BBLOCK,   ! pointer to cache block
        FID_CACHE       : REF BBLOCK,   ! pointer to file ID cache
        ADDRESS         : REF BITVECTOR, ! address of byte in buffer
        FREE_COUNT,                     ! count of cache entries to fill
        BITPOS,                         ! bit positon of free bit within byte
        BITPOS2,                        ! bit positon of first used bit
        FILE_NUMBER,                    ! file number found
        IDX_VBN;                        ! current block in index bitmap

BIND_COMMON;


! If the cache is not currently marked valid, attempt to take out the
! cache lock if we are in a cluster and may do so.
```

```
;  542          1530  2 !
;  543          1531  2
;  544          1532  2 CACHE = .VCB[VCB$L_CACHE];
;  545          1533  2 FID_CACHE = .CACHE[VCA$L_FIDCACHE];
;  546          1534  2 IF NOT .CACHE[VCA$V_FIDC_VALID]
;  547          1535  2 THEN INIT_FID_CACHE (.CACHE);
;  548          1536  2
;  549          1537  2 ! Fill the cache from the supplied bitmap buffer. Find each byte containing
;  550          1538  2 ! a free bit, and then find the free bit.
;  551          1539  2 !
;  552          1540  2
;  553          1541  2 ADDRESS = .BUFFER;
;  554          1542  2 FREE_COUNT = .FID_CACHE[VCA$W_FIDSIZE]/2 - .FID_CACHE[VCA$W_FIDCOUNT] + 1;
;  555          1543  2
;  556          1544  2 WHILE 1 DO
;  557          1545  3     BEGIN
;  558          1546  3     IF CH$FAIL (ADDRESS = CH$FIND_NOT_CH (.BUFFER+512-.ADDRESS, .ADDRESS, 255))
;  559          1547  3     THEN EXITLOOP;
;  560          1548  3     FFC (%REF (0), %REF (8), .ADDRESS, BITPOS);
;  561          1549  3     FILE_NUMBER = .VBN*4096 + (.ADDRESS-.BUFFER)*8 + .BITPOS + 1;
;  562          1550  3
;  563          1551  3 ! Check file number against index file EOF and the maximum file limit.
;  564          1552  3 !
;  565          1553  3
;  566          1554  3     IF .FILE_NUMBER + .VCB[VCB$B_IBMAPSIZE] + .VCB[VCB$W_CLUSTER]*4
;  567          1555  3         GTRU .BBLOCK [.VCB[VCB$L_FCBFL], FCB$L_EFBLK]
;  568          1556  3     OR .FILE_NUMBER GTRU .VCB[VCB$L_MAXFILES]
;  569          1557  3     THEN EXITLOOP;
;  570          1558  3
;  571          1559  3 ! Enter the file number in the cache and mark it busy in the bitmap.
;  572          1560  3 ! Exit the loop if the cache is now full enough.
;  573          1561  3 !
;  574          1562  3
;  575          1563  3     ADDRESS[.BITPOS] = 1;
;  576          1564  3     FID_CACHE[VCA$W_FIDCOUNT] = .FID_CACHE[VCA$W_FIDCOUNT] + 1;
;  577          1565  3     VECTOR [FID_CACHE[VCA$L_FIDLIST], .FID_CACHE[VCA$W_FIDCOUNT]-1] = .FILE_NUMBER;
;  578          1566  3     FREE_COUNT = .FREE_COUNT - 1;
;  579          1567  3     IF .FREE_COUNT LEQ 0
;  580          1568  3     OR NOT .CACHE[VCA$V_FIDC_VALID]
;  581          1569  3     THEN EXITLOOP;
;  582          1570  2     END;                                  ! end of bitmap processing loop
;  583          1571  2
;  584          1572  2 IDX_VBN = .VBN;                           ! update current VBN of index file bitmap
;  585          1573  2 IF .FILE_NUMBER<0,12> EQL 0
;  586          1574  2 THEN IDX_VBN = .IDX_VBN + 1;
;  587          1575  2 VCB[VCB$B_IBMAPVBN] = .IDX_VBN;
;  588          1576  2
;  589          1577  1 END;                                      ! end of routine FILL_FID_CACHE


                                        01FC 00000 FILL_FID_CACHE:
                                                         .WORD    Save R2,R3,R4,R5,R6,R7,R8                    ; 1473
                         50        04   AC  D0 00002     MOVL     VCB, R0                                      ; 1532
                         54        58   A0  D0 00006     MOVL     88(R0), CACHE                                ;
```

