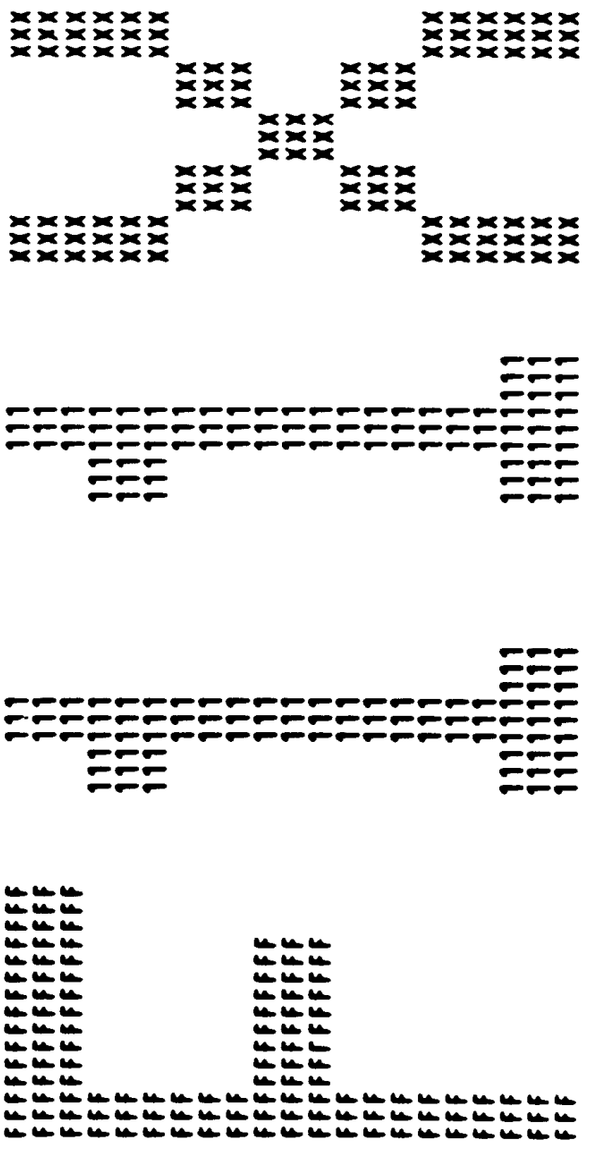

```
  CCCCCCCC  RRRRRRRR    EEEEEEEEEE  HH      HH  DDDDDDDD    RRRRRRRR
  CCCCCCCC  RRRRRRRR    EEEEEEEEEE  HH      HH  DDDDDDDD    RRRRRRRR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RR      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
```

```
;     1          0001  0 MODULE CREHDR (
;     2          0002  0                 LANGUAGE (BLISS32),
;001  ! CDS0018  0003  0                 IDENT = 'V04-001'
;     4-1        0004  0                 ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !*****************************************************************************
;     9          0009  1 !*                                                                           *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                      *
;    13          0013  1 !*                                                                           *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    15          0015  1 !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;    18          0018  1 !*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY         *
;    19          0019  1 !*  TRANSFERRED.                                                              *
;    20          0020  1 !*                                                                           *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
;    22          0022  1 !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
;    23          0023  1 !*  CORPORATION.                                                              *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*****************************************************************************
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:  F11ACP Structure Level 2
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This routine creates a new file ID by allocating a file number from the
;    38          0038  1 !       index file bitmap. It returns an empty file header, verified for use.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT:
;    41          0041  1 !
;    42          0042  1 !       STARLET operating system, including privileged system services
;    43          0043  1 !       and internal exec routines.
;    44          0044  1 !
;    45          0045  1 !--
;    46          0046  1 !
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  28-Mar-1977  13:49
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;001  ! CDS0018  0052  1 !       V04-001 CDS0018         Christian D. Saether    15-Nov-1984
;002  ! CDS0018  0053  1 !               Expand test for clusterness to look at clu$gl_club.
;003  ! CDS0018  0054  1 !
;    52          0055  1 !       V03-022 CDS0017         Christian D. Saether    20-Aug-1984
;    53          0056  1 !               Force fcb for indexf to be stale always before
;    54          0057  1 !               attempting to map vbns.
```

```
;    55          0058  1 ;
;    56          0059  1 ;       V03-021 CDS0016         Christian D. Saether    13-Aug-1984
;    57          0060  1 ;               Back off an extra dot in ACG0438.
;    58          0061  1 ;
;    59          0062  1 ;       V03-020 ACG0438         Andrew C. Goldstein,    1-Aug-1984  11:55
;    60          0063  1 ;               Add cache interlock logic on FID cache; use central
;    61          0064  1 ;               dequeue routine.
;    62          0065  1 ;
;    63          0066  1 ;       V03-019 LMP0278         L. Mark Pilant,         12-Jul-1984  10:58
;    64          0067  1 ;               Fix a bug that caused the EXBYTLM error if it was necessary
;    65          0068  1 ;               to turn the index file window.
;    66          0069  1 ;
;    67          0070  1 ;       V03-018 CDS0015         Christian D. Saether    17-Apr-1984
;    68          0071  1 ;               Have MAP_IDX check to see whether curr_lckindx is
;    69          0072  1 ;               for the index file to avoid releasing it if so.
;    70          0073  1 ;
;    71          0074  1 ;       V03-017 CDS0014         Christian D. Saether    11-Apr-1984
;    72          0075  1 ;               Release allocation lock prior to serializing on
;    73          0076  1 ;               new primary header.  This eliminates potential
;    74          0077  1 ;               deadlocks when the new primary header is a valid
;    75          0078  1 ;               header that someone else is messing with.
;    76          0079  1 ;
;    77          0080  1 ;       V03-016 CDS0013         Christian D. Saether     1-Apr-1984
;    78          0081  1 ;               ACG0409 forgot to rewrite indexf bitmap buffer.  No joke.
;    79          0082  1 ;
;    80          0083  1 ;       V03-015 ACG0409         Andrew C. Goldstein,    21-Mar-1984  19:40
;    81          0084  1 ;               Redesign file ID cacheing algorithm so that file ID's
;    82          0085  1 ;               beyond the index file EOF are not cached. Eliminate
;    83          0086  1 ;               BASH_HEADERS routine; general code cleanup to remove
;    84          0087  1 ;               kernel calls. CHECK_HEADER2 no longer writes USER_STATUS.
;    85          0088  1 ;
;    86          0089  1 ;       V03-014 ACG0404         Andrew C. Goldstein,    15-Mar-1984  17:37
;    87          0090  1 ;               Correct releasing of file sync lock when retrying for a header
;    88          0091  1 ;
;    89          0092  1 ;       V03-013 CDS0012         Christian D. Saether    23-Feb-1984
;    90          0093  1 ;               Eliminate references to FLUSH_LOCK_BASIS.
;    91          0094  1 ;
;    92          0095  1 ;       V03-012 CDS0011         Christian D. Saether    27-Dec-1983
;    93          0096  1 ;               Use BIND_COMMON macro.
;    94          0097  1 ;
;    95          0098  1 ;       V03-011 CDS0010         Christian D. Saether    12-Dec-1983
;    96          0099  1 ;               Start of XQP code is at symbol INITXQP now.
;    97          0100  1 ;
;    98          0101  1 ;       V03-010 CDS0009         Christian D. Saether     5-Oct-1983
;    99          0102  1 ;               Fix bug restoring privileges to the PCB.
;   100          0103  1 ;
;   101          0104  1 ;       V03-009 CDS0008         Christian D. Saether     3-Oct-1983
;   102          0105  1 ;               Save/restore CURR_LCKINDX where necessary rather
;   103          0106  1 ;               than PRIM_LCKINDX.
;   104          0107  1 ;
;   105          0108  1 ;       V03-008 CDS0007         Christian D. Saether    13-Sep-1983
;   106          0109  1 ;               Modify interface to allocation serialization.
;   107          0110  1 ;
;   108          0111  1 ;       V03-007 CDS0006         Christian D. Saether    12-May-1983
;   109          0112  1 ;               Serialize header creation.
;   110          0113  1 ;
;   111          0114  1 ;       V03-006 CDS0005         Christian D. Saether     1-Mar-1983
```

```
;  112          0115  1 !                 Need BYPASS privilege also.
;  113          0116  1 !
;  114          0117  1 !       V03-005 CDS0004         Christian D. Saether    20-Feb-1983
;  115          0118  1 !               Call MAP_VBN before checking FILESIZE so that
;  116          0119  1 !               header is checked before deciding to extend
;  117          0120  1 !               index file.
;  118          0121  1 !               Also make READ_IDX_HEADER insensitive to headers that
;  119          0122  1 !               map more than the FCB knows about.
;  120          0123  1 !               Totally punt figuring out what to do with EFBLK
;  121          0124  1 !               for the index file.
;  122          0125  1 !
;  123          0126  1 !       V03-004 CDS0003         Christian D. Saether    13-Jan-1983
;  124          0127  1 !               Separately save and restore PHD privs.
;  125          0128  1 !
;  126          0129  1 !       V03-003 CDS0002         Christian D. Saether    28-Dec-1982
;  127          0130  1 !               Give priv around QIO.
;  128          0131  1 !
;  129          0132  1 !       V03-002 CDS0001         C Saether                3-Aug-1982
;  130          0133  1 !               Change QIOW to QIO with completion AST.
;  131          0134  1 !
;  132          0135  1 !       V03-001 ACG0273         Andrew C. Goldstein,    23-Mar-1982  10:50
;  133          0136  1 !               Use random file sequence number if old header is junk,
;  134          0137  1 !               use alternate index file header if primary is suspect
;  135          0138  1 !
;  136          0139  1 !       V02-007 ACG0229         Andrew C. Goldstein,    23-Dec-1981  21:53
;  137          0140  1 !               Count file ID cache hits and misses
;  138          0141  1 !
;  139          0142  1 !       V02-006 ACG0167         Andrew C. Goldstein,    16-Apr-1980  19:25
;  140          0143  1 !               Previous revision history moved to F11B.REV
;  141          0144  1 !--
;  142          0145  1
;  143          0146  1
;  144          0147  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;  145          0148  1 REQUIRE 'SRC$:FCPDEF.B32';
;  146          1139  1
;  147          1140  1
;  148          1141  1 FORWARD ROUTINE
;  149          1142  1         CREATE_HEADER    : L_NORM,          ! create file ID and header
;  150          1143  1         FILL_FID_CACHE   : L_NORM NOVALUE,  ! load file ID cache from bitmap
;  151          1144  1         INIT_FID_CACHE   : L_NORM NOVALUE,  ! initialize file ID cache lock
;  152          1145  1         READ_NEW_HEADER  : L_NORM,          ! read new file header block
;  153          1146  1         HANDLER,                            ! local condition handler
;  154          1147  1         READ_IDX_HEADER  : L_NORM,          ! read index file header
;  155          1148  1         MAP_IDX          : L_NORM;          ! map vbn for index file.
