H 5

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

```
_$25

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

KILL
KILL
LB_E
LB_D
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

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

MAKE
MAKE
MAP_
MAP_

MAP
MARK
MARK
MARK
MARK
MARK
```

```
**FILE**ID**SMALOC                                  L  3
```

```
  SSSSSSSS  MM      MM    AAAAAA    LL            000000      CCCCCCCC
  SSSSSSSS  MM      MM    AAAAAA    LL            000000      CCCCCCCC
SS          MMMM  MMMM  AA      AA  LL          00      00  CC
SS          MMMM  MMMM  AA      AA  LL          00      00  CC
SS          MM  MM  MM  AA      AA  LL          00      00  CC
SS          MM  MM  MM  AA      AA  LL          00      00  CC
  SSSSSS    MM      MM  AA      AA  LL          00      00  CC
  SSSSSS    MM      MM  AA      AA  LL          00      00  CC
        SS  MM      MM  AAAAAAAAAA  LL          00      00  CC
        SS  MM      MM  AAAAAAAAAA  LL          00      00  CC
        SS  MM      MM  AA      AA  LL          00      00  CC              ....
        SS  MM      MM  AA      AA  LL          00      00  CC              ....
SSSSSSSS    MM      MM  AA      AA  LLLLLLLLLL    000000      CCCCCCCC      ....
SSSSSSSS    MM      MM  AA      AA  LLLLLLLLLL    000000      CCCCCCCC      ....

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

```
0 MODULE SMALOC (
0                LANGUAGE (BLISS32),
0                IDENT = 'V04-000'
0                ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  F11ACP Structure Level 2
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains the routines that manipulate the volume
1 !       storage bitmap. These include the routines to allocate a contiguous
1 !       area, deallocate an area, and the basic bitmap scanner.
1 !       Also included are the routines that manage the extent cache.
1 !
1 ! ENVIRONMENT:
1 !
1 !       STARLET operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  21-Feb-1977  18:42
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-012 ACG0445         Andrew C. Goldstein,    21-Aug-1984  20:48
1 !               Fix handling of null extent cache in RETURN_BLOCKS
1 !
1 !       V03-011 ACG0438         Andrew C. Goldstein,    1-Aug-1984  18:51
```

```
1 !                    Add extent cache interlock logic; remove kernel calls,
1 !                    fold in UPDATE_FREE and SET_SMVBN routines. Use central
1 !                    dequeue routine.
1 !
1 !          V03-010 LMP0257          L. Mark Pilant,         25-Jun-1984  9:42
1 !                    Use double precision when calculation cluster round-up to
1 !                    insure that the cluster calculation is unsigned.
1 !
1 !          V03-009 CDS0004          Christian D. Saether    29-Dec-1983
1 !                    Use L_NORM linkage and BIND_COMMON macro.
1 !
1 !          V03-008 CDS0003          Christian D. Saether    25-Sep-1983
1 !                    Manually merge in STJ3106.
1 !
1 !          V03-007 STJ3106          Steven T. Jeffreys,     20-Jun-1983
1 !                    - Implement Erase On Extend (EOE).
1 !
1 !          V03-006 CDS0002          Christian D. Saether    13-Sep-1983
1 !                    Change interface to allocation serialization routine.
1 !
1 !          V03-005 CDS0001          Christian D. Saether    13-May-1983
1 !                    Serialize storage allocation/deallocation activity.
1 !
1 !          V03-004 STJ3081          Steven T. Jeffreys,     30-Mar-1983
1 !                    - Added CHANNEL parameter to ERASE_BLOCKS call.
1 !
1 !          V03-003 STJ3062          Steven T. Jeffreys,     18-Mar-1982
1 !                    - Added call to ERASE_BLOCKS from RETURN_BLOCKS.
1 !                    - Added ERASE_REQUESTED parameter to RETURN_BLOCKS.
1 !
1 !          V03-002 ACG0298          Andrew C. Goldstein,    25-Aug-1982  16:32
1 !                    Detect attempts to create negative extent cache entries
1 !
1 !          V03-001 ACG45949         Andrew C. Goldstein,    8-Jun-1982  16:11
1 !                    Prevent volume free space from going negative
1 !
1 !          V02-014 ACG43131         Andrew C. Goldstein,    4-Jan-1982  18:11
1 !                    Fix spurious allocation failures in approx. placed allocation
1 !
1 !          V02-013 ACG0229          Andrew C. Goldstein,    23-Dec-1981  22:10
1 !                    Count extent cache hits and misses
1 !
1 !          V02-012 ACG38789         Andrew C. Goldstein,    1-Jul-1981  19:48
1 !                    Check for running out bit count in cylinder round up
1 !
1 !          V02-011 ACG0195          Andrew C. Goldstein,    3-Mar-1981  22:54
1 !                    Fix 4096 block boundary problem by checking zero in BITSCAN
1 !
1 !          V02-010 ACG0180          Andrew C. Goldstein,    10-Sep-1980  14:44
1 !                    Fix cluster and cylinder rounding in extent cache allocator
1 !
1 !          V02-009 ACG0172          Andrew C. Goldstein,    9-May-1980  10:42
1 !                    Check map pointer count for non-zero in RETURN_BLOCKS
1 !
1 !          V02-008 ACG0167          Andrew C. Goldstein,    16-Apr-1980  19:28
1 !                    Previous revision history moved to F11B.REV
1 !**
```

```
1
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32';
1 REQUIRE 'SRC$:FCPDEF.B32';
1
1
1 !
1 ! Modes of operation of the bit scanner.
1 !
1
1 LITERAL
1         FIND_SET        = 0,            ! find first one
1         FIND_CLEAR      = 1,            ! find first zero
1         SET_BITS        = 2,            ! set n bits
1         CLEAR_BITS      = 3;            ! clear n bits
1
1
1 FORWARD ROUTINE
1         ALLOC_BLOCKS    : L_NORM,
1         RETURN_BLOCKS   : L_NORM NOVALUE,
1         INIT_EXT_CACHE  : L_NORM NOVALUE, ! set up extent cache lock
1         ALLOC_EXTENT    : L_NORM,       ! allocate entry from extent cache
1         RETURN_EXTENT   : L_NORM,       ! return entry to extent cache
1         PURGE_EXTENT    : L_NORM NOVALUE, ! return cache entries back to bitmap
1         REMOVE_EXTENT   : L_NORM,       ! remove entry from extent cache
1         ALLOC_BITMAP    : L_NORM,       ! allocate blocks from storage bitmap
1         RETURN_BITMAP   : L_NORM NOVALUE, ! return blocks to storage bitmap
1         BITSCAN         : L_NORM;
```