```
                              52            64  D0 0000A       MOVL     (CACHE), FID_CACHE                    ; 1533
                              07       0B   A4  E8 0000D       BLBS     11(CACHE), 1$                         ; 1534
                                            54  DD 00011       PUSHL    CACHE                                 ; 1535
                     0000V    CF            01  FB 00013       CALLS    #1, INIT_FID_CACHE                    ;
                              57       08   AC  D0 00018 1$:   MOVL     BUFFER, ADDRESS                       ; 1541
                              53            62  3C 0001C       MOVZWL   (FID_CACHE), R3                       ; 1542
                              53            02  C6 0001F       DIVL2    #2, R3                                ;
                              50       02   A2  3C 00022       MOVZWL   2(FID_CACHE), R0                      ;
                              53            50  C2 00026       SUBL2    R0, R3                                ;
                                            53  D6 00029       INCL     FREE_COUNT                            ;
                 50      08   AC            57  C3 0002B 2$:   SUBL3    ADDRESS, BUFFER, R0                   ; 1546
                              50     0200   C0  9E 00030       MOVAB    512(R0), R0                           ;
                 67           50       FF   8F  3B 00035       SKPC     #255, R0, (ADDRESS)                   ;
                                            02  12 0003A       BNEQ     3$                                    ;
                                            51  D4 0003C       CLRL     R1                                    ;
                              57            51  D0 0003E 3$:   MOVL     R1, ADDRESS                           ;
                                            53  13 00041       BEQL     5$                                    ;
        58       67           08            00  EB 00043       FFC      #0, #8, (ADDRESS), BITPOS             ; 1548
                 50      0C   AC            0C  78 00048       ASHL     #12, VBN, R0                          ; 1549
                 51           57       08   AC  C3 0004D       SUBL3    BUFFER, ADDRESS, R1                   ;
                              50          6041  7E 00052       MOVAQ    (R0)[R1], R0                          ;
                              56       01 A840  9E 00056       MOVAB    1(BITPOS)[R0], FILE_NUMBER            ;
                              51       04   AC  D0 0005B       MOVL     VCB, R1                               ; 1554
                              50       38   A1  9A 0005F       MOVZBL   56(R1), R0                            ;
                 55           56            50  C1 00063       ADDL3    R0, FILE_NUMBER, R5                   ;
                              50       3C   A1  3C 00067       MOVZWL   60(R1), R0                            ;
                              55          6540  DE 0006B       MOVAL    (R5)[R0], R5                          ;
                              50            61  D0 0006F       MOVL     (R1), R0                              ; 1555
                         3C   A0            55  D1 00072       CMPL     R5, 60(R0)                            ;
                                            1E  1A 00076       BGTRU    5$                                    ;
                         44   A1            56  D1 00078       CMPL     FILE_NUMBER, 68(R1)                   ; 1556
                                            18  1A 0007C       BGTRU    5$                                    ;
                 00           67            58  E2 0007E       BBSS     BITPOS, (ADDRESS), 4$                 ; 1563
                                       02   A2  B6 00082 4$:   INCW     2(FID_CACHE)                          ; 1564
                              50       02   A2  3C 00085       MOVZWL   2(FID_CACHE), R0                      ; 1565
                         20 A240            56  D0 00089       MOVL     FILE_NUMBER, 32(FID_CACHE)[R0]        ;
                                            53  D7 0008E       DECL     FREE_COUNT                            ; 1566
                                            04  15 00090       BLEQ     5$                                    ; 1567
                              95       0B   A4  E8 00092       BLBS     11(CACHE), 2$                         ; 1568
                              51       0C   AC  D0 00096 5$:   MOVL     VBN, IDX_VBN                          ; 1572
                       0FFF   8F            56  B3 0009A       BITW     FILE_NUMBER, #4095                    ; 1573
                                            02  12 0009F       BNEQ     6$                                    ;
                                            51  D6 000A1       INCL     IDX_VBN                               ; 1574
                              50       04   AC  D0 000A3 6$:   MOVL     VCB, R0                               ; 1575
                         3A   A0            51  90 000A7       MOVB     IDX_VBN, 58(R0)                       ;
                                                04 000AB       RET                                            ; 1577