```

```
;  157     1149  1 GLOBAL ROUTINE CREATE_HEADER (FILE_ID) : L_NORM =
;  158     1150  1
;  159     1151  1 !++
;  160     1152  1 !
;  161     1153  1 ! FUNCTIONAL DESCRIPTION:
;  162     1154  1 !
;  163     1155  1 !       This routine creates a new file ID by searching the volume's index
;  164     1156  1 !       file bitmap for the first free file number. It also checks that a
;  165     1157  1 !       header for the file number is present in the index file. It reads
;  166     1158  1 !       the old header and establishes the file sequence number for the
;  167     1159  1 !       new one.
;  168     1160  1 !
;  169     1161  1 ! CALLING SEQUENCE:
;  170     1162  1 !       CREATE_HEADER (ARG1)
;  171     1163  1 !
;  172     1164  1 ! INPUT PARAMETERS:
;  173     1165  1 !       NONE
;  174     1166  1 !
;  175     1167  1 ! IMPLICIT INPUTS:
;  176     1168  1 !       CURRENT_VCB: address of volume's VCB
;  177     1169  1 !
;  178     1170  1 ! OUTPUT PARAMETERS:
;  179     1171  1 !       ARG1: address to store file ID of created header
;  180     1172  1 !
;  181     1173  1 ! IMPLICIT OUTPUTS:
;  182     1174  1 !       NEW_FID: file number of header created
;  183     1175  1 !       NEW_FID_RVN: RVN of above
;  184     1176  1 !
;  185     1177  1 ! ROUTINE VALUE:
;  186     1178  1 !       address of buffer containing new header
;  187     1179  1 !
;  188     1180  1 ! SIDE EFFECTS:
;  189     1181  1 !       VCB and index file bitmap altered, header block read
;  190     1182  1 !
;  191     1183  1 !--
;  192     1184  1
;  193     1185  2 BEGIN
;  194     1186  2
;  195     1187  2 MAP
;  196     1188  2         FILE_ID         : REF BBLOCK;   ! new file ID of header
;  197     1189  2
;  198     1190  2 LABEL
;  199     1191  2         GET_FILE_NUM;                   ! acquire a file number
;  200     1192  2
;  201     1193  2 LOCAL
;  202     1194  2         CACHE_FLUSHED,                  ! flag indicating cluster caches flushed
;  203     1195  2         NEW_LCKINDX     : INITIAL (0),
;  204     1196  2         TEMP,                           ! temp storage for current lock index
;  205     1197  2         VCB             : REF BBLOCK,   ! local copy of VCB address
;  206     1198  2         FID_CACHE       : REF BBLOCK,   ! pointer to file ID cache
;  207     1199  2         VBN,                            ! relative block number in bitmap
;  208     1200  2         BUFFER          : REF BITVECTOR,! address of index file bitmap buffer
;  209     1201  2         ADDRESS         : REF BITVECTOR,! address of byte in buffer
;  210     1202  2         CURRENT_EOF,                    ! current EOF of index file
;  211     1203  2         COUNT,                          ! number of index blocks to bash
;  212     1204  2         FILE_NUMBER,                    ! file number allocated
;  213     1205  2         IDX_FCB         : REF BBLOCK,   ! FCB of index file
```

```
;  214          1206  2          LBN,                            ! LBN of new file header
;  215          1207  2          HEADER          : REF BBLOCK,   ! address of header buffer
;  216          1208  2          STATUS;                         ! value of CHECK_HEADER call
;  217          1209  2
;  218          1210  2 EXTERNAL
;001 !CDS0018   1211  2          CLU$GL_CLUB     : ADDRESSING_MODE (GENERAL),
;  219          1212  2          PMS$GL_FIDHIT   : ADDRESSING_MODE (GENERAL),
;  220          1213  2                                          ! count of file ID cache hits
;  221          1214  2          PMS$GL_FIDMISS  : ADDRESSING_MODE (GENERAL),
;  222          1215  2                                          ! count of file ID cache misses
;  223          1216  2          EXE$GQ_SYSTIME  : ADDRESSING_MODE (GENERAL);
;  224          1217  2                                          ! system time of day
;  225          1218  2 BIND_COMMON;
;  226          1219  2
;  227          1220  2 EXTERNAL ROUTINE
;  228          1221  2          ALLOCATION_LOCK : L_NORM NOVALUE, ! interlock allocation
;  229          1222  2          ALLOCATION_UNLOCK : L_NORM NOVALUE, ! release allocation lock.
;  230          1223  2          SERIAL_FILE     : L_NORM,       ! serialize file processing
;  231          1224  2          RELEASE_SERIAL_LOCK : L_NORM NOVALUE, ! release processing lock
;  232          1225  2          DEQ_LOCK        : L_NORM,       ! dequeue a lock
;  233          1226  2          READ_BLOCK      : L_NORM,       ! read block from disk
;  234          1227  2          WRITE_BLOCK     : L_NORM,       ! write block to disk
;  235          1228  2          DELETE_FID      : L_NORM,       ! flush file ID cache and release lock
;  236          1229  2          RELEASE_LOCKBASIS : L_NORM,     ! release buffers under specified lock
;  237          1230  2          CACHE_LOCK      : L_NORM,       ! acquire cache sync lock
;  238          1231  2          EXTEND_INDEX    : L_NORM,       ! extend the index file
;  239          1232  2          ERASE_BLOCKS    : L_NORM,       ! erase blocks on disk
;  240          1233  2          CHECKSUM        : L_NORM,       ! compute file header checksum
;  241          1234  2          WRITE_HEADER    : L_NORM,       ! write current file header
;  242          1235  2          RESET_LBN       : L_NORM,       ! change backing LBN of buffer
;  243          1236  2          INVALIDATE      : L_NORM,       ! invalidate a buffer
;  244          1237  2          CREATE_BLOCK    : L_NORM,       ! materialize a block buffer
;  245          1238  2          CHECK_HEADER2   : L_NORM,       ! verify file header
;  246          1239  2          MARK_DIRTY      : L_NORM;       ! mark buffer for write-back
;  247          1240  2
;  248          1241  2 ! Serialize further file header creation processing.
;  249          1242  2 !
;  250          1243  2
;  251          1244  2 ALLOCATION_LOCK ();
;  252          1245  2
;  253          1246  2 ! The outer loop performs retries if blocks in the index file are bad or
;  254          1247  2 ! are valid file headers. A block containing a valid file header is never
;  255          1248  2 ! used to create a new file; it is simply left marked in use for recovery.
;  256          1249  2 ! Bad header blocks are simply left marked in use in the index file bitmap;
;  257          1250  2 ! they will show up in a verify but are otherwise harmless.
;  258          1251  2 !
;  259          1252  2
;  260          1253  2 VCB = .CURRENT_VCB;
;  261          1254  2 FID_CACHE = .BBLOCK [.VCB[VCB$L_CACHE], VCA$L_FIDCACHE];
;  262          1255  2 CACHE_FLUSHED = 0;
;  263          1256  2 WHILE 1 DO
;  264          1257  3     GET_FILE_NUM: BEGIN
;  265          1258  3
;  266          1259  3 ! See if a file number is available in the file number cache. If not,
;  267          1260  3 ! we scan the index file bitmap for the first free (zero) bit. This is done
;  268          1261  3 ! by starting with the block recorded in the VCB and looking at each block
;  269          1262  3 ! with a character scan.
```

```
;  270          1263  3 !
;  271          1264  3
;  272          1265  3     IF .FID_CACHE[VCA$W_FIDCOUNT] EQL 0
;  273          1266  3     THEN
;  274          1267  4         BEGIN
;  275          1268  4         PMS$GL_FIDMISS = .PMS$GL_FIDMISS + 1;
;  276          1269  4         VBN = .VCB[VCB$B_IBMAPVBN];
;  277          1270  4
;  278          1271  4         IF NOT
;  279          1272  5             BEGIN
;  280          1273  5             UNTIL .VBN GEQ .VCB[VCB$B_IBMAPSIZE] DO
;  281          1274  6                 BEGIN
;  282          1275  6                 BUFFER = READ_BLOCK (.VBN + .VCB[VCB$L_IBMAPLBN], 1, INDEX_TYPE);
;  283          1276  6                 IF NOT CH$FAIL (ADDRESS = CH$FIND_NOT_CH (512, .BUFFER, 255))
;  284          1277  6                 THEN EXITLOOP 0;
;  285          1278  6                 VBN = .VBN + 1;
;  286          1279  6                 END
;  287          1280  5             END
;  288          1281  5
;  289          1282  5 ! Having found a bitmap block with free files in it, attempt to fill the
;  290          1283  5 ! file ID cache. If it refuses to fill, it's because we're at the index
;  291          1284  5 ! file EOF.
;  292          1285  5 !
;  293          1286  5
;  294          1287  4         THEN FILL_FID_CACHE (.VCB, .BUFFER, .VBN);
;  295          1288  4         IF .FID_CACHE[VCA$W_FIDCOUNT] EQL 0
;  296          1289  4         THEN
;  297          1290  5             BEGIN
;  298          1291  5
;  299          1292  5 ! If the index file EOF coincides with the physical end of file, we have to
;  300          1293  5 ! extend the index file. Otherwise, we just have to push the EOF. Before
;  301          1294  5 ! extending the index file, if we are in a cluster, ask for a cluster-wide
;  302          1295  5 ! flush of the file ID caches.
;  303          1296  5 !
;  304          1297  5
;  305          1298  5             IDX_FCB = .VCB[VCB$L_FCBFL];
;  306          1299  5             CURRENT_EOF = .IDX_FCB[FCB$L_EFBLK];
;  307          1300  5             IF .CURRENT_EOF GEQU .IDX_FCB[FCB$L_FILESIZE]
;  308          1301  5             THEN
;  309          1302  6                 BEGIN
;001  !CDS0018   1303  6                 IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
;002  !CDS0018   1304  6                 AND .CLU$GL_CLUB NEQ 0
;  311-1        1305  6                 AND NOT .CACHE_FLUSHED
;  312          1306  6                 THEN
;  313          1307  7                     BEGIN
;  314          1308  7                     LOCAL IDX_FILE_ID, LOCK_ID;
;  315          1309  7                     DELETE_FID (0);
;  316          1310  7                     RELEASE_LOCKBASIS (-1);
;  317          1311  7                     ALLOCATION_UNLOCK ();
;  318          1312  7                     IDX_FILE_ID = FID$C_INDEXF OR .CURRENT_VCB[VCB$W_RVN] ^ 24;
;  319          1313  7                     LOCK_ID = 0;
;  320          1314  7                     CACHE_LOCK (.IDX_FILE_ID, LOCK_ID, 1);
;  321          1315  7                     ALLOCATION_LOCK ();
;  322          1316  7                     DEQ_LOCK (.LOCK_ID);
;  323          1317  7                     CACHE_FLUSHED = -1;
;  324          1318  7                     LEAVE GET_FILE_NUM;
;  325          1319  7                     END
```

```
;  326          1320  6                ELSE
;  327          1321  6                    EXTEND_INDEX ();
;  328          1322  6                END
;  329          1323  6
;  330          1324  6 ! Move the EOF and zero the intervening blocks. Note that this version
;  331          1325  6 ! of the file system always sets the index file EOF to be physical end
;  332          1326  6 ! of file, because the index file is zeroed on extend. This code is
;  333          1327  6 ! present for compatibility with past and future file systems that may
;  334          1328  6 ! not zero the index file on extend. Serialize activity on the index
;  335          1329  6 ! file header.