```
1 GLOBAL ROUTINE ALLOC_BLOCKS (FIB, BLOCKS_NEEDED, START_LBN, BLOCKS_ALLOC) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine allocates a single contiguous area of disk. It first
1 !       attempts allocation from the extent cache. If that fails, it performs
1 !       the allocation from the storage bitmap.
1 !
1 !       As part of system security, the blocks allocated will be erased
1 !       before returning the extent to the caller.
1 !
1 ! CALLING SEQUENCE:
1 !       ALLOC_BLOCKS (ARG1, ARG2, ARG3, ARG4)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of FIB for this operation
1 !       ARG2: number of blocks to allocate
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !       CURRENT_UCB: UCB of volume
1 !
1 ! OUTPUT PARAMETERS:
1 !       ARG3: address of longword to store starting LBN
1 !       ARG4: address of longword to store block count
1 !
1 ! IMPLICIT OUTPUTS:
1 !       LOC_LBN: plcement LBN of allocation or 0
1 !
1 ! ROUTINE VALUE:
1 !       1 if successful allocation
1 !       0 if failure
1 !
1 ! SIDE EFFECTS:
1 !       storage map, VCB, and extent cache modified
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       FIB             : REF BBLOCK;   ! FIB of operation
2
2 LITERAL
2       ALLOC_RETRIES   = 3;            ! Number of times to retry allocation
2
2 LOCAL
2       ERASED,                         ! status of erase operation
2       ATTEMPTS,                       ! number of attempts at cache allocation
2       STATUS,                         ! status return value
2       CACHE           : REF BBLOCK,   ! pointer to main cache block
2       EXTENT_CACHE    : REF BBLOCK,   ! pointer to extent cache
2       TEMP            : VECTOR [2],   ! quadword temp for EMUL & EDIV
2       EXT_LIMIT,                      ! local longword copy of extent limit parameter
2       DUMMY,                          ! dummy to receive remainder from EDIV
```

```
2               CACHE_TOTAL,                    ! total disk space to allocate into cache
2               LBN,                            ! LBN being allocated
2               COUNT;                          ! block count being allocated
2
2       BIND
2               DUMMY_FIB       = UPLIT (REP FIB$C_EXTDATA OF (BYTE (0)));
2                                               ! default FIB for allocation for cache
2
2       BIND_COMMON;
2
2       EXTERNAL ROUTINE
2               ALLOCATION_LOCK : L_NORM,       ! serialize allocation/deallocation
2               ERASE_BLOCKS    : L_NORM,       ! Erase blocks before reusing them
2               ALLOCATION_UNLOCK : L_NORM NOVALUE, ! release allocation lock.
2               RELEASE_LOCKBASIS : L_NORM,     ! release buffers under specified lock
2               DEQ_LOCK        : L_NORM,       ! dequeue a lock
2               CACHE_LOCK      : L_NORM;       ! acquire cache sync lock
2
2       EXTERNAL
2               PMS$GL_EXTHIT   : ADDRESSING_MODE (GENERAL),
2                                               ! count of extent cache hits
2               PMS$GL_EXTMISS  : ADDRESSING_MODE (GENERAL);
2                                               ! count of extent cache misses
2
2       ! Serialize processing against other storage/header allocation/deallocation.
2       !
2
2       ALLOCATION_LOCK ();
2
2       ! First attempt to allocate the space from the extent cache. Note that
2       ! a placed allocation can actually split a cache entry; therefore, if the
2       ! cache is full after the allocation, purge it to half.
2       !
2       CACHE = .CURRENT_VCB[VCB$L_CACHE];
2       EXTENT_CACHE = .CACHE[VCA$L_EXTCACHE];
3       IF (STATUS = ALLOC_EXTENT (.FIB, .BLOCKS_NEEDED, .START_LBN, .BLOCKS_ALLOC))
2       THEN
3           BEGIN
3           IF .EXTENT_CACHE[VCA$W_EXTCOUNT] GEQU .EXTENT_CACHE[VCA$W_EXTSIZE]
3           THEN
4               BEGIN
4               PMS$GL_EXTMISS = .PMS$GL_EXTMISS + 1;
4               PURGE_EXTENT (.EXTENT_CACHE[VCA$W_EXTSIZE] / 2, -1);
4               END
3           ELSE
3               PMS$GL_EXTHIT = .PMS$GL_EXTHIT + 1;
3           END
3
3       ! If the cache allocation failed, attempt allocation from the bitmap.
3       ! If this fails,  purge the cache if there is anything in it, to make
3       ! the bitmap consistent. Then attempt allocation from the bitmap again.
3       !
3
2       ELSE
3           BEGIN
3           PMS$GL_EXTMISS = .PMS$GL_EXTMISS + 1;
3
```

```
3        DECR J FROM 2 TO 1
3        DO
4            BEGIN
5            IF (STATUS = ALLOC_BITMAP (.FIB, .BLOCKS_NEEDED, .START_LBN, .BLOCKS_ALLOC, 0))
4            THEN EXITLOOP;
4
4    ! Can't get the space from the bitmap as is. Purge back the extent cache,
4    ! and, if we're in a cluster, ask for a flush of all others and try
4    ! once more.
4    !
4
4            PURGE_EXTENT (0, 0);
4            IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
4            THEN
5                BEGIN
5                LOCAL BIT_FILE_ID, LOCK_ID, STATUS;
5                RELEASE_LOCKBASIS (-1);
5                ALLOCATION_UNLOCK ();
5                BIT_FILE_ID = FID$C_BITMAP OR .CURRENT_VCB[VCB$W_RVN] ^ 24;
5                LOCK_ID = 0;
5                CACHE_LOCK (.BIT_FILE_ID, LOCK_ID, 1);
5                ALLOCATION_LOCK ();
5                DEQ_LOCK (.LOCK_ID);
4                END;
3            END;
3
3    ! If extent caching is not shut off, now refill the cache from the
3    ! bitmap block currently in memory.
3    !
3
3        IF NOT .CACHE[VCA$V_EXTC_VALID]
3        THEN INIT_EXT_CACHE (.CACHE);
3
3        IF .CACHE[VCA$V_EXTC_VALID]
3        THEN
4            BEGIN
4            LOC_LBN = 0;                        ! discard placement
4            EXT_LIMIT = .EXTENT_CACHE[VCA$W_EXTLIMIT];
4            EMUL (EXT_LIMIT, CURRENT_VCB[VCB$L_FREE], %REF (0), TEMP);
4            EDIV (%REF (1000), TEMP, CACHE_TOTAL, DUMMY);
4            UNTIL .EXTENT_CACHE[VCA$W_EXTCOUNT] GEQU .EXTENT_CACHE[VCA$W_EXTSIZE]/2
4            DO
5                BEGIN
5                IF NOT ALLOC_BITMAP (DUMMY_FIB, .CACHE_TOTAL, LBN, COUNT, 1)
5                THEN EXITLOOP;
5                RETURN_EXTENT (.LBN, .COUNT);
5                CACHE_TOTAL = .CACHE_TOTAL - .COUNT;
5                IF .CACHE_TOTAL LEQ 0
5                THEN EXITLOOP;
4                END;
3            END;
2        END;                                    ! end of bitmap processing conditional
2
2    ! If we successfully allocated something, erase the space if called for
2    ! and deduct it from the volume's free space.
2    !
2
```

```
;  315        1304  2 IF .STATUS
;  316        1305  2 THEN
;  317        1306  3     BEGIN
;  318        1307  3     IF NOT .CURRENT_VCB[VCB$V_NOHIGHWATER]
;  319        1308  3     THEN ERASE_BLOCKS (..START_LBN, ..BLOCKS_ALLOC, .IO_CHANNEL);
;  320        1309  3     CURRENT_VCB[VCB$L_FREE] = .CURRENT_VCB[VCB$L_FREE] - ..BLOCKS_ALLOC;
;  321        1310  3     IF .CURRENT_VCB[VCB$L_FREE] LSS 0
;  322        1311  3     THEN CURRENT_VCB[VCB$L_FREE] = 0;
;  323        1312  2     END;
;  324        1313  2
;  325        1314  2 RETURN .STATUS;
;  326        1315  2
;  327        1316  1 END;                                  ! end of routine ALLOC_BLOCKS