; Routine Size:  172 bytes,    Routine Base:  $CODE$ + 028F
```

```
1 GLOBAL ROUTINE INIT_FID_CACHE (CACHE) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine refills the cache from the supplied bitmap buffer.
1 !       It will not fill the cache with file ID's that represent
1 !       headers past the current index file EOF.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       INIT_FID_CACHE (CACHE)
1 !
1 ! INPUT PARAMETERS:
1 !       CACHE: pointer to main cache block
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       cache marked valid, lock taken out
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       CACHE           : REF BBLOCK;   ! pointer to cache block
2
2 LOCAL
2       FID_CACHE       : REF BBLOCK,   ! pointer to file ID cache
2       INDEX_FID;                      ! lock basis for index file
2
2 BIND_COMMON;
2
2 EXTERNAL ROUTINE
2       CACHE_LOCK      : L_NORM;       ! acquire special cache lock
2
2
2 ! If the cache is not currently marked valid, attempt to take out the
2 ! cache lock if we are in a cluster and may do so.
2 !
2
2 FID_CACHE = .CACHE[VCA$L_FIDCACHE];
2 IF NOT .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_DMT]
2 AND NOT .CURRENT_VCB[VCB$V_WRITE_IF]
2 AND .FID_CACHE[VCA$W_FIDSIZE] GTRU 1
```

```
;  648          1635  2 THEN
;  649          1636  3     BEGIN
;  650          1637  3     IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
;  651          1638  3     THEN
;  652          1639  4         BEGIN
;  653          1640  4         INDEX_FID = FID$C_INDEXF OR .CURRENT_VCB[VCB$W_RVN] ^ 24;
;  654          1641  4         IF CACHE_LOCK (.INDEX_FID, FID_CACHE[VCA$L_FIDCLKID], 0)
;  655          1642  4         THEN CACHE[VCA$V_FIDC_VALID] = 1;
;  656          1643  4         END
;  657          1644  3     ELSE
;  658          1645  3         CACHE[VCA$V_FIDC_VALID] = 1;
;  659          1646  2     END;
;  660          1647  2
;  661          1648  1 END;                                   ! end of routine INIT_FID_CACHE



                                          000C 00000          .ENTRY   INIT_FID_CACHE, Save R2,R3                        ; 1578
                     52        04      AC   D0 00002          MOVL     CACHE, R2                                         ; 1631
                     53                62   D0 00006          MOVL     (R2), FID_CACHE                                   ;
                     51        94      AA   D0 00009          MOVL     -108(BASE), R1                                    ; 1632
        3C       3A  A1                05   E0 0000D          BBS      #5, 58(R1), 2$                                    ;
                     50        98      AA   D0 00012          MOVL     -104(BASE), R0                                    ; 1633
                     34        0B      A0   E8 00016          BLBS     11(R0), 2$                                        ;
                     01                63   B1 0001A          CMPW     (FID_CACHE), #1                                   ; 1634
                                       2F   1B 0001D          BLEQU    2$                                                ;
                     27        3C      A1   E9 0001F          BLBC     60(R1), 1$                                        ; 1637
                     50        98      AA   D0 00023          MOVL     -104(BASE), R0                                    ; 1640
                     50        0E      A0   3C 00027          MOVZWL   14(R0), R0                                        ;
        50           50                18   78 0002B          ASHL     #24, R0, R0                                       ;
                     50                01   88 0002F          BISB2    #1, INDEX_FID                                     ;
                                       7E   D4 00032          CLRL     -(SP)                                             ; 1641
                               04      A3   9F 00034          PUSHAB   4(FID_CACHE)                                      ;
                                       50   DD 00037          PUSHL    INDEX_FID                                         ;
              0000G  CF                03   FB 00039          CALLS    #3, CACHE_LOCK                                    ;
                     0D                50   E9 0003E          BLBC     R0, 2$                                            ;
                     50        04      AC   D0 00041          MOVL     CACHE, R0                                         ; 1642
                 0B  A0                01   88 00045          BISB2    #1, 11(R0)                                        ;
                                            04 00049          RET                                                        ; 1637
                 0B  A2                01   88 0004A 1$:      BISB2    #1, 11(R2)                                        ; 1645
                                            04 0004E 2$:      RET                                                        ; 1648