;  336          1330  6 !
;  337          1331  6
;  338          1332  5            ELSE
;  339          1333  6                BEGIN
;  340          1334  6                TEMP = .CURR_LCKINDX;
;  341          1335  6                SERIAL_FILE (IDX_FCB [FCB$W_FID]);
;  342          1336  6
;  343          1337  6                LBN = MAP_IDX (.CURRENT_EOF+1, COUNT);
;  344          1338  6                ERASE_BLOCKS (.LBN, .COUNT, .IO_CHANNEL);
;  345          1339  6                CURRENT_EOF = .CURRENT_EOF + .COUNT;
;  346          1340  6
;  347          1341  6                HEADER = READ_IDX_HEADER ();
;  348          1342  6                BBLOCK [HEADER[FH2$W_RECATTR], FAT$L_EFBLK] = ROT (.CURRENT_EOF+1, 16);
;  349          1343  6                BBLOCK [HEADER[FH2$W_RECATTR], FAT$W_FFBYTE] = 0;
;  350          1344  6                IF .HEADER [FH2$B_IDOFFSET] GEQU ($BYTEOFFSET (FH2$L_HIGHWATER)+4)/2
;  351          1345  6                THEN HEADER [FH2$L_HIGHWATER] = .CURRENT_EOF + 1;
;  352          1346  6
;  353          1347  6                CHECKSUM (.HEADER);
;  354          1348  6                WRITE_HEADER ();
;  355          1349  6                IDX_FCB[FCB$L_EFBLK] = .CURRENT_EOF;
;  356          1350  6                RESET_LBN (.HEADER, .VCB[VCB$L_IXHDR2LBN]);
;  357          1351  6                WRITE_BLOCK (.HEADER);
;  358          1352  6                INVALIDATE (.HEADER);
;  359          1353  6
;  360          1354  6                RELEASE_SERIAL_LOCK (.CURR_LCKINDX);
;  361          1355  6                CURR_LCKINDX = .TEMP;
;  362          1356  5                END;
;  363          1357  5
;  364          1358  5 ! Go around the loop to try to allocate a file number again.
;  365          1359  5 !
;  366          1360  5
;  367          1361  5                LEAVE GET_FILE_NUM;
;  368          1362  5                END
;  369          1363  4            ELSE
;  370          1364  4
;  371          1365  4 ! We successfully filled the file ID cache from the bitmap.  Write back
;  372          1366  4 ! the index file bitmap buffer.
;  373          1367  4 !
;  374          1368  4
;  375          1369  4                WRITE_BLOCK (.BUFFER);
;  376          1370  4
;  377          1371  4            END
;  378          1372  4
;  379          1373  4 ! If the file ID cache had entries in it, all we have to do is check one out.
;  380          1374  4 !
;  381          1375  4
;  382          1376  3        ELSE
```

```
;  383          1377  3          PMS$GL_FIDHIT = .PMS$GL_FIDHIT + 1;
;  384          1378  3
;  385          1379  3      FILE_NUMBER = .FID_CACHE[VCA$L_FIDLIST];
;  386          1380  3      FID_CACHE[VCA$W_FIDCOUNT] = .FID_CACHE[VCA$W_FIDCOUNT] - 1;
;  387          1381  3      CH$MOVE (.FID_CACHE[VCA$W_FIDCOUNT]*4,
;  388          1382  3               FID_CACHE[VCA$L_FIDLIST]+4,
;  389          1383  3               FID_CACHE[VCA$L_FIDLIST]);
;  390          1384  3
;  391          1385  3      NEW_FID = .FILE_NUMBER;
;  392          1386  3      NEW_FID_RVN = .CURRENT_RVN;          ! record for cleanup
;  393          1387  3
;  394          1388  3  ! Map the file header. If it fails to map, we have screwed up badly.
;  395          1389  3  !
;  396          1390  3
;  397          1391  3      VBN = .FILE_NUMBER + .VCB[VCB$B_IBMAPSIZE] + .VCB[VCB$W_CLUSTER]*4;
;  398          1392  3      LBN = MAP_IDX (.VBN);
;  399          1393  3      IF .LBN EQL -1 THEN BUG_CHECK (HDRNOTMAP, FATAL, 'Allocated file header not mapped');
;  400          1394  3
;  401          1395  3      FILE_ID[FID$W_NUM] = .FILE_NUMBER<0,16>;
;  402          1396  3      FILE_ID[FID$B_NMX] = .FILE_NUMBER<16,8>;
;  403          1397  3      FILE_ID[FID$B_RVN] = .CURRENT_RVN;
;  404          1398  3
;  405          1399  3  ! If this is the creation of a new primary header, PRIM_LCKINDX will
;  406          1400  3  ! be zero.  In that case, serialize further processing on that header.
;  407          1401  3  ! If extension headers are being allocated, the primary lock index has
;  408          1402  3  ! already been established.
;  409          1403  3  !
;  410          1404  3
;  411          1405  3      IF .PRIM_LCKINDX EQL 0
;  412          1406  3      THEN
;  413          1407  4          BEGIN
;  414          1408  4
;  415          1409  4  ! Release the allocation lock prior to serializing on this file id.
;  416          1410  4  ! This could be a valid header that another process is trying to modify
;  417          1411  4  ! allocation on, and if so, we would deadlock if the allocation lock
;  418          1412  4  ! were not released now.
;  419          1413  4  !
;  420          1414  4
;  421          1415  4          ALLOCATION_UNLOCK ();
;  422          1416  4          PRIM_LCKINDX = SERIAL_FILE (.FILE_ID);
;  423          1417  4          NEW_LCKINDX = 1;
;  424          1418  3          END;
;  425          1419  3
;  426          1420  3  ! Read the header; then check the block read for resemblence to a file header.
;  427          1421  3  !
;  428          1422  3
;  429          1423  3      HEADER = READ_NEW_HEADER (.LBN);
;  430          1424  3
;  431          1425  3      IF .HEADER NEQ 0
;  432          1426  3      THEN
;  433          1427  4          BEGIN
;  434          1428  4          FILE_ID[FID$W_SEQ] = .HEADER[FH2$W_FID_SEQ];
;  435          1429  4          STATUS = CHECK_HEADER2 (.HEADER, .FILE_ID);
;  436          1430  4
;  437          1431  4  ! Make the final checks that the block is acceptable as a file header. We do
;  438          1432  4  ! not use valid file headers. Also, we skip file numbers with the low 16 bits
;  439          1433  4  ! all zero to avoid confusing the old FCS-11. Also skip file numbers in the
```

```
;  440          1434  4 ! reserved file number range to avoid total confusion if the volume is damaged.
;  441          1435  4 !
;  442          1436  4
;  443          1437  4         IF .FILE_ID[FID$W_NUM] EQL 0
;  444          1438  4         THEN
;  445          1439  4             WRITE_BLOCK (.HEADER)
;  446          1440  4         ELSE
;  447          1441  4             IF NOT .STATUS
;  448          1442  5             AND NOT (.FILE_ID[FID$B_NMX] EQL 0
;  449          1443  5                     AND .FILE_ID[FID$W_NUM] LEQU .CURRENT_VCB[VCB$B_RESFILES])
;  450          1444  4             THEN EXITLOOP;
;  451          1445  3         END;
;  452          1446  3
;  453          1447  3 ! If we got this far, i.e., did not exit the loop, we do not want to use
;  454          1448  3 ! this file header for some reason.  Before going around another time,
;  455          1449  3 ! release the serialization lock if we got one in this routine, and then
;  456          1450  3 ! reacquire the allocation lock for another pass around the loop.
;  457          1451  3 !