                                                    .TITLE  SMALOC
                                                    .IDENT  \V04-000\

                                                    .PSECT  $CODE$,NOWRT,2

                                00  00000 P.AAA:    .BYTE   0                                          ;
                                00  00001           .BYTE   0                                          ;
                                00  00002           .BYTE   0                                          ;
                                00  00003           .BYTE   0                                          ;
                                00  00004           .BYTE   0                                          ;
                                00  00005           .BYTE   0                                          ;
                                00  00006           .BYTE   0                                          ;
                                00  00007           .BYTE   0                                          ;
                                00  00008           .BYTE   0                                          ;
                                00  00009           .BYTE   0                                          ;
                                00  0000A           .BYTE   0                                          ;
                                00  0000B           .BYTE   0                                          ;
                                00  0000C           .BYTE   0                                          ;
                                00  0000D           .BYTE   0                                          ;
                                00  0000E           .BYTE   0                                          ;
                                00  0000F           .BYTE   0                                          ;
                                00  00010           .BYTE   0                                          ;
                                00  00011           .BYTE   0                                          ;
                                00  00012           .BYTE   0                                          ;
                                00  00013           .BYTE   0                                          ;
                                00  00014           .BYTE   0                                          ;
                                00  00015           .BYTE   0                                          ;
                                00  00016           .BYTE   0                                          ;
                                00  00017           .BYTE   0                                          ;
                                00  00018           .BYTE   0                                          ;
                                00  00019           .BYTE   0                                          ;
                                00  0001A           .BYTE   0                                          ;
                                00  0001B           .BYTE   0                                          ;
                                00  0001C           .BYTE   0                                          ;
                                00  0001D           .BYTE   0                                          ;
                                00  0001E           .BYTE   0                                          ;
                                00  0001F           .BYTE   0                                          ;

                                          DUMMY_FIB=        P.AAA
                                                    .EXTRN  ALLOCATION_LOCK
                                                    .EXTRN  ERASE_BLOCKS, ALLOCATION_UNLOCK
                                                    .EXTRN  RELEASE_LOCKBASIS
```

```
                                                             .EXTRN  DEQ_LOCK, CACHE_LOCK
                                                             .EXTRN  PMS$GL_EXTHIT, PMS$GL_EXTMISS

                                      00FC 00000             .ENTRY  ALLOC_BLOCKS, Save R2,R3,R4,R5,R6,R7          ; 1133
                57 00000000G   00     9E 00002             MOVAB   PMS$GL_EXTMISS, R7                              ;
                5E             14     C2 00009             SUBL2   #20, SP                                         ;
                56       98    AA     9E 0000C             MOVAB   -104(BASE), R6                                  ; 1195
          0000G CF             00     FB 00010             CALLS   #0, ALLOCATION_LOCK                             ; 1217
                50             66     D0 00015             MOVL    (R6), R0                                        ; 1223
                53       58    A0     D0 00018             MOVL    88(R0), CACHE                                   ;
                52       04    A3     D0 0001C             MOVL    4(CACHE), EXTENT_CACHE                          ; 1224
                7E       0C    AC     7D 00020             MOVQ    START_LBN, -(SP)                                ; 1225
                7E       04    AC     7D 00024             MOVQ    FIB, -(SP)                                      ;
          0000V CF             04     FB 00028             CALLS   #4, ALLOC_EXTENT                                ;
                55             50     D0 0002D             MOVL    R0, STATUS                                      ;
                22             55     E9 00030             BLBC    STATUS, 3$                                      ;
                62       02    A2     B1 00033             CMPW    2(EXTENT_CACHE), (EXTENT_CACHE)                 ; 1228
                               13     1F 00037             BLSSU   1$                                              ;
                               67     D6 00039             INCL    PMS$GL_EXTMISS                                  ; 1231
                7E             01     CE 0003B             MNEGL   #1, -(SP)                                       ; 1232
                50             62     3C 0003E             MOVZWL  (EXTENT_CACHE), R0                              ;
  7E            50             02     C7 00041             DIVL3   #2, R0, -(SP)                                   ;
          0000V CF             02     FB 00045             CALLS   #2, PURGE_EXTENT                                ;
                               06     11 0004A             BRB     2$                                              ; 1228
                  00000000G    00     D6 0004C  1$:        INCL    PMS$GL_EXTHIT                                   ; 1235
                             00C2     31 00052  2$:        BRW     9$                                              ; 1225
                               67     D6 00055  3$:        INCL    PMS$GL_EXTMISS                                  ; 1245
                54             02     D0 00057             MOVL    #2, J                                           ; 1247
                               7E     D4 0005A  4$:        CLRL    -(SP)                                           ; 1250
                7E       0C    AC     7D 0005C             MOVQ    START_LBN, -(SP)                                ;
                7E       04    AC     7D 00060             MOVQ    FIB, -(SP)                                      ;
          0000V CF             05     FB 00064             CALLS   #5, ALLOC_BITMAP                                ;
                55             50     D0 00069             MOVL    R0, STATUS                                      ;
                47             55     E8 0006C             BLBS    STATUS, 6$                                      ;
                               7E     7C 0006F             CLRQ    -(SP)                                           ; 1258
          0000V CF             02     FB 00071             CALLS   #2, PURGE_EXTENT                                ;
                50       94    AA     D0 00076             MOVL    -108(BASE), R0                                  ; 1259
                35       3C    A0     E9 0007A             BLBC    60(R0), 5$                                      ;
                7E             01     CE 0007E             MNEGL   #1, -(SP)                                       ; 1263
          0000G CF             01     FB 00081             CALLS   #1, RELEASE_LOCKBASIS                           ;
          0000G CF             00     FB 00086             CALLS   #0, ALLOCATION_UNLOCK                           ; 1264
                50             66     D0 0008B             MOVL    (R6), R0                                        ; 1265
                50       0E    A0     3C 0008E             MOVZWL  14(R0), R0                                      ;
  50            50             18     78 00092             ASHL    #24, R0, R0                                     ;
                50             02     88 00096             BISB2   #2, BIT_FILE_ID                                 ;
                               6E     D4 00099             CLRL    LOCK_ID                                         ; 1266
                               01     DD 0009B             PUSHL   #1                                              ; 1267
                         04    AE     9F 0009D             PUSHAB  LOCK_ID                                         ;
                               50     DD 000A0             PUSHL   BIT_FILE_ID                                     ;
          0000G CF             03     FB 000A2             CALLS   #3, CACHE_LOCK                                  ;
          0000G CF             00     FB 000A7             CALLS   #0, ALLOCATION_LOCK                             ; 1268
                               6E     DD 000AC             PUSHL   LOCK_ID                                         ; 1269
          0000G CF             01     FB 000AE             CALLS   #1, DEQ_LOCK                                    ;
                A4             54     F5 000B3  5$:        SOBGTR  J, 4$                                           ; 1247
  0C       0B   A3             01     E0 000B6  6$:        BBS     #1, 11(CACHE), 7$                               ; 1277
                               53     DD 000BB             PUSHL   CACHE                                           ; 1278
          0000V CF             01     FB 000BD             CALLS   #1, INIT_EXT_CACHE                              ;
```