; Routine Size:  79 bytes,    Routine Base:  $CODE$ + 033B
```

```
ROUTINE READ_NEW_HEADER (LBN) : L_NORM =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine reads the block about to be used for a new file header.
!       It uses a local condition handler to fix up errors.
!
!
! CALLING SEQUENCE:
!       READ_NEW_HEADER (ARG1)
!
! INPUT PARAMETERS:
!       ARG1: LBN of block to read
!
! IMPLICIT INPUTS:
!       NONE
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! ROUTINE VALUE:
!       address of buffer containing block or 0 if bad
!
! SIDE EFFECTS:
!       block read and/or written
!
!--

BEGIN

LOCAL
        HEADER          : REF BBLOCK;   ! address of block read

BASE_REGISTER;

EXTERNAL ROUTINE
        READ_BLOCK      : L_NORM,       ! read a block
        WRITE_BLOCK     : L_NORM,       ! write a block
        INVALIDATE      : L_NORM,       ! invalidate a buffer
        CREATE_BLOCK    : L_NORM;       ! create a new block buffer

! Under control of the condition handler, we read the block. If the read
! fails, we attempt to rewrite the block and then read it again. If either
! of the latter fails, we return failure.
!

ENABLE HANDLER;

HEADER = READ_BLOCK (.LBN, 1, HEADER_TYPE);

IF .HEADER EQL 0
THEN
```

```
;   720          1706  3          BEGIN
;   721          1707  3          HEADER = CREATE_BLOCK (.LBN, 1, HEADER_TYPE);
;   722          1708  3          (.HEADER)<0,32> = 1;
;   723          1709  3          WRITE_BLOCK (.HEADER);
;   724          1710  3          INVALIDATE (.HEADER);
;   725          1711  3          HEADER = READ_BLOCK (.LBN, 1, HEADER_TYPE);
;   726          1712  2          END;
;   727          1713  2
;   728          1714  2  RETURN .HEADER;
;   729          1715  2
;   730          1716  1  END;                                        ! end of routine READ_NEW_HEADER



                                           0004 00000 READ_NEW_HEADER:
                                                              .WORD   Save R2                                      ; 1649
                        6D      0042    CF   DE 00002         MOVAL   2$, (FP)                                     ; 1682
                        7E              01   7D 00007         MOVQ    #1, -(SP)                                    ; 1702
                                  04    AC   DD 0000A         PUSHL   LBN                                          ;
                 0000G  CF              03   FB 0000D         CALLS   #3, READ_BLOCK                               ;
                        52              50   D0 00012         MOVL    R0, HEADER                                   ;
                                        2D   12 00015         BNEQ    1$                                           ; 1704
                        7E              01   7D 00017         MOVQ    #1, -(SP)                                    ; 1707
                                  04    AC   DD 0001A         PUSHL   LBN                                          ;
                 0000G  CF              03   FB 0001D         CALLS   #3, CREATE_BLOCK                             ;
                        52              50   D0 00022         MOVL    R0, HEADER                                   ;
                        62              01   D0 00025         MOVL    #1, (HEADER)                                 ; 1708
                                        52   DD 00028         PUSHL   HEADER                                       ; 1709
                 0000G  CF              01   FB 0002A         CALLS   #1, WRITE_BLOCK                              ;
                                        52   DD 0002F         PUSHL   HEADER                                       ; 1710
                 0000G  CF              01   FB 00031         CALLS   #1, INVALIDATE                               ;
                        7E              01   7D 00036         MOVQ    #1, -(SP)                                    ; 1711
                                  04    AC   DD 00039         PUSHL   LBN                                          ;
                 0000G  CF              03   FB 0003C         CALLS   #3, READ_BLOCK                               ;
                        52              50   D0 00041         MOVL    R0, HEADER                                   ;
                        50              52   D0 00044 1$:     MOVL    HEADER, R0                                   ; 1714
                                             04 00047         RET                                                  ; 1716
                                           0000 00048 2$:     .WORD   Save nothing                                 ; 1682
                                        7E   D4 0004A         CLRL    -(SP)                                        ;
                                        5E   DD 0004C         PUSHL   SP                                           ;
                        7E        04    AC   7D 0004E         MOVQ    4(AP), -(SP)                                 ;
                 0000V  CF              03   FB 00052         CALLS   #3, HANDLER                                  ;
                                             04 00057         RET                                                  ;