;  458          1452  3
;  459          1453  3     IF .NEW_LCKINDX
;  460          1454  3     THEN
;  461          1455  4         BEGIN
;  462          1456  4         IF .HEADER NEQ 0
;  463          1457  4         THEN INVALIDATE (.HEADER);
;  464          1458  4         RELEASE_SERIAL_LOCK (.PRIM_LCKINDX);
;  465          1459  4         PRIM_LCKINDX = 0;
;  466          1460  4         ALLOCATION_LOCK ();
;  467          1461  3         END;
;  468          1462  3
;  469          1463  2     END;                                    ! end of file number allocation loop
;  470          1464  2
;  471          1465  2 HEADER_LBN = .LBN;                          ! record LBN of new header
;  472          1466  2
;  473          1467  2 IF .STATUS EQL 0
;  474          1468  2 AND .(.HEAD[R)<0,32> NEQ 0
;  475          1469  2 THEN FILE_ID[FID$W_SEQ] = .EXE$GQ_SYSTIME<16,16>;
;  476          1470  2 FILE_ID[FID$W_SEQ] = .FILE_ID[FID$W_SEQ] + 1;
;  477          1471  2 CH$MOVE (FID$C_LENGTH, .FILE_ID, HEADER[FH2$W_FID]);
;  478          1472  2 HEADER[FH2$B_FID_RVN] = 0;
;  479          1473  2
;  480          1474  2 MARK_DIRTY (.HEADER);
;  481          1475  2 .HEADER
;  482          1476  2
;  483          1477  1 END;                                        ! end of routine CREATE_HEADER


                                .TITLE  CREHDR
                                .IDENT  \V04-001\

                                .EXTRN  CLU$GL_CLUB, PMS$GL_FIDHIT
                                .EXTRN  PMS$GL_FIDMISS, EXE$GQ_SYSTIME
                                .EXTRN  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$_HDRNOTMAP

                                                        .PSECT  $CODE$,NOWRT,2

                                      0BFC 00000        .ENTRY  CREATE_HEADER, Save R2,R3,R4,R5,R6,R7,R8,-   ; 1149
                                                                R9,R11
                         5E           2C  C2 00002      SUBL2   #44, SP
                                 20   AE  D4 00005      CLRL    NEW_LCKINDX                                  ; 1185
                  0000G  CF           00  FB 00008      CALLS   #0, ALLOCATION_LOCK                          ; 1244
                         59      98   AA  D0 0000D      MOVL    -104(BASE), VCB                              ; 1253
                         56      58   B9  D0 00011      MOVL    @88(VCB), FID_CACHE                          ; 1254
                                 1C   AE  D4 00015      CLRL    CACHE_FLUSHED                                ; 1255
                                 02   A6  B5 00018 1$:  TSTW    2(FID_CACHE)                                 ; 1265
                                      03  13 0001B      BEQL    2$
                                    0153  31 0001D      BRW     13$
                             00000000G 00 D6 00020 2$:  INCL    PMS$GL_FIDMISS                               ; 1268
                         18  AE       3A  A9 9A 00026   MOVZBL  58(VCB), VBN                                 ; 1269
 18   AE     38   A9          08      00  ED 0002B 3$:  CMPZV   #0, #8, 56(VCB), VBN                         ; 1273
                                      3A  15 00032      BLEQ    6$
                                      03  DD 00034      PUSHL   #3                                           ; 1275
                                      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                          ; 1276
                                      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                                          ; 1278
                                      CA  11 0005F      BRB     3$                                           ; 1273
                                 18   AE  DD 00061 5$:  PUSHL   VBN                                          ; 1287
                                 10   AE  DD 00064      PUSHL   BUFFER
                                      59  DD 00067      PUSHL   VCB
                  0000V  CF           03  FB 00069      CALLS   #3, FILL_FID_CACHE
                                 02   A6  B5 0006E 6$:  TSTW    2(FID_CACHE)                                 ; 1288
                                      03  13 00071      BEQL    7$
                                    00F3  31 00073      BRW     12$
                         5B           69  D0 00076 7$:  MOVL    (VCB), IDX_FCB                               ; 1298
                         57      3C   AB  D0 00079      MOVL    60(IDX_FCB), CURRENT_EOF                     ; 1299
                     38  AB           57  D1 0007D      CMPL    CURRENT_EOF, 56(IDX_FCB)                     ; 1300
                                      61  1F 00081      BLSSU   10$
                         50      94   AA  D0 00083      MOVL    -108(BASE), R0                               ; 1303
                         51      3C   A0  E9 00087      BLBC    60(R0), 8$
                             00000000G 00 D5 0008B      TSTL    CLU$GL_CLUB                                  ; 1304
                                      49  13 00091      BEQL    8$
                         45      1C   AE  E8 00093      BLBS    CACHE_FLUSHED, 8$                            ; 1305
                                      7E  D4 00097      CLRL    -(SP)                                        ; 1309
                  0000G  CF           01  FB 00099      CALLS   #1, DELETE_FID
                         7E           01  CE 0009E      MNEGL   #1, -(SP)                                    ; 1310
```

```
                0000G   CF              01 FB 000A1          CALLS   #1, RELEASE_LOCKBASIS                   ;
                0000G   CF              00 FB 000A6          CALLS   #0, ALLOCATION_UNLOCK                   ; 1311
                    50      98          AA D0 000AB          MOVL    -104(BASE), R0                          ; 1312
                    50      0E          A0 3C 000AF          MOVZWL  14(R0), R0                              ;
          50        50                  18 78 000B3          ASHL    #24, R0, R0                             ;
                    50                  01 88 000B7          BISB2   #1, IDX_FILE_ID                         ;
                            24          AE D4 000BA          CLRL    LOCK_ID                                 ; 1313
                            01          DD 000BD             PUSHL   #1                                      ; 1314
                            28          AE 9F 000BF          PUSHAB  LOCK_ID                                 ;
                                        50 DD 000C2          PUSHL   IDX_FILE_ID                             ;
                0000G   CF              03 FB 000C4          CALLS   #3, CACHE_LOCK                          ;
                0000G   CF              00 FB 000C9          CALLS   #0, ALLOCATION_LOCK                     ; 1315
                            24          AE DD 000CE          PUSHL   LOCK_ID                                 ; 1316
                0000G   CF              01 FB 000D1          CALLS   #1, DEQ_LOCK                            ;
                  1C    AE              01 CE 000D6          MNEGL   #1, CACHE_FLUSHED                       ; 1317
                                        05 11 000DA          BRB     9$                                      ; 1318
                0000G   CF              00 FB 000DC  8$:     CALLS   #0, EXTEND_INDEX                        ; 1321
                                    FF34 31 000E1  9$:       BRW     1$                                      ; 1300
                  04    AE          14  AA D0 000E4  10$:    MOVL    20(BASE), TEMP                          ; 1334
                                    24  AB 9F 000E9          PUSHAB  36(IDX_FCB)                             ; 1335
                0000G   CF              01 FB 000EC          CALLS   #1, SERIAL_FILE                         ;
                                    28  AE 9F 000F1          PUSHAB  COUNT                                   ; 1337
                                    01  A7 9F 000F4          PUSHAB  1(CURRENT_EOF)                          ;
                0000V   CF              02 FB 000F7          CALLS   #2, MAP_IDX                             ;
                  10    AE              50 D0 000FC          MOVL    R0, LBN                                 ;
                                    FF78 CA DD 00100         PUSHL   -136(BASE)                              ; 1338
                                    2C  AE DD 00104          PUSHL   COUNT                                   ;
                                    18  AE DD 00107          PUSHL   LBN                                     ;
                0000G   CF              03 FB 0010A          CALLS   #3, ERASE_BLOCKS                        ;
                    57      28          AE C0 0010F          ADDL2   COUNT, CURRENT_EOF                      ; 1339
                0000V   CF              00 FB 00113          CALLS   #0, READ_IDX_HEADER                     ; 1341
                    58                  50 D0 00118          MOVL    R0, HEADER                              ;
                    50          01      A7 9E 0011B          MOVAB   1(R7), R0                               ; 1342
    1C  A8          50                  10 9C 0011F          ROTL    #16, R0, 28(HEADER)                     ;
                                    20  A8 B4 00124          CLRW    32(HEADER)                              ; 1343
                    28                  68 91 00127          CMPB    (HEADER), #40                           ; 1344
                                        05 1F 0012A          BLSSU   11$                                     ;
                  4C    A8          01  A7 9E 0012C          MOVAB   1(R7), 76(HEADER)                       ; 1345
                                        58 DD 00131  11$:    PUSHL   HEADER                                  ; 1347
                0000G   CF              01 FB 00133          CALLS   #1, CHECKSUM                            ;
                0000G   CF              00 FB 00138          CALLS   #0, WRITE_HEADER                        ; 1348
                  3C    AB              57 D0 0013D          MOVL    CURRENT_EOF, 60(IDX_FCB)                ; 1349
                                    2C  A9 DD 00141          PUSHL   44(VCB)                                 ; 1350
                                        58 DD 00144          PUSHL   HEADER                                  ;
                0000G   CF              02 FB 00146          CALLS   #2, RESET_LBN                           ;
                                        58 DD 0014B          PUSHL   HEADER                                  ; 1351
                0000G   CF              01 FB 0014D          CALLS   #1, WRITE_BLOCK                         ;
                                        58 DD 00152          PUSHL   HEADER                                  ; 1352
                0000G   CF              01 FB 00154          CALLS   #1, INVALIDATE                          ;
                                    14  AA DD 00159          PUSHL   20(BASE)                                ; 1354
                0000G   CF              01 FB 0015C          CALLS   #1, RELEASE_SERIAL_LOCK                 ;
                  14    AA          04  AE D0 00161          MOVL    TEMP, 20(BASE)                          ; 1355
                                    FEAF 31 00166            BRW     1$                                      ; 1361
                                    0C  AE DD 00169  12$:    PUSHL   BUFFER                                  ; 1369
                0000G   CF              01 FB 0016C          CALLS   #1, WRITE_BLOCK                         ;
                                        06 11 00171          BRB     14$                                     ; 1265
                        00000000G       00 D6 00173  13$:    INCL    PMS$GL_FIDHIT                           ; 1377
```

```
                    14    AE        24    A6   D0 00179 14$:    MOVL    36(FID_CACHE), FILE_NUMBER                  ; 1379
                                    02    A6   B7 0017E          DECW    2(FID_CACHE)                                ; 1380
                          50        02    A6   3C 00181          MOVZWL  2(FID_CACHE), R0                            ; 1381
                          50              04   C4 00185          MULL2   #4, R0                                      ;
          24    A6        28    A6        50   28 00188          MOVC3   R0, 40(FID_CACHE), 36(FID_CACHE)            ; 1383
                          A8    AA        14   AE D0 0018E       MOVL    FILE_NUMBER, -88(BASE)                      ; 1385
                          AC    AA        A0   AA D0 00193       MOVL    -96(BASE), -84(BASE)                        ; 1386
                          50        38    A9   9A 00198          MOVZBL  56(VCB), R0                                 ; 1391
                51        50        14    AE   C1 0019C          ADDL3   FILE_NUMBER, R0, R1                         ;