```
                         50      0B  A3              01  E1 000C2         BBC      #1, 11(CACHE), 9$                                   ; 1280
                                                20   AA  D4 000C7 7$:     CLRL     32(BASE)                                            ; 1283
                                   51           08   A2  3C 000CA         MOVZWL   8(EXTENT_CACHE), EXT_LIMIT                          ; 1284
                                   50                66  D0 000CE         MOVL     (R6), R0                                            ; 1285
     0C   AE             00    40  A0                51  7A 000D1         EMUL     EXT_LIMIT, 64(R0), #0, TEMP                         ;
          50             53    0C  AE 000003E8       8F  7B 000D8         EDIV     #1000, TEMP, CACHE_TOTAL, DUMMY                     ; 1286
                                   50                62  3C 000E2 8$:     MOVZWL   (EXTENT_CACHE), R0                                  ; 1287
                                   50                02  C6 000E5         DIVL2    #2, R0                                              ;
          50       02    A2        10                00  ED 000E8         CMPZV    #0, #16, 2(EXTENT_CACHE), R0                        ;
                                                     27  1E 000EE         BGEQU    9$                                                  ;
                                                     01  DD 000F0         PUSHL    #1                                                  ; 1290
                                                08   AE  9F 000F2         PUSHAB   COUNT                                               ;
                                                10   AE  9F 000F5         PUSHAB   LBN                                                 ;
                                                     53  DD 000F8         PUSHL    CACHE_TOTAL                                         ;
                                              FEE2   CF  9F 000FA         PUSHAB   DUMMY_FIB                                           ;
                              0000V  CF              05  FB 000FE         CALLS    #5, ALLOC_BITMAP                                    ;
                                     11              50  E9 00103         BLBC     R0, 9$                                              ;
                                                04   AE  DD 00106         PUSHL    COUNT                                               ; 1292
                                                0C   AE  DD 00109         PUSHL    LBN                                                 ;
                              0000V  CF              02  FB 0010C         CALLS    #2, RETURN_EXTENT                                   ;
                                     53         04   AE  C2 00111         SUBL2    COUNT, CACHE_TOTAL                                  ; 1293
                                                     CB  14 00115         BGTR     8$                                                  ; 1294
                                     2A              55  E9 00117 9$:     BLBC     STATUS, 11$                                         ; 1304
                                     50              66  D0 0011A         MOVL     (R6), R0                                            ; 1307
                         0F      53  A0              04  E0 0011D         BBS      #4, 83(R0), 10$                                     ;
                                              FF78   CA  DD 00122         PUSHL    -136(BASE)                                          ; 1308
                                                10   BC  DD 00126         PUSHL    @BLOCKS_ALLOC                                       ;
                                                0C   BC  DD 00129         PUSHL    @START_LBN                                          ;
                              0000G  CF              03  FB 0012C         CALLS    #3, ERASE_BLOCKS                                    ;
                                     50              66  D0 00131 10$:    MOVL     (R6), R0                                            ; 1309
                                 40  A0         10   BC  C2 00134         SUBL2    @BLOCKS_ALLOC, 64(R0)                               ;
                                     50              66  D0 00139         MOVL     (R6), R0                                            ; 1310
                                                40   A0  D5 0013C         TSTL     64(R0)                                              ;
                                                     03  18 0013F         BGEQ     11$                                                 ;
                                                40   A0  D4 00141         CLRL     64(R0)                                              ; 1311
                                     50              55  D0 00144 11$:    MOVL     STATUS, R0                                          ; 1314
                                                         04 00147         RET                                                          ; 1316

; Routine Size:  328 bytes,    Routine Base:  $CODE$ + 0020
```

```
1 GLOBAL ROUTINE RETURN_BLOCKS (START_LBN, BLOCK_COUNT, ERASE_REQUESTED) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine returns a single contiguous area to the storage pool.
1 !       If there is space in the cache, the blocks are simply returned to
1 !       the cache. If the cache is full, if first purges some of the cache
1 !       entries and then returns the blocks.
1 !
1 ! CALLING SEQUENCE:
1 !       RETURN_BLOCKS (ARG1, ARG2, ARG3)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: starting LBN to free
1 !       ARG2: number of blocks to free
1 !       ARG3: boolean.  1 if blocks are to be erased, 0 if not.
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !       CURRENT_UCB: UCB of device
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       storage map, VCB, and extent cache modified
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       STATUS,                                 ! local storage for routine status
2       CACHE           : REF BBLOCK,           ! pointer to main cache block
2       EXTENT_CACHE    : REF BBLOCK,           ! pointer to extent cache
2       TEMP            : VECTOR [2],           ! quadword temp for EMUL & EDIV
2       EXT_LIMIT,                              ! local longword copy of extent limit parameter
2       DUMMY,                                  ! dummy to receive remainder from EDIV
2       CACHE_LIMIT;                            ! total disk space to allocate into cache
2
2 BIND_COMMON;
2
2 EXTERNAL
2       PMS$GL_EXTHIT   : ADDRESSING_MODE (GENERAL),
2                                               ! count of extent cache hits
2       PMS$GL_EXTMISS  : ADDRESSING_MODE (GENERAL);
2                                               ! count of extent cache misses
2
2 EXTERNAL ROUTINE
```

```
2         ALLOCATION_LOCK : L_NORM,
2         ERASE_BLOCKS    : L_NORM;       ! Erase blocks before reusing them
2
2
2 ! First check the block count for non-zero.
2 !
2
2 IF .BLOCK_COUNT EQL 0
2 THEN ERR_EXIT (SS$_BADFILEHDR);
2
2 ! Check the blocks being returned against the volume size.
2 !
2
2 IF .START_LBN + .BLOCK_COUNT GTRU .CURRENT_UCB[UCB$L_MAXBLOCK]
2 THEN ERR_EXIT (SS$_BADFILEHDR);
2
2 ! Check that the start LBN and count are integral multiples of the
2 ! cluster factor. If not, reject the operation on grounds of a bad
2 ! file header.
2 !
2
2 IF .START_LBN MOD .CURRENT_VCB[VCB$W_CLUSTER] NEQ 0
2 OR .BLOCK_COUNT MOD .CURRENT_VCB[VCB$W_CLUSTER] NEQ 0
2 THEN ERR_EXIT (SS$_BADFILEHDR);
2
2 ! Before returning the blocks, erase them if need be.
2 ! Notify the user if an error is encountered.
2 !
2
2 IF .ERASE_REQUESTED
2 THEN
3     IF NOT (STATUS = ERASE_BLOCKS (.START_LBN, .BLOCK_COUNT, .IO_CHANNEL))
2     THEN
2         ERR_STATUS (.STATUS);
2
2 ! Serialize processing against other storage/header allocation/deallocation.
2 !
2
2 ALLOCATION_LOCK();
2
2 ! Attempt to activate the extent cache if it is not active. If it refuses
2 ! to activate (e.g., is null, or is inhibited due to interlocks), return
2 ! the space directly to the bitmap.
2 !
2
2 CACHE = .CURRENT_VCB[VCB$L_CACHE];
2 EXTENT_CACHE = .CACHE[VCA$L_EXTCACHE];
2
2 IF NOT .CACHE[VCA$V_EXTC_VALID]
2 THEN INIT_EXT_CACHE (.CACHE);
2
2 IF NOT .CACHE[VCA$V_EXTC_VALID]
2 THEN
3     BEGIN
3     RETURN_BITMAP (.START_LBN, .BLOCK_COUNT);
3     PMS$GL_EXTMISS = .PMS$GL_EXTMISS + 1;
3     END
```