; Routine Size:  88 bytes,    Routine Base:  $CODE$ + 038A
```

```
1 ROUTINE HANDLER (SIGNAL, MECHANISM) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is the condition handler for the initial header read.
1 !       On surface errors, it unwinds and causes a return of 0 to the caller
1 !       of the I/O routine to indicate error. Hard drive errors cause the
1 !       usual error exit.
1 !
1 ! CALLING SEQUENCE:
1 !       HANDLER (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of signal array
1 !       ARG2: address of mechanism array
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       SS$_RESIGNAL or none if unwind
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
1
2 BEGIN
2
2 MAP
2       SIGNAL          : REF BBLOCK,   ! signal arg array
2       MECHANISM       : REF BBLOCK;   ! mechanism arg array
2
2
2 ! If the condition is change mode to user (error exit) and the status is
2 ! read error, zero the return R0 and unwind to the the establisher. On
2 ! most write errors, zero the return R0 and unwind to the caller.
2 ! Otherwise, just resignal the condition.
2 !
2
2 IF .SIGNAL[CHF$L_SIG_NAME] EQL SS$_CMODUSER
2 THEN
3     BEGIN
3     MECHANISM[CHF$L_MCH_SAVR0] = 0;
3
3     IF SURFACE_ERROR (.SIGNAL[CHF$L_SIG_ARG1])
3     THEN
4         $UNWIND (DEPADR = MECHANISM[CHF$L_MCH_DEPTH])
```

```
;  789          1774  2       END;
;  790          1775  2
;  791          1776  2 RETURN SS$_RESIGNAL;                       ! status is irrelevant if unwinding
;  792          1777  2
;  793          1778  1 END;                                       ! end of routine HANDLER


                                                                        .EXTRN  SYS$UNWIND

                                              0000 00000 HANDLER:.WORD  Save nothing                                    ; 1717
                        50         04     AC   D0 00002         MOVL    SIGNAL, R0                                      ; 1766
              00000424  8F         04     A0   D1 00006         CMPL    4(R0), #1060                                    ;
                                          41   12 0000E         BNEQ    2$                                              ;
                        50         08     AC   D0 00010         MOVL    MECHANISM, R0                                   ; 1769
                                   0C     A0   D4 00014         CLRL    12(R0)                                          ;
                        50         04     AC   D0 00017         MOVL    SIGNAL, R0                                      ; 1771
              000001F4  8F         08     A0   D1 0001B         CMPL    8(R0), #500                                     ;
                                          1E   13 00023         BEQL    1$                                              ;
              0000005C  8F         08     A0   D1 00025         CMPL    8(R0), #92                                      ;
                                          14   13 0002D         BEQL    1$                                              ;
              000000BC  8F         08     A0   D1 0002F         CMPL    8(R0), #188                                     ;
                                          0A   13 00037         BEQL    1$                                              ;
              00002144  8F         08     A0   D1 00039         CMPL    8(R0), #8516                                    ;
                                          0E   12 00041         BNEQ    2$                                              ;
                                          7E   D4 00043 1$:     CLRL    -(SP)                                           ; 1773
           7E        08 AC                08   C1 00045         ADDL3   #8, MECHANISM, -(SP)                            ;
              00000000G 00                02   FB 0004A         CALLS   #2, SYS$UNWIND                                  ;
                        50       0918     8F   3C 00051 2$:     MOVZWL  #2328, R0                                       ; 1776
                                               04 00056         RET                                                     ; 1778

; Routine Size:  87 bytes,    Routine Base:  $CODE$ + 03E2
```

```
1 GLOBAL ROUTINE READ_IDX_HEADER : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine reads the volume's index file header, using the
1 !       alternate if it seems appropriate.
1 !
1 ! CALLING SEQUENCE:
1 !       READ_IDX_HEADER ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       address of file header read
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2
2
2 LOCAL
2         HEADER          : REF BBLOCK,   ! address of header read
2         FCB             : REF BBLOCK;   ! address of index file FCB
2
2 BIND_COMMON;
2
2 EXTERNAL ROUTINE
2         FILE_SIZE       : L_NORM,       ! compute file header file size
2         READ_HEADER     : L_NORM,       ! read file header
2         READ_BLOCK      : L_NORM,       ! read a disk block
2         CHECK_HEADER2   : L_NORM,       ! validate file header
2         RESET_LBN       : L_NORM,       ! reassign LBN of buffer
2         INVALIDATE      : L_NORM;       ! invalidate buffer
2
2
2 ! Read the index file header. Check the file size against the
2 ! file size in the FCB. A mismatch indicates a failure in writing the
2 ! header the last time; if this occurs, try the alternate header instead.
2 !
2
2 SAVE_STATUS = .USER_STATUS;
2
```