                          50        3C    A9   3C 001A1          MOVZWL  60(VCB), R0                                 ;
                    18    AE              6140 DE 001A5          MOVAL   (R1)[R0], VBN                               ;
                                    18    AE   DD 001AA          PUSHL   VBN                                         ; 1392
                  0000V   CF              01   FB 001AD          CALLS   #1, MAP_IDX                                 ;
                     10   AE              50   D0 001B2          MOVL    R0, LBN                                     ;
                FFFFFFFF   8F        10   AE   D1 001B6          CMPL    LBN, #-1                                    ; 1393
                                          04   12 001BE          BNEQ    15$                                         ;
                                             FEFF 001C0          BUGW                                                ;
                                           0000* 001C2          .WORD   <BUG$_HDRNOTMAP!4>                          ;
                    04    BC        14    AE   B0 001C4 15$:    MOVW    FILE_NUMBER, @FILE_ID                       ; 1395
                          50        04    AC   D0 001C9          MOVL    FILE_ID, R0                                 ; 1396
                    05    A0        16    AE   90 001CD          MOVB    FILE_NUMBER+2, 5(R0)                        ;
                          50        04    AC   D0 001D2          MOVL    FILE_ID, R0                                 ; 1397
                    04    A0        A0    AA   90 001D6          MOVB    -96(BASE), 4(R0)                            ;
                                    18    AA   D5 001DB          TSTL    24(BASE)                                    ; 1405
                                          15   12 001DE          BNEQ    16$                                         ;
                  0000G   CF              00   FB 001E0          CALLS   #0, ALLOCATION_UNLOCK                       ; 1415
                                    04    AC   DD 001E5          PUSHL   FILE_ID                                     ; 1416
                  0000G   CF              01   FB 001E8          CALLS   #1, SERIAL_FILE                             ;
                     18   AA              50   D0 001ED          MOVL    R0, 24(BASE)                                ;
                     20   AE              01   D0 001F1          MOVL    #1, NEW_LCKINDX                             ; 1417
                                    10    AE   DD 001F5 16$:    PUSHL   LBN                                         ; 1423
                  0000V   CF              01   FB 001F8          CALLS   #1, READ_NEW_HEADER                         ;
                          58              50   D0 001FD          MOVL    R0, HEADER                                  ;
                                          3E   13 00200          BEQL    18$                                         ; 1425
                          50        04    AC   D0 00202          MOVL    FILE_ID, R0                                 ; 1428
                    02    A0        0A    A8   B0 00206          MOVW    10(HEADER), 2(R0)                           ;
                                    04    AC   DD 00208          PUSHL   FILE_ID                                     ; 1429
                                    58    DD   0020E             PUSHL   HEADER                                      ;
                  0000G   CF              02   FB 00210          CALLS   #2, CHECK_HEADER2                           ;
                     08   AE              50   D0 00215          MOVL    R0, STATUS                                  ;
                          52        04    AC   D0 00219          MOVL    FILE_ID, R2                                 ; 1437
                                          62   B5 0021D          TSTW    (R2)                                        ;
                                          09   12 0021F          BNEQ    17$                                         ;
                                    58    DD   00221             PUSHL   HEADER                                      ; 1439
                  0000G   CF              01   FB 00223          CALLS   #1, WRITE_BLOCK                             ;
                                          16   11 00228          BRB     18$                                         ;
                          12        08    AE   E8 0022A 17$:    BLBS    STATUS, 18$                                 ; 1441
                                    05    A2   95 0022E          TSTB    5(R2)                                       ; 1442
                                          2F   12 00231          BNEQ    21$                                         ;
                          50        98    AA   D0 00233          MOVL    -104(BASE), R0                              ; 1443
                          51        4F    A0   9A 00237          MOVZBL  79(R0), R1                                  ;
                          62              51   B1 0023B          CMPW    R1, (R2)                                    ;
                                          12   1F 0023E          BLSSU   21$                                         ;
                          1B        20    AE   E9 00240 18$:    BLBC    NEW_LCKINDX, 20$                            ; 1453
                                          58   D5 00244          TSTL    HEADER                                      ; 1456
                                          07   13 00246          BEQL    19$                                         ;
                                          58   DD 00248          PUSHL   HEADER                                      ; 1457
```

```
                      0000G   CF               01   FB 0024A          CALLS   #1, INVALIDATE                        ;
                                         18    AA   DD 0024F  19$:    PUSHL   24(BASE)                              ; 1458
                      0000G   CF               01   FB 00252          CALLS   #1, RELEASE_SERIAL_LOCK               ;
                                         18    AA   D4 00257          CLRL    24(BASE)                              ; 1459
                      0000G   CF               00   FB 0025A          CALLS   #0, ALLOCATION_LOCK                   ; 1460
                                             FDB6   31 0025F  20$:    BRW     1$                                    ; 1256
                         B0   AA         10    AE   D0 00262  21$:    MOVL    LBN, -80(BASE)                        ; 1465
                                         08    AE   D5 00267          TSTL    STATUS                                ; 1467
                                               10   12 0026A          BNEQ    22$                                   ;
                                               68   D5 0026C          TSTL    (HEADER)                              ; 1468
                                               0C   13 0026E          BEQL    22$                                   ;
                              50         04    AC   D0 00270          MOVL    FILE_ID, R0                           ; 1469
                         02   A0  00000000G    00   B0 00274          MOVW    EXE$GQ_SYSTIME+2, 2(R0)               ;
                              50         04    AC   D0 0027C  22$:    MOVL    FILE_ID, R0                           ; 1470
                                         02    A0   B6 00280          INCW    2(R0)                                 ;
          08   A8         04   BC              06   28 00283          MOVC3   #6, @FILE_ID, 8(HEADER)               ; 1471
                                         0C    A8   94 00289          CLRB    12(HEADER)                            ; 1472
                                               58   DD 0028C          PUSHL   HEADER                                ; 1474
                      0000G   CF               01   FB 0028E          CALLS   #1, MARK_DIRTY                        ;
                              50               58   D0 00293          MOVL    HEADER, R0                            ; 1477
                                                    04 00296          RET                                           ;

; Routine Size:  663 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  485          1478  1 ROUTINE FILL_FID_CACHE (VCB, BUFFER, VBN) : L_NORM NOVALUE =
;  486          1479  1
;  487          1480  1 !++
;  488          1481  1 !
;  489          1482  1 ! FUNCTIONAL DESCRIPTION:
;  490          1483  1 !
;  491          1484  1 !       This routine refills the cache from the supplied bitmap buffer.
;  492          1485  1 !       It will not fill the cache with file ID's that represent
;  493          1486  1 !       headers past the current index file EOF.
;  494          1487  1 !
;  495          1488  1 !
;  496          1489  1 ! CALLING SEQUENCE:
;  497          1490  1 !       FILL_FID_CACHE (ARG1, ARG2, ARG3)
;  498          1491  1 !
;  499          1492  1 ! INPUT PARAMETERS:
;  500          1493  1 !       ARG1: address of volume VCB
;  501          1494  1 !       ARG2: address of bitmap buffer
;  502          1495  1 !       ARG3: relative block number in bitmap
;  503          1496  1 !
;  504          1497  1 ! IMPLICIT INPUTS:
;  505          1498  1 !       NONE
;  506          1499  1 !
;  507          1500  1 ! OUTPUT PARAMETERS:
;  508          1501  1 !       NONE
;  509          1502  1 !
;  510          1503  1 ! IMPLICIT OUTPUTS:
;  511          1504  1 !       NONE
;  512          1505  1 !
;  513          1506  1 ! ROUTINE VALUE:
;  514          1507  1 !       NONE
;  515          1508  1 !
;  516          1509  1 ! SIDE EFFECTS:
;  517          1510  1 !       file ID cache modified
;  518          1511  1 !
;  519          1512  1 !--
;  520          1513  1
;  521          1514  2 BEGIN
;  522          1515  2
;  523          1516  2 MAP
;  524          1517  2         VCB             : REF BBLOCK,   ! local copy of VCB address
;  525          1518  2         BUFFER          : REF BITVECTOR; ! address of index file bitmap buffer
;  526          1519  2
;  527          1520  2 LOCAL
;  528          1521  2         CACHE           : REF BBLOCK,   ! pointer to cache block
;  529          1522  2         FID_CACHE       : REF BBLOCK,   ! pointer to file ID cache
;  530          1523  2         ADDRESS         : REF BITVECTOR, ! address of byte in buffer
;  531          1524  2         FREE_COUNT,                     ! count of cache entries to fill
;  532          1525  2         BITPOS,                         ! bit positon of free bit within byte
;  533          1526  2         BITPOS2,                        ! bit positon of first used bit
;  534          1527  2         FILE_NUMBER,                    ! file number found
;  535          1528  2         IDX_VBN;                        ! current block in index bitmap
;  536          1529  2
;  537          1530  2 BIND_COMMON;
;  538          1531  2
;  539          1532  2
;  540          1533  2 ! If the cache is not currently marked valid, attempt to take out the
.  541          1534  2 ! cache lock if we are in a cluster and may do so.
```

```
;  542          1535  2 !
;  543          1536  2
;  544          1537  2 CACHE = .VCB[VCB$L_CACHE];
;  545          1538  2 FID_CACHE = .CACHE[VCA$L_FIDCACHE];
;  546          1539  2 IF NOT .CACHE[VCA$V_FIDC_VALID]
;  547          1540  2 THEN INIT_FID_CACHE (.CACHE);
;  548          1541  2
;  549          1542  2 ! Fill the cache from the supplied bitmap buffer. Find each byte containing
;  550          1543  2 ! a free bit, and then find the free bit.
;  551          1544  2 !
;  552          1545  2
;  553          1546  2 ADDRESS = .BUFFER;
;  554          1547  2 FREE_COUNT = .FID_CACHE[VCA$W_FIDSIZE]/2 - .FID_CACHE[VCA$W_FIDCOUNT] + 1;
;  555          1548  2
;  556          1549  2 WHILE 1 DO
;  557          1550  3     BEGIN
;  558          1551  3     IF CH$FAIL (ADDRESS = CH$FIND_NOT_CH (.BUFFER+512-.ADDRESS, .ADDRESS, 255))
;  559          1552  3     THEN EXITLOOP;
;  560          1553  3     FFC (%REF (0), %REF (8), .ADDRESS, BITPOS);
;  561          1554  3     FILE_NUMBER = .VBN*4096 + (.ADDRESS-.BUFFER)*8 + .BITPOS + 1;
;  562          1555  3
;  563          1556  3 ! Check file number against index file EOF and the maximum file limit.
;  564          1557  3 !
;  565          1558  3
;  566          1559  3     IF .FILE_NUMBER + .VCB[VCB$B_IBMAPSIZE] + .VCB[VCB$W_CLUSTER]*4
;  567          1560  3         GTRU .BBLOCK [.VCB[VCB$L_FCBFL], FCB$L_EFBLK]
;  568          1561  3     OR .FILE_NUMBER GTRU .VCB[VCB$L_MAXFILES]
;  569          1562  3     THEN EXITLOOP;
;  570          1563  3
;  571          1564  3 ! Enter the file number in the cache and mark it busy in the bitmap.
;  572          1565  3 ! Exit the loop if the cache is now full enough.
;  573          1566  3 !