```
;  443              1431 3
;  444              1432 3 ! Return the blocks to the cache. If the cache is full or if it now contains
;  445              1433 3 ! more space than we want, then purge it to half and/or below the limit.
;  446              1434 3 !
;  447              1435 3
;  448              1436 2 ELSE
;  449              1437 3     BEGIN
;  450              1438 3     IF NOT RETURN_EXTENT (.START_LBN, .BLOCK_COUNT)
;  451              1439 3     THEN ERR_EXIT (SS$_BADFILEHDR);
;  452              1440 3
;  453              1441 3     EXT_LIMIT = .EXTENT_CACHE[VCA$W_EXTLIMIT];
;  454              1442 3     EMUL (EXT_LIMIT, CURRENT_VCB[VCB$L_FREE], %REF (0), TEMP);
;  455              1443 3     EDIV (%REF (1000), TEMP, CACHE_LIMIT, DUMMY);
;  456              1444 3     IF .EXTENT_CACHE[VCA$W_EXTCOUNT] GEQU .EXTENT_CACHE[VCA$W_EXTSIZE]
;  457              1445 3     OR .EXTENT_CACHE[VCA$L_EXTTOTAL] GTRU .CACHE_LIMIT
;  458              1446 3     THEN
;  459              1447 4         BEGIN
;  460              1448 4         PURGE_EXTENT (.EXTENT_CACHE[VCA$W_EXTSIZE] / 2, .CACHE_LIMIT);
;  461              1449 4         PMS$GL_EXTMISS = .PMS$GL_EXTMISS + 1;
;  462              1450 4         END
;  463              1451 3     ELSE
;  464              1452 3         PMS$GL_EXTHIT = .PMS$GL_EXTHIT + 1;
;  465              1453 2     END;
;  466              1454 2
;  467              1455 2 CURRENT_VCB[VCB$L_FREE] = .CURRENT_VCB[VCB$L_FREE] + .BLOCK_COUNT;
;  468              1456 2
;  469              1457 1 END;                                        ! end of routine RETURN_BLOCKS



                                              000C 00000         .ENTRY  RETURN_BLOCKS, Save R2,R3                              ; 1317
                         5E               08    C2 00002         SUBL2   #8, SP                                                 ;
                                    08    AC    D5 00005         TSTL    BLOCK_COUNT                                            ; 1381
                                          03    12 00008         BNEQ    1$                                                     ;
                                        0090    31 0000A         BRW     4$                                                     ;
              51         04   AC    08    AC    C1 0000D 1$:     ADDL3   BLOCK_COUNT, START_LBN, R1                             ; 1387
                              50    94    AA    D0 00013         MOVL    -108(BASE), R0                                         ;
                    00B0      C0          51    D1 00017         CMPL    R1, 176(R0)                                            ;
                                          7F    1A 0001C         BGTRU   4$                                                     ;
                              50    98    AA    D0 0001E         MOVL    -104(BASE), R0                                         ; 1395
                              51    3C    A0    3C 00022         MOVZWL  60(R0), R1                                             ;
   7E         00         04   AC          01    7A 00026         EMUL    #1, START_LBN, #0, -(SP)                               ;
   51         51              8E          51    7B 0002C         EDIV    R1, (SP)+, R1, R1                                      ;
                                          51    D5 00031         TSTL    R1                                                     ;
                                          68    12 00033         BNEQ    4$                                                     ;
                              50    3C    A0    3C 00035         MOVZWL  60(R0), R0                                             ; 1396
   7E         00         08   AC          01    7A 00039         EMUL    #1, BLOCK_COUNT, #0, -(SP)                             ;
   50         50              8E          50    7B 0003F         EDIV    R0, (SP)+, R0, R0                                      ;
                                          50    D5 00044         TSTL    R0                                                     ;
                                          55    12 00046         BNEQ    4$                                                     ;
                              18    0C    AC    E9 00048         BLBC    ERASE_REQUESTED, 2$                                    ; 1403
                                  FF78    CA    DD 0004C         PUSHL   -136(BASE)                                             ; 1405
                              7E    04    AC    7D 00050         MOVQ    START_LBN, -(SP)                                       ;
                    0000G     CF          03    FB 00054         CALLS   #3, ERASE_BLOCKS                                       ;
                              08          50    E8 00059         BLBS    STATUS, 2$                                             ;
```

```
                                04        80    AA  E9 0005C          BLBC    -128(BASE), 2$                      ; 1407
                          80    AA              50  B0 00060          MOVW    STATUS, -128(BASE)                  ;
                       0000G    CF              00  FB 00064  2$:     CALLS   #0, ALLOCATION_LOCK                 ; 1412
                                50        98    AA  D0 00069          MOVL    -104(BASE), R0                      ; 1419
                                52        58    A0  D0 0006D          MOVL    88(R0), CACHE                       ;
                                53        04    A2  D0 00071          MOVL    4(CACHE), EXTENT_CACHE              ; 1420
                17        0B    A2              01  E0 00075          BBS     #1, 11(CACHE), 3$                   ; 1422
                                                52  DD 0007A          PUSHL   CACHE                               ; 1423
                       0000V    CF              01  FB 0007C          CALLS   #1, INIT_EXT_CACHE                  ;
                08        0B    A2              01  E0 00081          BBS     #1, 11(CACHE), 3$                   ; 1425
                                7E        04    AC  7D 00086          MOVQ    START_LBN, -(SP)                    ; 1428
                       0000V    CF              02  FB 0008A          CALLS   #2, RETURN_BITMAP                   ;
                                                42  11 0008F          BRB     7$                                  ; 1429
                                7E        04    AC  7D 00091  3$:     MOVQ    START_LBN, -(SP)                    ; 1438
                       0000V    CF              02  FB 00095          CALLS   #2, RETURN_EXTENT                   ;
                                05              50  E8 0009A          BLBS    R0, 5$                              ;
                                        0810    8F  BF 0009D  4$:     CHMU    #2064                               ; 1439
                                                    04 000A1          RET                                         ;
                                51        08    A3  3C 000A2  5$:     MOVZWL  8(EXTENT_CACHE), EXT_LIMIT          ; 1441
                                50        98    AA  D0 000A6          MOVL    -104(BASE), R0                      ; 1442
      6E        00        40    A0              51  7A 000AA          EMUL    EXT_LIMIT, 64(R0), #0, TEMP         ;
      51        50              6E  000003E8    8F  7B 000B0          EDIV    #1000, TEMP, CACHE_LIMIT, DUMMY     ; 1443
                                63        02    A3  B1 000B9          CMPW    2(EXTENT_CACHE), (EXTENT_CACHE)     ; 1444
                                                06  1E 000BD          BGEQU   6$                                  ;
                                50        04    A3  D1 000BF          CMPL    4(EXTENT_CACHE), CACHE_LIMIT        ; 1445
                                                16  1B 000C3          BLEQU   8$                                  ;
                                                50  DD 000C5  6$:     PUSHL   CACHE_LIMIT                         ; 1448
                                50              63  3C 000C7          MOVZWL  (EXTENT_CACHE), R0                  ;
                7E              50              02  C7 000CA          DIVL3   #2, R0, -(SP)                       ;
                       0000V    CF              02  FB 000CE          CALLS   #2, PURGE_EXTENT                    ;
                                    00000000G   00  D6 000D3  7$:     INCL    PMS$GL_EXTMISS                      ; 1449
                                                06  11 000D9          BRB     9$                                  ; 1444
                                    00000000G   00  D6 000DB  8$:     INCL    PMS$GL_EXTHIT                       ; 1452
                                50        98    AA  D0 000E1  9$:     MOVL    -104(BASE), R0                      ; 1455
                          40    A0        08    AC  C0 000E5          ADDL2   BLOCK_COUNT, 64(R0)                 ;
                                                    04 000EA          RET                                         ; 1457