```
;  852          1836  2 FCB = .CURRENT_VCB[VCB$L_FCBFL];
;  853          1837  2 HEADER = READ_HEADER (0, .FCB);
;  854          1838  2 IF FILE_SIZE (.HEADER) LSSU .FCB[FCB$L_FILESIZE]
;  855          1839  2 THEN
;  856          1840  3     BEGIN
;  857          1841  3     FILE_HEADER = 0;
;  858          1842  3     INVALIDATE (.HEADER);
;  859          1843  3     HEADER = READ_BLOCK (.CURRENT_VCB[VCB$L_IXHDR2LBN], 1, HEADER_TYPE);
;  860          1844  3     IF NOT CHECK_HEADER2 (.HEADER, UPLIT WORD (FID$C_INDEXF, FID$C_INDEXF, 0))
;  861          1845  3     THEN
;  862          1846  4         BEGIN
;  863          1847  4         INVALIDATE (.HEADER);
;  864          1848  4         ERR_EXIT (0);
;  865          1849  3         END;
;  866          1850  3     IF FILE_SIZE (.HEADER) LSSU .FCB[FCB$L_FILESIZE]
;  867          1851  3     THEN ERR_EXIT (SS$_BADFILEHDR);
;  868          1852  3     FILE_HEADER = .HEADER;
;  869          1853  3     RESET_LBN (.HEADER, .FCB[FCB$L_HDLBN]);
;  870          1854  2     END;
;  871          1855  2
;  872          1856  2 USER_STATUS = .SAVE_STATUS;
;  873          1857  2
;  874          1858  2 .HEADER
;  875          1859  1 END;                                   ! end of routine READ_IDX_HEADER


                                          00439           .BLKB    1
                    0000  0001  0001  0043A P.AAA:    .WORD    1, 1, 0                                          ;

                                                      .EXTRN   FILE_SIZE, READ_HEADER

                                     000C 00000           .ENTRY   READ_IDX_HEADER, Save R2,R3                      ; 1779
              C0   AA       80    AA   D0 00002           MOVL     -128(BASE), -64(BASE)                            ; 1834
                   52       98    BA   D0 00007           MOVL     @-104(BASE), FCB                                 ; 1836
                                  52   DD 0000B           PUSHL    FCB                                              ; 1837
                                  7E   D4 0000D           CLRL     -(SP)                                            ;
           0000G   CF             02   FB 0000F           CALLS    #2, READ_HEADER                                  ;
                   53             50   D0 00014           MOVL     R0, HEADER                                       ;
                                  53   DD 00017           PUSHL    HEADER                                           ; 1838
           0000G   CF             01   FB 00019           CALLS    #1, FILE_SIZE                                    ;
              38   A2             50   D1 0001E           CMPL     R0, 56(FCB)                                      ;
                                  53   1E 00022           BGEQU    3$                                               ;
                            04    AA   D4 00024           CLRL     4(BASE)                                          ; 1841
                                  53   DD 00027           PUSHL    HEADER                                           ; 1842
           0000G   CF             01   FB 00029           CALLS    #1, INVALIDATE                                   ;
                   7E             01   7D 0002E           MOVQ     #1, -(SP)                                        ; 1843
                   50       98    AA   D0 00031           MOVL     -104(BASE), R0                                   ;
                            2C    A0   DD 00035           PUSHL    44(R0)                                           ;
           0000G   CF             03   FB 00038           CALLS    #3, READ_BLOCK                                   ;
                   53             50   D0 0003D           MOVL     R0, HEADER                                       ;
                            B7    AF   9F 00040           PUSHAB   P.AAA                                            ; 1844
                                  53   DD 00043           PUSHL    HEADER                                           ;
           0000G   CF             02   FB 00045           CALLS    #2, CHECK_HEADER2                                ;
                   0A             50   E8 0004A           BLBS     R0, 1$                                           ;
                                  53   DD 0004D           PUSHL    HEADER                                           ; 1847
           0000G   CF             01   FB 0004F           CALLS    #1, INVALIDATE                                   ;
```