;  574          1567  3
;  575          1568  3     ADDRESS[.BITPOS] = 1;
;  576          1569  3     FID_CACHE[VCA$W_FIDCOUNT] = .FID_CACHE[VCA$W_FIDCOUNT] + 1;
;  577          1570  3     VECTOR [FID_CACHE[VCA$L_FIDLIST], .FID_CACHE[VCA$W_FIDCOUNT]-1] = .FILE_NUMBER;
;  578          1571  3     FREE_COUNT = .FREE_COUNT - 1;
;  579          1572  3     IF .FREE_COUNT LEQ 0
;  580          1573  3     OR NOT .CACHE[VCA$V_FIDC_VALID]
;  581          1574  3     THEN EXITLOOP;
;  582          1575  2     END;                                ! end of bitmap processing loop
;  583          1576  2
;  584          1577  2 IDX_VBN = .VBN;                        ! update current VBN of index file bitmap
;  585          1578  2 IF .FILE_NUMBER<0,12> EQL 0
;  586          1579  2 THEN IDX_VBN = .IDX_VBN + 1;
;  587          1580  2 VCB[VCB$B_IBMAPVBN] = .IDX_VBN;
;  588          1581  2
;  589          1582  1 END;                                  ! end of routine FILL_FID_CACHE


                                01FC 00000 FILL_FID_CACHE:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8          ; 1478
                          50    04   AC  D0 00002          MOVL    VCB, R0           ; 1537
                          54    58   A0  D0 00006          MOVL    88(R0), CACHE     ;
```

```
                            52          64 D0 0000A          MOVL    (CACHE), FID_CACHE                  : 1538
                            07     0B   A4 E8 0000D          BLBS    11(CACHE), 1$                       : 1539
                                        54 DD 00011          PUSHL   CACHE                               : 1540
                     0000V  CF          01 FB 00013          CALLS   #1, INIT_FID_CACHE
                            57     08   AC D0 00018 1$:      MOVL    BUFFER, ADDRESS                     : 1546
                            53          62 3C 0001C          MOVZWL  (FID_CACHE), R3                     : 1547
                            53          02 C6 0001F          DIVL2   #2, R3
                            50     02   A2 3C 00022          MOVZWL  2(FID_CACHE), R0
                            53          50 C2 00026          SUBL2   R0, R3
                                        53 D6 00029          INCL    FREE_COUNT
               50      08   AC          57 C3 0002B 2$:      SUBL3   ADDRESS, BUFFER, R0                 : 1551
                            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           : 1553
               50      0C   AC          0C 78 00048          ASHL    #12, VBN, R0                        : 1554
               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                             : 1559
                            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                            : 1560
                       3C   A0          55 D1 00072          CMPL    R5, 60(R0)
                                        1E 1A 00076          BGTRU   5$
                       44   A1          56 D1 00078          CMPL    FILE_NUMBER, 68(R1)                 : 1561
                                        18 1A 0007C          BGTRU   5$
               00           67          58 E2 0007E          BBSS    BITPOS, (ADDRESS), 4$               : 1568
                                   02   A2 B6 00082 4$:      INCW    2(FID_CACHE)                        : 1569
                            50     02   A2 3C 00085          MOVZWL  2(FID_CACHE), R0                    : 1570
                       20 A240          56 D0 00089          MOVL    FILE_NUMBER, 32(FID_CACHE)[R0]
                                        53 D7 0008E          DECL    FREE_COUNT                          : 1571
                                        04 15 00090          BLEQ    5$                                  : 1572
                            95     0B   A4 E8 00092          BLBS    11(CACHE), 2$                       : 1573
                            51     0C   AC D0 00096 5$:      MOVL    VBN, IDX_VBN                        : 1577
                     0FFF   8F          56 B3 0009A          BITW    FILE_NUMBER, #4095                  : 1578
                                        02 12 0009F          BNEQ    6$
                                        51 D6 000A1          INCL    IDX_VBN                             : 1579
                            50     04   AC D0 000A3 6$:      MOVL    VCB, R0                             : 1580
                       3A   A0          51 90 000A7          MOVB    IDX_VBN, 58(R0)
                                           04 000AB          RET                                         : 1582

; Routine Size:  172 bytes,    Routine Base:  $CODE$ + 0297
```

```
;  591          1583  1 GLOBAL ROUTINE INIT_FID_CACHE (CACHE) : L_NORM NOVALUE =
;  592          1584  1
;  593          1585  1 !++
;  594          1586  1 !
;  595          1587  1 ! FUNCTIONAL DESCRIPTION:
;  596          1588  1 !
;  597          1589  1 !       This routine refills the cache from the supplied bitmap buffer.
;  598          1590  1 !       It will not fill the cache with file ID's that represent
;  599          1591  1 !       headers past the current index file EOF.
;  600          1592  1 !
;  601          1593  1 !
;  602          1594  1 ! CALLING SEQUENCE:
;  603          1595  1 !       INIT_FID_CACHE (CACHE)
;  604          1596  1 !
;  605          1597  1 ! INPUT PARAMETERS:
;  606          1598  1 !       CACHE: pointer to main cache block
;  607          1599  1 !
;  608          1600  1 ! IMPLICIT INPUTS:
;  609          1601  1 !       NONE
;  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 !       cache marked valid, lock taken out
;  622          1614  1 !
;  623          1615  1 !--
;  624          1616  1
;  625          1617  2 BEGIN
;  626          1618  2
;  627          1619  2 MAP
;  628          1620  2         CACHE           : REF BBLOCK;   ! pointer to cache block
;  629          1621  2
;  630          1622  2 LOCAL
;  631          1623  2         FID_CACHE       : REF BBLOCK,   ! pointer to file ID cache
;  632          1624  2         INDEX_FID;                      ! lock basis for index file
;  633          1625  2
;  634          1626  2 BIND_COMMON;
;  635          1627  2
;001  !CDS0018  1628  2 EXTERNAL
;002  !CDS0018  1629  2         CLU$GL_CLUB     : ADDRESSING_MODE (GENERAL);
;003  !CDS0018  1630  2
;  636          1631  2 EXTERNAL ROUTINE
;  637          1632  2         CACHE_LOCK      : L_NORM;       ! acquire special cache lock
;  638          1633  2
;  639          1634  2
;  640          1635  2 ! If the cache is not currently marked valid, attempt to take out the
;  641          1636  2 ! cache lock if we are in a cluster and may do so.
;  642          1637  2 !
;  643          1638  2
;  644          1639  2 FID_CACHE = .CACHE[VCA$L_FIDCACHE];
```

```
;  645          1640  2 IF NOT .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_DMT]
;  646          1641  2 AND NOT .CURRENT_VCB[VCB$V_WRITE_IF]
;  647          1642  2 AND .FID_CACHE[VCA$W_FIDSIZE] GTRU 1
;  648          1643  2 THEN
;  649          1644  3     BEGIN
;  650          1645  3     IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
;001 !CDS0018   1646  3         AND .CLU$GL_CLUB NEQ 0
;  651          1647  3     THEN
;  652          1648  4         BEGIN
;  653          1649  4         INDEX_FID = FID$C_INDEXF OR .CURRENT_VCB[VCB$W_RVN] ^ 24;
;  654          1650  4         IF CACHE_LOCK (.INDEX_FID, FID_CACHE[VCA$L_FIDCLKID], 0)
;  655          1651  4    .    THEN CACHE[VCA$V_FIDC_VALID] = 1;
;  656          1652  4         END
;  657          1653  3     ELSE
;  658          1654  3         CACHE[VCA$V_FIDC_VALID] = 1;
;  659          1655  2     END;
;  660          1656  2
;  661          1657  1 END;                                    ! end of routine INIT_FID_CACHE


                                                000C 00000        .ENTRY  INIT_FID_CACHE, Save R2,R3                  ; 1583
                                  52        04  AC  D0 00002        MOVL    CACHE, R2                                   ; 1639
                                  53            62  D0 00006        MOVL    (R2), FID_CACHE
                                  51        94  AA  D0 00009        MOVL    -108(BASE), R1                              ; 1640
                              44      3A  A1            05  E0 0000D  BBS     #5, 58(R1), 2$
                                  50        98  AA  D0 00012        MOVL    -104(BASE), R0                              ; 1641
                                  3C        0B  A0  E8 00016        BLBS    11(R0), 2$
                                  01            63  B1 0001A        CMPW    (FID_CACHE), #1                             ; 1642
                                                37  1B 0001D        BLEQU   2$
                                  2F        3C  A1  E9 0001F        BLBC    60(R1), 1$                                  ; 1645
                                      00000000G 00  D5 00023        TSTL    CLU$GL_CLUB                                 ; 1646
                                                27  13 00029        BEQL    1$
                                  50        98  AA  D0 0002B        MOVL    -104(BASE), R0                              ; 1649
                                  50        0E  A0  3C 0002F        MOVZWL  14(R0), R0
                              50                  50  18  78 00033  ASHL    #24, R0, R0
                                  50                50  01  88 00037  BISB2   #1, INDEX_FID
                                                    7E  D4 0003A    CLRL    -(SP)                                       ; 1650
                                        04      A3  9F 0003C        PUSHAB  4(FID_CACHE)
                                                50  DD 0003F        PUSHL   INDEX_FID
                                  0000G     CF            03  FB 00041  CALLS   #3, CACHE_LOCK
                                            0D            50  E9 00046  BLBC    R0, 2$
                                  50        04  AC  D0 00049        MOVL    CACHE, R0                                   ; 1651
                                0B      A0            01  88 0004D  BISB2   #1, 11(R0)
                                                      04  00051    RET                                                 ; 1645
                                0B      A2            01  88 00052 1$:  BISB2   #1, 11(R2)                          ; 1654
                                                      04  00056 2$:  RET                                             ; 1657

; Routine Size:  87 bytes,    Routine Base:  $CODE$ + 0343
```

```
;  663     1658  1 ROUTINE READ_NEW_HEADER (LBN) : L_NORM =
;  664     1659  1
;  665     1660  1 !++
;  666     1661  1 !
;  667     1662  1 ! FUNCTIONAL DESCRIPTION:
;  668     1663  1 !
;  669     1664  1 !       This routine reads the block about to be used for a new file header.
;  670     1665  1 !       It uses a local condition handler to fix up errors.
;  671     1666  1 !
;  672     1667  1 !
;  673     1668  1 ! CALLING SEQUENCE:
;  674     1669  1 !       READ_NEW_HEADER (ARG1)
;  675     1670  1 !
;  676     1671  1 ! INPUT PARAMETERS:
;  677     1672  1 !       ARG1: LBN of block to read
;  678     1673  1 !
;  679     1674  1 ! IMPLICIT INPUTS:
;  680     1675  1 !       NONE
;  681     1676  1 !
;  682     1677  1 ! OUTPUT PARAMETERS:
;  683     1678  1 !       NONE
;  684     1679  1 !
;  685     1680  1 ! IMPLICIT OUTPUTS:
;  686     1681  1 !       NONE
;  687     1682  1 !
;  688     1683  1 ! ROUTINE VALUE:
;  689     1684  1 !       address of buffer containing block or 0 if bad
;  690     1685  1 !
;  691     1686  1 ! SIDE EFFECTS:
;  692     1687  1 !       block read and/or written
;  693     1688  1 !
;  694     1689  1 !--
;  695     1690  1
;  696     1691  2 BEGIN
;  697     1692  2
;  698     1693  2 LOCAL
;  699     1694  2         HEADER          : REF BBLOCK;   ! address of block read
;  700     1695  2
;  701     1696  2 BASE_REGISTER;
;  702     1697  2
;  703     1698  2 EXTERNAL ROUTINE
;  704     1699  2         READ_BLOCK      : L_NORM,       ! read a block
;  705     1700  2         WRITE_BLOCK     : L_NORM,       ! write a block
;  706     1701  2         INVALIDATE      : L_NORM,       ! invalidate a buffer
;  707     1702  2         CREATE_BLOCK    : L_NORM;       ! create a new block buffer
;  708     1703  2
;  709     1704  2 ! Under control of the condition handler, we read the block. If the read
;  710     1705  2 ! fails, we attempt to rewrite the block and then read it again. If either
;  711     1706  2 ! of the latter fails, we return failure.