; Routine Size:  235 bytes,    Routine Base:  $CODE$ + 0168
```

```
1 GLOBAL ROUTINE INIT_EXT_CACHE (CACH    : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine sets up the extent cache interlock as necessary
1 !       and marks the cache valid, if this is possible, considering
1 !       dismount state of the volume and write access to the storage map.
1 !
1 ! CALLING SEQUENCE:
1 !       INIT_EXT_CACHE (CACHE)
1 !
1 ! INPUT PARAMETERS:
1 !       CACHE: pointer to main cache block
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       cache marked valid, lock taken out
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       CACHE           : REF BBLOCK;   ! pointer to cache block
2
2 LOCAL
2       EXT_CACHE       : REF BBLOCK,   ! pointer to file ID cache
2       BITMAP_FID;                     ! lock basis for index file
2
2 BIND_COMMON;
2
2 EXTERNAL ROUTINE
2       CACHE_LOCK      : L_NORM;       ! acquire special cache lock
2
2
2 ! If the cache is not currently marked valid, attempt to take out the
2 ! cache lock if we are in a cluster and may do so.
2 !
2
2 EXT_CACHE = .CACHE[VCA$L_EXTCACHE];
2 IF NOT .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_DMT]
2 AND NOT .CURRENT_VCB[VCB$V_WRITE_SM]
2 AND .EXT_CACHE[VCA$W_EXTSIZE] GTRU 2
2 THEN
```

```
;  528          1515  3          BEGIN
;  529          1516  3          IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR2], DEV$V_CLU]
;  530          1517  3          THEN
;  531          1518  4              BEGIN
;  532          1519  4              BITMAP_FID = FID$C_BITMAP OR .CURRENT_VCB[VCB$W_RVN] ^ 24;
;  533          1520  4              IF CACHE_LOCK (.BITMAP_FID, EXT_CACHE[VCA$L_EXTCLKID], 0)
;  534          1521  4              THEN CACHE[VCA$V_EXTC_VALID] = T;
;  535          1522  4              END
;  536          1523  3          ELSE
;  537          1524  3              CACHE[VCA$V_EXTC_VALID] = 1;
;  538          1525  2          END;
;  539          1526  2
;  540          1527  1  END;                                       ! end of routine INIT_EXT_CACHE



                                            000C 00000          .ENTRY  INIT_EXT_CACHE, Save R2,R3                       ; 1458
                             52       04    AC   D0 00002          MOVL    CACHE, R2                                        ; 1510
                             53       04    A2   D0 00006          MOVL    4(R2), EXT_CACHE
                             51       94    AA   D0 0000A          MOVL    -108(BASE), R1                                   ; 1511
                3D      3A   A1             05   E0 0000E          BBS     #5, 58(R1), 2$
                             50       98    AA   D0 00013          MOVL    -104(BASE), R0                                   ; 1512
                34      0B   A0             01   E0 00017          BBS     #1, 11(R0), 2$
                             02             63   B1 0001C          CMPW    (EXT_CACHE), #2                                  ; 1513
                                            2F   1B 0001F          BLEQU   2$
                             27       3C    A1   E9 00021          BLBC    60(R1), 1$                                       ; 1516
                             50       98    AA   D0 00025          MOVL    -104(BASE), R0                                   ; 1519
                             50       0E    A0   3C 00029          MOVZWL  14(R0), R0
                50           50             18   78 0002D          ASHL    #24, R0, R0
                             50             02   88 00031          BISB2   #2, BITMAP_FID
                                            7E   D4 00034          CLRL    -(SP)                                            ; 1520
                                      0C    A3   9F 00036          PUSHAB  12(EXT_CACHE)
                                            50   DD 00039          PUSHL   BITMAP_FID
                      0000G  CF             03   FB 0003B          CALLS   #3, CACHE_LOCK
                             0D             50   E9 00040          BLBC    R0, 2$
                             50       04    AC   D0 00043          MOVL    CACHE, R0                                        ; 1521
                        0B   A0             02   88 00047          BISB2   #2, 11(R0)
                                                 04 0004B          RET                                                      ; 1516
                        0B   A2             02   88 0004C 1$:      BISB2   #2, 11(R2)                                       ; 1524
                                                 04 00050 2$:      RET                                                      ; 1527

; Routine Size:  81 bytes,    Routine Base:  $CODE$ + 0253
```

```
1 ROUTINE ALLOC_EXTENT (FIB, BLOCKS_NEEDED, START_LBN, BLOCKS_ALLOC) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine allocates a single contiguous area of disk from
1 !       the extent cache. Mode of allocation is determined by the
1 !       allocation control in the FIB.
1 !
1 ! CALLING SEQUENCE:
1 !       ALLOC_EXTENT (ARG1, ARG2, ARG3, ARG4)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of FIB for this operation
1 !       ARG2: number of blocks to allocate
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: ADDRESS OF VCB IN PROCESS
1 !       CURRENT_UCB: ADDRESS OF UCB IN PROCESS
1 !
1 ! OUTPUT PARAMETERS:
1 !       ARG3: address of longword to store starting LBN
1 !       ARG4: address of longword to store block count
1 !
1 ! IMPLICIT OUTPUTS:
1 !       LOC_LBN: placement LBN of allocation or 0
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1 if successful allocation
1 !       0 if failure
1 !
1 ! SIDE EFFECTS:
1 !       Extent cache modified
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       FIB             : REF BBLOCK;   ! FIB or operation
2
2 LABEL
2       CACHE_SEARCH;                   ! extent cache search procedure
2
2 REGISTER
2       EXTENT_LIST     : REF BBLOCKVECTOR [,8]; ! pointer to extent list
2
2 LOCAL
2       EXTENT_CACHE    : REF BBLOCK,   ! pointer to extent cache
2       BLOCK_COUNT,                    ! blocks needed rounded up to cluster
2       J,                              ! loop and extent list index
2       LBN,                            ! LBN of current extent
2       COUNT,                          ! block count of current extent
2       CYL_SIZE,                       ! size in blocks of volume's cylinder
2       CYL_BOUNDARY;                   ! LBN of next cylinder boundary
```