```
                                       00  BF 00054         CHMU    #0                                    ; 1848
                                           04 00056         RET                                           ;
                                       53  DD 00C57 1$:     PUSHL   HEADER                                ; 1850
                 0000G   CF            01  FB 00059         CALLS   #1, FILE_SIZE                         ;
                    38   A2            50  D1 0005E         CMPL    R0, 56(FCB)                           ;
                                       05  1E 00062         BGEQU   2$                                    ;
                               0810    8F  BF 00064         CHMU    #2064                                 ; 1851
                                           04 00068         RET                                           ;
                    04   AA            53  D0 00069 2$:     MOVL    HEADER, 4(BASE)                       ; 1852
                                 34    A2  DD 0006D         PUSHL   52(FCB)                               ; 1853
                                       53  DD 00070         PUSHL   HEADER                                ;
                 0000G   CF            02  FB 00072         CALLS   #2, RESET_LBN                         ;
                    80   AA      C0    AA  D0 00077 3$:     MOVL    -64(BASE), -128(BASE)                 ; 1856
                         50            53  D0 0007C         MOVL    HEADER, R0                            ; 1859
                                           04 0007F         RET                                           ;

; Routine Size:  128 bytes,    Routine Base:  $CODE$ + 0440


;  876          1860  1
```

```
1 GLOBAL ROUTINE MAP_IDX (VBN, COUNT) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine maps a virtual block in the index file.
1 !
1 ! CALLING SEQUENCE:
1 !       MAP_IDX (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: VBN of block to map
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       COUNT: (optional) address to store count of contiguous blocks
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       LBN of blocks mapped or -1 if failure
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2
2 EXTERNAL ROUTINE
2       MAP_VBN         : L_NORM,       ! map VBN and turn window if necessary
2       MAP_WINDOW      : L_NORM,       ! map VBN with current window
2       RELEASE_SERIAL_LOCK : L_NORM,   ! release sync lock on file
2       SERIAL_FILE     : L_NORM;       ! get sync lock on file
2
2 LOCAL
2       INCOMPLETE_FLAG,                ! Saved state of CLF_INCOMPLETE
2       IDX_FCB         : REF BBLOCK,   ! address of index file FCB
2       LBN,                            ! resulting LBN from map
2       UNMAPPED,                       ! received count of unmapped blocks
2       TEMP;                           ! dummy to store resulting UCB
2
2 BIND_COMMON;
2
2 ! Try to map with the existing window first. This can be done without
2 ! taking out the sync lock on the index file.
2 !
2
2 IDX_FCB = .CURRENT_VCB [VCB$L_FCBFL];
2
3 IF (LBN = MAP_WINDOW (.VBN, .IDX_FCB [FCB$L_WLFL], 1000, UNMAPPED, TEMP))
2     EQL -1
2 THEN
```

```
;  935        1918 3          BEGIN
;  936        1919 3          TEMP = .CURR_LCKINDX;
;  937        1920 3          SERIAL_FILE (IDX_FCB [FCB$W_FID]);
;  938        1921 3          INCOMPLETE_FLAG = .CLEANUP_FLAGS[CLF_INCOMPLETE];    ! Save current state
;  939        1922 3          IDX_FCB [FCB$V_STALE] = 1;
;  940        1923 3          LBN = MAP_VBN (.VBN, .IDX_FCB [FCB$L_WLFL], 1000, UNMAPPED);
;  941        1924 3          CLEANUP_FLAGS[CLF_INCOMPLETE] = .INCOMPLETE_FLAG;    ! Restore saved state
;  942        1925 3
;  943        1926 3          IF .TEMP NEQ .CURR_LCKINDX
;  944        1927 3          THEN
;  945        1928 4              BEGIN
;  946        1929 4              RELEASE_SERIAL_LOCK (.CURR_LCKINDX);
;  947        1930 4              CURR_LCKINDX = .TEMP;
;  948        1931 3              END;
;  949        1932 3
;  950        1933 2          END;
;  951        1934 2
;  952        1935 2      ! Return the block count if asked for.
;  953        1936 2      !
;  954        1937 2
;  955        1938 2      IF ACTUALCOUNT GEQU 2
;  956        1939 2      THEN .COUNT = 1000 - .UNMAPPED;
;  957        1940 2      .LBN
;  958        1941 2
;  959        1942 1      END;                                    ! of routine MAP_IDX