;  712     1707  2 !
;  713     1708  2
;  714     1709  2 ENABLE HANDLER;
;  715     1710  2
;  716     1711  2 HEADER = READ_BLOCK (.LBN, 1, HEADER_TYPE);
;  717     1712  2
;  718     1713  2 IF .HEADER EQL 0
;  719     1714  2 THEN
```

```
;  720          1715  3     BEGIN
;  721          1716  3     HEADER = CREATE_BLOCK (.LBN, 1, HEADER_TYPE);
;  722          1717  3     (.HEADER)<0,32> = 1;
;  723          1718  3     WRITE_BLOCK (.HEADER);
;  724          1719  3     INVALIDATE (.HEADER);
;  725          1720  3     HEADER = READ_BLOCK (.LBN, 1, HEADER_TYPE);
;  726          1721  2     END;
;  727          1722  2
;  728          1723  2 RETURN .HEADER;
;  729          1724  2
;  730          1725  1 END;                                    ! end of routine READ_NEW_HEADER


                                        0004 00000 READ_NEW_HEADER:
                                                   .WORD   Save R2                          ;  1658
                            6D      0042   CF  DE 00002           MOVAL   2$, (FP)         ;  1691
                            7E             01  7D 00007           MOVQ    #1, -(SP)        ;  1711
                                04         AC  DD 0000A           PUSHL   LBN              ;
                     0000G  CF             03  FB 0000D           CALLS   #3, READ_BLOCK   ;
                            52             50  D0 00012           MOVL    R0, HEADER       ;
                                           2D  12 00015           BNEQ    1$               ;  1713
                            7E             01  7D 00017           MOVQ    #1, -(SP)        ;  1716
                                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)     ;  1717
                                           52  DD 00028           PUSHL   HEADER           ;  1718
                     0000G  CF             01  FB 0002A           CALLS   #1, WRITE_BLOCK  ;
                                           52  DD 0002F           PUSHL   HEADER           ;  1719
                     0000G  CF             01  FB 00031           CALLS   #1, INVALIDATE   ;
                            7E             01  7D 00036           MOVQ    #1, -(SP)        ;  1720
                                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$:      MOVL    HEADER, R0       ;  1723
                                           04  00047             RET                      ;  1725
                                     0000  00048 2$:      .WORD   Save nothing             ;  1691
                                           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:  $CODE$ + 039A
```

```
;  732     1726  1 ROUTINE HANDLER (SIGNAL, MECHANISM) =
;  733     1727  1
;  734     1728  1 !++
;  735     1729  1 !
;  736     1730  1 ! FUNCTIONAL DESCRIPTION:
;  737     1731  1 !
;  738     1732  1 !       This routine is the condition handler for the initial header read.
;  739     1733  1 !       On surface errors, it unwinds and causes a return of 0 to the caller
;  740     1734  1 !       of the I/O routine to indicate error. Hard drive errors cause the
;  741     1735  1 !       usual error exit.
;  742     1736  1 !
;  743     1737  1 ! CALLING SEQUENCE:
;  744     1738  1 !       HANDLER (ARG1, ARG2)
;  745     1739  1 !
;  746     1740  1 ! INPUT PARAMETERS:
;  747     1741  1 !       ARG1: address of signal array
;  748     1742  1 !       ARG2: address of mechanism array
;  749     1743  1 !
;  750     1744  1 ! IMPLICIT INPUTS:
;  751     1745  1 !       NONE
;  752     1746  1 !
;  753     1747  1 ! OUTPUT PARAMETERS:
;  754     1748  1 !       NONE
;  755     1749  1 !
;  756     1750  1 ! IMPLICIT OUTPUTS:
;  757     1751  1 !       NONE
;  758     1752  1 !
;  759     1753  1 ! ROUTINE VALUE:
;  760     1754  1 !       SS$_RESIGNAL or none if unwind
;  761     1755  1 !
;  762     1756  1 ! SIDE EFFECTS:
;  763     1757  1 !       NONE
;  764     1758  1 !
;  765     1759  1 !--
;  766     1760  1
;  767     1761  1
;  768     1762  2 BEGIN
;  769     1763  2
;  770     1764  2 MAP
;  771     1765  2         SIGNAL          : REF BBLOCK,   ! signal arg array
;  772     1766  2         MECHANISM       : REF BBLOCK;   ! mechanism arg array
;  773     1767  2
;  774     1768  2
;  775     1769  2 ! If the condition is change mode to user (error exit) and the status is
;  776     1770  2 ! read error, zero the return R0 and unwind to the the establisher. On
;  777     1771  2 ! most write errors, zero the return R0 and unwind to the caller.
;  778     1772  2 ! Otherwise, just resignal the condition.
;  779     1773  2 !
;  780     1774  2
;  781     1775  2 IF .SIGNAL[CHF$L_SIG_NAME] EQL SS$_CMODUSER
;  782     1776  2 THEN
;  783     1777  3     BEGIN
;  784     1778  3     MECHANISM[CHF$L_MCH_SAVR0] = 0;
;  785     1779  3
;  786     1780  3     IF SURFACE_ERROR (.SIGNAL[CHF$L_SIG_ARG1])
;  787     1781  3     THEN
;  788     1782  4         $UNWIND (DEPADR = MECHANISM[CHF$L_MCH_DEPTH])
```

```
;  789     1783  2     END;
;  790     1784  2
;  791     1785  2 RETURN SS$_RESIGNAL;                   ! status is irrelevant if unwinding
;  792     1786  2
;  793     1787  1 END;                                   ! end of routine HANDLER


                                                                .EXTRN  SYS$UNWIND

                                        0000 00000 HANDLER:.WORD    Save nothing                        ; 1726
                       50         04  AC  D0 00002          MOVL     SIGNAL, R0                          ; 1775
         00000424      8F         04  A0  D1 00006          CMPL     4(R0), #1060
                                      41  12 0000E          BNEQ     2$
                       50         08  AC  D0 00010          MOVL     MECHANISM, R0                       ; 1778
                                  0C  A0  D4 00014          CLRL     12(R0)
                       50         04  AC  D0 00017          MOVL     SIGNAL, R0                          ; 1780
         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$:      CLRL     -(SP)                               ; 1782
  7E           08      AC             08  C1 00045          ADDL3    #8, MECHANISM, -(SP)
         00000000G     00             02  FB 0004A          CALLS    #2, SYS$UNWIND
                       50       0918  8F  3C 00051 2$:      MOVZWL   #2328, R0                           ; 1785
                                          04 00056          RET                                          ; 1787

; Routine Size:  87 bytes,    Routine Base:  $CODE$ + 03F2
```

```
;  795          1788  1 GLOBAL ROUTINE READ_IDX_HEADER : L_NORM =
;  796          1789  1
;  797          1790  1 !++
;  798          1791  1 !
;  799          1792  1 ! FUNCTIONAL DESCRIPTION:
;  800          1793  1 !
;  801          1794  1 !       This routine reads the volume's index file header, using the
;  802          1795  1 !       alternate if it seems appropriate.
;  803          1796  1 !
;  804          1797  1 ! CALLING SEQUENCE:
;  805          1798  1 !       READ_IDX_HEADER ()
;  806          1799  1 !
;  807          1800  1 ! INPUT PARAMETERS:
;  808          1801  1 !       NONE
;  809          1802  1 !
;  810          1803  1 ! IMPLICIT INPUTS:
;  811          1804  1 !       CURRENT_VCB· VCB of volume
;  812          1805  1 !
;  813          1806  1 ! OUTPUT PARAMETERS:
;  814          1807  1 !       NONE
;  815          1808  1 !
;  816          1809  1 ! IMPLICIT OUTPUTS:
;  817          1810  1 !       NONE
;  818          1811  1 !
;  819          1812  1 ! ROUTINE VALUE:
;  820          1813  1 !       address of file header read
;  821          1814  1 !
;  822          1815  1 ! SIDE EFFECTS:
;  823          1816  1 !       NONE
;  824          1817  1 !
;  825          1818  1 !--
;  826          1819  1
;  827          1820  2 BEGIN
;  828          1821  2
;  829          1822  2
;  830          1823  2 LOCAL
;  831          1824  2         HEADER          : REF BBLOCK,   ! address of header read
;  832          1825  2         FCB             : REF BBLOCK;   ! address of index file FCB
;  833          1826  2
;  834          1827  2 BIND_COMMON;
;  835          1828  2
;  836          1829  2 EXTERNAL ROUTINE
;  837          1830  2         FILE_SIZE       : L_NORM,       ! compute file header file size
;  838          1831  2         READ_HEADER     : L_NORM,       ! read file header
;  839          1832  2         READ_BLOCK      : L_NORM,       ! read a disk block
;  840          1833  2         CHECK_HEADER2   : L_NORM,       ! validate file header
;  841          1834  2         RESET_LBN       : L_NORM,       ! reassign LBN of buffer
;  842          1835  2         INVALIDATE      : L_NORM;       ! invalidate buffer
;  843          1836  2
;  844          1837  2
;  845          1838  2 ! Read the index file header. Check the file size against the
;  846          1839  2 ! file size in the FCB. A mismatch indicates a failure in writing the
;  847          1840  2 ! header the last time; if this occurs, try the alternate header instead.
;  848          1841  2 !