```
2
2 BIND_COMMON;
2
2 ! Search the extent cache. If placement is specified, check for a match
2 ! against the placement LBN.
2 !
2
3 CACHE_SEARCH: BEGIN
3
5 BLOCK_COUNT = ((.BLOCKS_NEEDED+.CURRENT_VCB[VCB$W_CLUSTER]-1)
3               / .CURRENT_VCB[VCB$W_CLUSTER]) * .CURRENT_VCB[VCB$W_CLUSTER];
3 EXTENT_CACHE = .BBLOCK [.CURRENT_VCB[VCB$L_CACHE], VCA$L_EXTCACHE];
3 EXTENT_LIST = EXTENT_CACHE[VCA$Q_EXTLIST];
3
3 J = 1;
3 WHILE .J LEQU  .EXTENT_CACHE[VCA$W_EXTCOUNT]
3 DO
4     BEGIN
4     LBN = .EXTENT_LIST[.J-1, VCA$L_EXTLBN];
4     COUNT = .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS];
4
4     IF .LOC_LBN EQL 0
5     OR (.LOC_LBN GEQU .LBN AND .LOC_LBN LSSU .LBN + .COUNT)
4     THEN
5         BEGIN
5
5 ! If placement is specified, adjust the base LBN and count accordingly.
5 ! Likewise, if on-cylinder allocation is requested, move the LBN to the
5 ! cylinder boundary. Then adjust to the cluster boundary.
5 !
5
5         IF .LOC_LBN NEQ 0 THEN LBN = .LOC_LBN / .CURRENT_VCB[VCB$W_CLUSTER]
5                                            * .CURRENT_VCB[VCB$W_CLUSTER];
5         IF .FIB[FIB$V_ONCYL]
5         THEN
6             BEGIN
6             CYL_SIZE = .CURRENT_UCB[UCB$B_SECTORS]
6                      * .CURRENT_UCB[UCB$B_TRACKS]
6                      / .CURRENT_VCB[VCB$B_BLOCKFACT];
6             CYL_BOUNDARY = (.LBN / .CYL_SIZE + 1) * .CYL_SIZE;
6             IF .CYL_BOUNDARY - .LBN LSSU .BLOCKS_NEEDED
6             THEN
7                 BEGIN
7                 IF NOT .FIB[FIB$V_EXACT]
9                 THEN LBN = ((.CYL_BOUNDARY + .CURRENT_VCB[VCB$W_CLUSTER] - 1)
7                            / .CURRENT_VCB[VCB$W_CLUSTER]) * .CURRENT_VCB[VCB$W_CLUSTER]
7                 ELSE RETURN 0;
6                 END;
5             END;
5
5         IF .LBN GEQU .EXTENT_LIST[.J-1, VCA$L_EXTLBN] + .COUNT
5         THEN COUNT = 0
5         ELSE COUNT = .COUNT + .EXTENT_LIST[.J-1, VCA$L_EXTLBN] - .LBN;
5
5 ! If the size is sufficient at this point, we win. If not, and the allocation
5 ! is neither exact nor on-cylinder, try backing off the adjustments made
5 ! above. Then check the size again; if the allocation is non-contiguous
```

```
5 ! or if the size is big enough, this is it.
5 !
5
5               IF .COUNT GEQU .BLOCK_COUNT
5               THEN LEAVE CACHE_SEARCH;
5
5               IF .LOC_LBN NEQ 0
5               AND NOT .FIB[FIB$V_ONCYL]
5               AND NOT .FIB[FIB$V_EXACT]
5               THEN
6                   BEGIN
6                   COUNT = MINU (.BLOCK_COUNT, .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS]);
6                   LBN = .EXTENT_LIST[.J-1, VCA$L_EXTLBN]
6                       + .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS]
6                       - .COUNT;
5                   END;
5               IF .COUNT GEQU .BLOCK_COUNT
6               OR  (.COUNT NEQ 0
6                    AND NOT .FIB[FIB$V_ALCON]
6                    AND NOT .FIB[FIB$V_ALCONB])
5               THEN LEAVE CACHE_SEARCH;
4               END;
4           J = .J + 1;
3           END;                                ! end of cache search loop
3
3   RETURN 0;                                   ! whole cache searched - nothing found
3
2   END;                                        ! end of block CACHE_SEARCH
2
2 ! We get here if we find a suitable cache entry. Deduct the count needed
2 ! from the count in the entry. If the result is zero, squish out the entry.
2 !
2 !
2
2 COUNT = MINU (.COUNT, .BLOCK_COUNT);
2 IF .COUNT EQL 0
2 THEN BUG_CHECK (MAPCNTZER, FATAL, 'Found zero extent in cache');
2
2 EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] = .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] - .COUNT;
2 IF .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] EQL 0
2 THEN
3     BEGIN
3     CH$MOVE ((.EXTENT_CACHE[VCA$W_EXTCOUNT]-.J)*8,
3              EXTENT_LIST[.J, VCA$L_EXTBLOCKS],
3              EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS]);
3     EXTENT_CACHE[VCA$W_EXTCOUNT] = .EXTENT_CACHE[VCA$W_EXTCOUNT] - 1;
3     END
3
3 ! Otherwise the allocation is only part of the extent. If it is from the
3 ! front of the extent, recompute the starting LBN of the extent.
3 !
3
2 ELSE IF .EXTENT_LIST[.J-1, VCA$L_EXTLBN] EQL .LBN
2 THEN
2     EXTENT_LIST[.J-1, VCA$L_EXTLBN] = .EXTENT_LIST[.J-1, VCA$L_EXTLBN] + .COUNT
2
2 ! If the allocation is from the end of the extent, no further action is necessary.
```

```
;  713        1699  2 ! If it is from the middle, we must split the extent. To do so, shuffle the
;  714        1700  2 ! remainder of the extent list up by one, bump the entry count, and compute
;  715        1701  2 ! the split entries.
;  716        1702  2 !
;  717        1703  2
;  718        1704  2 ELSE IF .EXTENT_LIST[.J-1, VCA$L_EXTLBN] + .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] NEQ .LBN
;  719        1705  2 THEN
;  720        1706  3     BEGIN
;  721        1707  3     CH$MOVE ((.EXTENT_CACHE[VCA$W_EXTCOUNT]-.J)*8,
;  722        1708  3              EXTENT_LIST[.J, VCA$L_EXTBLOCKS],
;  723        1709  3              EXTENT_LIST[.J+1, VCA$L_EXTBLOCKS]);
;  724        1710  3     EXTENT_CACHE[VCA$W_EXTCOUNT] = .EXTENT_CACHE[VCA$W_EXTCOUNT] + 1;
;  725        1711  3     EXTENT_LIST[.J, VCA$L_EXTLBN] = .COUNT + .LBN;
;  726        1712  3     EXTENT_LIST[.J, VCA$L_EXTBLOCKS] = .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS]
;  727        1713  3                                      + .EXTENT_LIST[.J-1, VCA$L_EXTLBN]
;  728        1714  3                                      - .LBN;
;  729        1715  3     EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] = .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS]
;  730        1716  3                                        - .EXTENT_LIST[.J, VCA$L_EXTBLOCKS];
;  731        1717  2     END;
;  732        1718  2
;  733        1719  2 .START_LBN = .LBN;
;  734        1720  2 .BLOCKS_ALLOC = .COUNT;
;  735        1721  2 EXTENT_CACHE[VCA$L_EXTTOTAL] = .EXTENT_CACHE[VCA$L_EXTTOTAL] - .COUNT;
;  736        1722  2
;  737        1723  2 RETURN 1;
;  738        1724  2
;  739        1725  1 END;                                     ! end of routine ALLOC_EXTENT