                                                                .EXTRN  MAP_VBN, MAP_WINDOW

                                                  001C 00000    .ENTRY  MAP_IDX, Save R2,R3,R4                         ; 1861
                                 5E          08    C2 00002     SUBL2   #8, SP                                         ;
                                 52     98   BA    D0 00005     MOVL    @-104(BASE), IDX_FCB                           ; 1913
                                             5E    DD 00009     PUSHL   SP                                             ; 1915
                                        08   AE    9F 0000B     PUSHAB  UNMAPPED                                       ;
                                 7E   03E8   8F    3C 0000E     MOVZWL  #1000, -(SP)                                   ;
                                        10   A2    DD 00013     PUSHL   16(IDX_FCB)                                    ;
                                        04   AC    DD 00016     PUSHL   VBN                                            ;
                          0000G  CF          05    FB 00019     CALLS   #5, MAP_WINDOW                                 ;
                                 54          50    D0 0001E     MOVL    R0, LBN                                        ;
                       FFFFFFFF  8F          54    D1 00021     CMPL    LBN, #-1                                       ; 1916
                                             42    12 00028     BNEQ    1$                                             ;
                                 6E     14   AA    D0 0002A     MOVL    20(BASE), TEMP                                 ; 1919
                                        24   A2    9F 0002E     PUSHAB  36(IDX_FCB)                                    ; 1920
                          0000G  CF          01    FB 00031     CALLS   #1, SERIAL_FILE                                ;
          53           6A        01          0A    EF 00036     EXTZV   #10, #1, (BASE), INCOMPLETE_FLAG               ; 1921
                            23   A2          01    88 0003B     BISB2   #1, 35(IDX_FCB)                                ; 1922
                                        04   AE    9F 0003F     PUSHAB  UNMAPPED                                       ; 1923
                                 7E   03E8   8F    3C 00042     MOVZWL  #1000, -(SP)                                   ;
                                        10   A2    DD 00047     PUSHL   16(IDX_FCB)                                    ;
                                        04   AC    DD 0004A     PUSHL   VBN                                            ;
                          0000G  CF          04    FB 0004D     CALLS   #4, MAP_VBN                                    ;
                                 54          50    D0 00052     MOVL    R0, LBN                                        ;
          6A           01        0A          53    F0 00055     INSV    INCOMPLETE_FLAG, #10, #1, (BASE)               ; 1924
                            14   AA          6E    D1 0005A     CMPL    TEMP, 20(BASE)                                 ; 1926
                                             0C    13 0005E     BEQL    1$                                             ;
                                        14   AA    DD 00060     PUSHL   20(BASE)                                       ; 1929
```

```
                                  0000G  CF            01  FB 00063         CALLS    #1, RELEASE_SERIAL_LOCK                         ;
                                     14  AA            6E  D0 00068         MOVL     TEMP, 20(BASE)                                  ; 1930
                                         02            6C  91 0006C 1$:     CMPB     (AP), #2                                        ; 1938
                                                       0A  1F 0006F         BLSSU    2$                                              ;
                      08    3C 000003E8  8F        04  AE  C3 00071         SUBL3    UNMAPPED, #1000, @COUNT                         ; 1939
                                         50            54  D0 0007B 2$:     MOVL     LBN, R0                                         ; 1942
                                                           04 0007E         RET                                                      ;

; Routine Size:  127 bytes,    Routine Base:  $CODE$ + 04C0


;  960          1943  1
;  961          1944  1 END
;  962          1945  0 ELUDOM




;                                        PSECT SUMMARY
;
;       Name                         Bytes                         Attributes
;
;  $CODE$                                1343  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                               Library Statistics
;
;                                                  -------- Symbols --------       Pages        Processing
;       File                                       Total    Loaded   Percent       Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1                   18619        67         0       1000           00:02.0




;                                         COMMAND QUALIFIERS

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

; Size:          1336 code + 7 data bytes
; Run Time:         01:03.7
; Elapsed Time:     02:03.8
; Lines/CPU Min:     1832
; Lexemes/CPU-Min: 55644
; Memory Used:  336 pages
; Compilation Complete
```

0169 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY

DEACCS
LIS

DELETE
LIS

DIRSCN
LIS

DISPAT
LIS

CREHDR
LIS

DIRACC
LIS

DELBAD
LIS

CREFCB
LIS

CREWIN
LIS

DISPATCH
LIS

ENTER
LIS

DELFIL
LIS