;  849          1842  2
;  850          1843  2 SAVE_STATUS = .USER_STATUS;
;  851          1844  2
```

```
;  852          1845  2 FCB = .CURRENT_VCB[VCB$L_FCBFL];
;  853          1846  2 HEADER = READ_HEADER (0, .FCB);
;  854          1847  2 IF FILE_SIZE (.HEADER) LSSU .FCB[FCB$L_FILESIZE]
;  855          1848  2 THEN
;  856          1849  3     BEGIN
;  857          1850  3     FILE_HEADER = 0;
;  858          1851  3     INVALIDATE (.HEADER);
;  859          1852  3     HEADER = READ_BLOCK (.CURRENT_VCB[VCB$L_IXHDR2LBN], 1, HEADER_TYPE);
;  860          1853  3     IF NOT CHECK_HEADER2 (.HEADER, UPLIT WORD (FID$C_INDEXF, FID$C_INDEXF, 0))
;  861          1854  3     THEN
;  862          1855  4         BEGIN
;  863          1856  4         INVALIDATE (.HEADER);
;  864          1857  4         ERR_EXIT (0);
;  865          1858  3         END;
;  866          1859  3     IF FILE_SIZE (.HEADER) LSSU .FCB[FCB$L_FILESIZE]
;  867          1860  3     THEN ERR_EXIT (SS$_BADFILEHDR);
;  868          1861  3     FILE_HEADER = .HEADER;
;  869          1862  3     RESET_LBN (.HEADER, .FCB[FCB$L_HDLBN]);
;  870          1863  2     END;
;  871          1864  2
;  872          1865  2 USER_STATUS = .SAVE_STATUS;
;  873          1866  2
;  874          1867  2 .HEADER
;  875          1868  1 END;                                    ! end of routine READ_IDX_HEADER


                                                00449          .BLKB   1
                                    0000  0001  0001  0044A P.AAA: .WORD   1, 1, 0                                  ;

                                                               .EXTRN  FILE_SIZE, READ_HEADER

                                                      000C 00000 .ENTRY  READ_IDX_HEADER, Save R2,R3               ; 1788
                                      C0    AA      80   AA  D0 00002 MOVL   -128(BASE), -64(BASE)                 ; 1843
                                            52      98   BA  D0 00007 MOVL   @-104(BASE), FCB                      ; 1845
                                                         52  DD 0000B PUSHL  FCB                                  ; 1846
                                                         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                               ; 1847
                                  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)                              ; 1850
                                                         53  DD 00027 PUSHL  HEADER                               ; 185
                                  0000G     CF             01  FB 00029 CALLS  #1, INVALIDATE
                                            7E             01  7D 0002E MOVQ   #1, -(SP)                           ; 1852
                                            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                                ; 1853
                                                         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                               ; 1856
                                  0000G     CF             01  FB 0004F CALLS  #1, INVALIDATE
```

```
                                00  BF 00054          CHMU    #0                              ; 1857
                                    04 00056          RET                                     ;
                                53  DD 00057 1$:      PUSHL   HEADER                          ; 1859
        0000G   CF              01  FB 00059          CALLS   #1, FILE_SIZE                   ;
           38   A2              50  D1 0005E          CMPL    R0, 56(FCB)                     ;
                                05  1E 00062          BGEQU   2$                              ;
                        0810    8F  BF 00064          CHMU    #2064                           ; 1860
                                    04 00068          RET                                     ;
           04   AA              53  D0 00069 2$:      MOVL    HEADER, 4(BASE)                 ; 1861
                          34    A2  DD 0006D          PUSHL   52(FCB)                         ; 1862
                                53  DD 00070          PUSHL   HEADER                          ;
        0000G   CF              02  FB 00072          CALLS   #2, RESET_LBN                   ;
           80   AA        C0    AA  D0 00077 3$:      MOVL    -64(BASE), -128(BASE)           ; 1865
                50              53  D0 0007C          MOVL    HEADER, R0                      ; 1868
                                    04 0007F          RET                                     ;
```

; Routine Size: 128 bytes, Routine Base: $CODE$ + 0450

; 876 1869 1

```
;  878     187C  1 GLOBAL ROUTINE MAP_IDX (VBN, COUNT) : L_NORM =
;  879     1871  1
;  880     1872  1 !++
;  881     1873  1 !
;  882     1874  1 ! FUNCTIONAL DESCRIPTION:
;  883     1875  1 !
;  884     1876  1 !       This routine maps a virtual block in the index file.
;  885     1877  1 !
;  886     1878  1 ! CALLING SEQUENCE:
;  887     1879  1 !       MAP_IDX (ARG1, ARG2)
;  888     1880  1 !
;  889     1881  1 ! INPUT PARAMETERS:
;  890     1882  1 !       ARG1: VBN of block to map
;  891     1883  1 !
;  892     1884  1 ! IMPLICIT INPUTS:
;  893     1885  1 !       NONE
;  894     1886  1 !
;  895     1887  1 ! OUTPUT PARAMETERS:
;  896     1888  1 !       COUNT: (optional) address to store count of contiguous blocks
;  897     1889  1 !
;  898     1890  1 ! IMPLICIT OUTPUTS:
;  899     1891  1 !       NONE
;  900     1892  1 !
;  901     1893  1 ! ROUTINE VALUE:
;  902     1894  1 !       LBN of blocks mapped or -1 if failure
;  903     1895  1 !
;  904     1896  1 ! SIDE EFFECTS:
;  905     1897  1 !       NONE
;  906     1898  1 !
;  907     1899  1 !--
;  908     1900  1
;  909     1901  2 BEGIN
;  910     1902  2
;  911     1903  2 EXTERNAL ROUTINE
;  912     1904  2         MAP_VBN         : L_NORM,       ! map VBN and turn window if necessary
;  913     1905  2         MAP_WINDOW      : L_NORM,       ! map VBN with current window
;  914     1906  2         RELEASE_SERIAL_LOCK : L_NORM,   ! release sync lock on file
;  915     1907  2         SERIAL_FILE     : L_NORM;       ! get sync lock on file
;  916     1908  2
;  917     1909  2 LOCAL
;  918     1910  2         INCOMPLETE_FLAG,                ! Saved state of CLF_INCOMPLETE
;  919     1911  2         IDX_FCB         : REF BBLOCK,   ! address of index file FCB
;  920     1912  2         LBN,                            ! resulting LBN from map
;  921     1913  2         UNMAPPED,                       ! received count of unmapped blocks
;  922     1914  2         TEMP;                           ! dummy to store resulting UCB
;  923     1915  2
;  924     1916  2 BIND_COMMON;
;  925     1917  2
;  926     1918  2 ! Try to map with the existing window first. This can be done without
;  927     1919  2 ! taking out the sync lock on the index file.
;  928     1920  2 !
;  929     1921  2
;  930     1922  2 IDX_FCB = .CURRENT_VCB [VCB$L_FCBFL];
;  931     1923  2
;  932     1924  3 IF (LBN = MAP_WINDOW (.VBN, .IDX_FCB [FCB$L_WLFL], 1000, UNMAPPED, TEMP))
;  933     1925  2     EQL -1
;  934     1926  2 THEN
```

```
;  935          1927  3      BEGIN
;  936          1928  3      TEMP = .CURR_LCKINDX;
;  937          1929  3      SERIAL_FILE (IDX_FCB [FCB$W_FID]);
;  938          1930  3      INCOMPLETE_FLAG = .CLEANUP_FLAGS[CLF_INCOMPLETE];   ! Save current state
;  939          1931  3      IDX_FCB [FCB$V_STALE] = 1;
;  940          1932  3      LBN = MAP_VBN (.VBN, .IDX_FCB [FCB$L_WLFL], 1000, UNMAPPED);
;  941          1933  3      CLEANUP_FLAGS[CLF_INCOMPLETE] = .INCOMPLETE_FLAG;   ! Restore saved state
;  942          1934  3
;  943          1935  3      IF .TEMP NEQ .CURR_LCKINDX
;  944          1936  3      THEN
;  945          1937  4          BEGIN
;  946          1938  4          RELEASE_SERIAL_LOCK (.CURR_LCKINDX);
;  947          1939  4          CURR_LCKINDX = .TEMP;
;  948          1940  3          END;
;  949          1941  3
;  950          1942  2      END;
;  951          1943  2
;  952          1944  2 ! Return the block count if asked for.
;  953          1945  2 !
;  954          1946  2
;  955          1947  2 IF ACTUALCOUNT GEQU 2
;  956          1948  2 THEN .COUNT = 1000 - .UNMAPPED;
;  957          1949  2 .LBN
;  958          1950  2
;  959          1951  1 END;                                    ! of routine MAP_IDX


                                                .EXTRN  MAP_VBN, MAP_WINDOW

                                   001C 00000   .ENTRY  MAP_IDX, Save R2,R3,R4                    ;  1870
                              5E        08 C2   00002   SUBL2  #8, SP
                              52   98   BA DO   00005   MOVL   @-104(BASE), IDX_FCB               ;  1922
                                   5E   DD     00009   PUSHL  SP                                  ;  1924
                                   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 DO  0001E   MOVL   R0, LBN
                    FFFFFFFF  8F         54 D1  00021   CMPL   LBN, #-1                           ;  1925
                                         42 12  00028   BNEQ   1$
                              6E   14   AA DO   0002A   MOVL   20(BASE), TEMP                    ;  1928
                                   24   A2 9F  0002E   PUSHAB 36(IDX_FCB)                       ;  1929
                         0000G CF        01 FB  00031   CALLS  #1, SERIAL_FILE
53             6A             01        0A EF  00036   EXTZV  #10, #1, (BASE), INCOMPLETE_FLAG   ;  1930
                        23   A2         01 88  0003B   BISB2  #1, 35(IDX_FCB)                    ;  1931
                                   04   AE 9F  0003F   PUSHAB UNMAPPED                           ;  1932
                              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 DO  00052   MOVL   R0, LBN
6A             01             0A         53 F0  00055   INSV   INCOMPLETE_FLAG, #10, #1, (BASE)  ;  1933
                        14   AA         6E D1  0005A   CMPL   TEMP, 20(BASE)                     ;  1935
                                         0C 13  0005E   BEQL   1$
                                   14   AA DD  00060   PUSHL  20(BASE)                           ;  1938
```

```
                          0000G  CF            01  FB 00063          CALLS   #1, RELEASE_SERIAL_LOCK                ;
                            14   AA            6E  D0 00068          MOVL    TEMP, 20(BASE)                        ; 1939
                                 02            6C  91 0006C 1$:      CMPB    (AP), #2                              ; 1947
                                               0A  1F 0006F          BLSSU   2$                                    ;
                      08   BC 000003E8  8F       04  AE C3 00071     SUBL3   UNMAPPED, #1000, @COUNT               ; 1948
                                 50            54  D0 00078 2$:      MOVL    LBN, R0                               ; 1951
                                               04  00U7E            RET                                           ;

; Routine Size:  127 bytes,    Routine Base:  $CODE$ + 04D0

;  960          1952  ;
;  961          1953  1 END
;  962          1954  0 ELUDOM



;                                    PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $CODE$                         1359  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                                Library Statistics
;
;                                        --------- Symbols ---------      Pages       Processing
;       File                             Total    Loaded   Percent       Mapped      Time
;
;  _$255$DUA18:[SYSLIB]LIB.L32;1          18619        67         0       1000          00:02.0



;                                    COMMAND QUALIFIERS

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

; Size:         1352 code + 7 data bytes
; Run Time:        01:05.5
; Elapsed Time:    01:31.3
; Lines/CPU Min:    1791
; Lexemes/CPU-Min: 54177
; Memory Used:  337 pages
; Compilation Complete
```

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