                                                                      .EXTRN  BUG$_MAPCNTZER

                                                  0BFC 00000 ALLOC_EXTENT:
                                                                      .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R11                    ; 1528
                                  5E         0C   C2 00002            SUBL2   #12, SP                                             ;
                          08      AE    98   AA   9E 00005            MOVAB   -104(BASE), 8(SP)                                   ; 1584
                                  50    08   BE   D0 0000A            MOVL    @8(SP), R0                                          ; 1594
                                  51    3C   A0   3C 0000E            MOVZWL  60(R0), R1                                          ;
                                  51    08   AC   C0 00012            ADDL2   BLOCKS_NEEDED, R1                                   ;
                                             51   D7 00016            DECL    R1                                                  ;
                                  52    3C   A0   3C 00018            MOVZWL  60(R0), R2                                          ; 1595
                                  51         52   C6 0001C            DIVL2   R2, R1                                              ;
                                  53    3C   A0   3C 0001F            MOVZWL  60(R0), R3                                          ;
                                  51         53   C4 00023            MULL2   R3, BLOCK_COUNT                                     ;
                                  50    08   BE   D0 00026            MOVL    @8(SP), R0                                          ; 1596
                                  50    58   A0   D0 0002A            MOVL    88(R0), R0                                          ;
                                  57    04   A0   D0 0002E            MOVL    4(R0), EXTENT_CACHE                                 ;
                                  56    2C   A7   9E 00032            MOVAB   44(R7), EXTENT_LIST                                 ; 1597
                                  5B         01   D0 00036            MOVL    #1, J                                               ; 1599
         5B        02    A7       10         00   ED 00039 1$:        CMPZV   #0, #16, 2(EXTENT_CACHE), J                         ; 1600
                                             03   1E 0003F            BGEQU   3$                                                  ;
                                           018D   31 00041 2$:        BRW     20$                                                 ;
                                  54       664B   7E 00044 3$:        MOVAQ   (EXTENT_LIST)[J], R4                                ; 1603
                                  59    FC   A4   D0 00048            MOVL    -4(R4), LBN                                         ;
                                             54   D0 0004C            MOVL    R4, R2                                              ; 1604
                                  58    F8   A2   D0 0004F            MOVL    -8(R2), COUNT                                       ;
                                  53    20   AA   D0 00053            MOVL    32(BASE), R3                                        ; 1606
```

;

```
                                  11 13 00057        BEQL     6$                                 ;
                    59            53 D1 00059        CMPL     R3, LBN                            ; 1607
                                  03 1E 0005C        BGEQU    5$                                 ;
                                00D0 31 0005E 4$:    BRW      13$                                ;
      52            59            58 C1 00061 5$:    ADDL3    COUNT, LBN, R2                     ;
                    52            53 D1 00065        CMPL     R3, R2                             ;
                                  F4 1E 00068        BGEQU    4$                                 ;
                                  53 D5 0006A 6$:    TSTL     R3                                 ; 1616
                                  12 13 0006C        BEQL     7$                                 ;
                    52      08    BE D0 0006E        MOVL     @8(SP), R2                         ;
                    55      3C    A2 3C 00072        MOVZWL   60(R2), R5                         ;
                    53            55 C6 00076        DIVL2    R5, R3                             ;
                    59      3C    A2 3C 00079        MOVZWL   60(R2), LBN                        ; 1617
                    59            53 C4 0007D        MULL2    R3, LBN                            ;
                    53      04    AC D0 00080 7$:    MOVL     FIB, R3                            ; 1618
      4C      20    A3            01 E1 00084        BBC      #1, 32(R3), 8$                     ;
                    52      94    AA D0 00089        MOVL     -108(BASE), R2                     ; 1621
                    55      44    A2 9A 0008D        MOVZBL   68(R2), R5                         ; 1622
                    52      45    A2 9A 00091        MOVZBL   69(R2), R2                         ;
                    52            55 C4 00095        MULL2    R5, R2                             ;
                    55      08    BE D0 00098        MOVL     @8(SP), R5                         ; 1623
                    6E      52    A5 9A 0009C        MOVZBL   82(R5), (SP)                       ;
04    AE            52            6E C7 000A0        DIVL3    (SP), R2, CYL_SIZE                 ;
      52            59      04    AE C7 000A5        DIVL3    CYL_SIZE, LBN, R2                  ; 1624
                                  52 D6 000AA        INCL     R2                                 ;
      50            52      04    AE C5 000AC        MULL3    CYL_SIZE, R2, CYL_BOUNDARY         ;
      52            50            59 C3 000B1        SUBL3    LBN, CYL_BOUNDARY, R2              ; 1625
              08    AC            52 D1 000B5        CMPL     R2, BLOCKS_NEEDED                  ;
                                  1A 1E 000B9        BGEQU    8$                                 ;
                    82      20    A3 E8 000BB        BLBS     32(R3), 2$                         ; 1628
                    52      08    BE D0 000BF        MOVL     @8(SP), R2                         ; 1629
                    52            3C C0 000C3        ADDL2    #60, R2                            ;
                    52            62 3C 000C6        MOVZWL   (R2), R2                           ;
                    55      FF  A240 9E 000C9        MOVAB    -1(R2)[CYL_BOUNDARY], R5           ;
                    55            52 C6 000CE        DIVL2    R2, R5                             ; 1630
      59            55            52 C5 000D1        MULL3    R2, R5, LBN                        ;
      52            58      FC    A4 C1 000D5 8$:    ADDL3    -4(R4), COUNT, R2                  ; 1635
                    52            59 D1 000DA        CMPL     LBN, R2                            ;
                                  04 1F 000DD        BLSSU    9$                                 ;
                                  58 D4 000DF        CLRL     COUNT                              ; 1636
                                  09 11 000E1        BRB      10$                                ;
      52            58      FC    A4 C1 000E3 9$:    ADDL3    -4(R4), COUNT, R2                  ; 1637
      58            52            59 C3 000E8        SUBL3    LBN, R2, COUNT                     ;
                    51            58 D1 000EC 10$:   CMPL     COUNT, BLOCK_COUNT                 ; 1645
                                  45 1E 000EF        BGEQU    14$                                ;
                            20    AA D5 000F1        TSTL     32(BASE)                           ; 1648
                                  29 13 000F4        BEQL     12$                                ;
      24      20    A3            01 E0 000F6        BBS      #1, 32(R3), 12$                    ; 1649
                    20      20    A3 E8 000FB        BLBS     32(R3), 12$                        ; 1650
                    52            54 D0 000FF        MOVL     R4, R2                             ; 1653
                    55            51 D0 00102        MOVL     BLOCK_COUNT, R5                    ;
              F8    A2            55 D1 00105        CMPL     R5, -8(R2)                         ;
                                  04 1B 00109        BLEQU    11$                                ;
                    55      F8    A2 D0 0010B        MOVL     -8(R2), R5                         ;
                    58            55 D0 0010F 11$:   MOVL     R5, COUNT                          ;
                    52            54 D0 00112        MOVL     R4, R2                             ; 1655
      54      FC    A4      F8    A2 C1 00115        ADDL3    -8(R2), -4(R4), R4                 ;
```

```
59            54             58 C3 0011B        SUBL3   COUNT, R4, LBN                  ; 1656
              51             58 D1 0011F 12$:   CMPL    COUNT, BLOCK_COUNT              ; 1658
                             12 1E 00122        BGEQU   14$                             ;
                             58 D5 00124        TSTL    COUNT                           ; 1659
                             09 13 00126        BEQL    13$                             ;
              05       16    A3 E8 00128        BLBS    22(R3), 13$                     ; 1660
05      16    A3             01 E1 0012C        BBC     #1, 22(R3), 14$                 ; 1661
                             5B D6 00131 13$:   INCL    J                               ; 1664
                           FF03 31 00133        BRW     1*                              ; 1600
              50             58 D0 00136 14$:   MOVL    COUNT, R0                       ; 1676
              51             50 D1 00139        CMPL    R0, BLOCK_COUNT                 ;
                             03 1B 0013C        BLEQU   15$                             ;
              50             51 D0 0013E        MOVL    BLOCK_COUNT, R0                 ;
              58             50 D0 00141 15$:   MOVL    R0, COUNT                       ;
                             04 12 00144        BNEQ    16$                             ; 1677
                              FEFF 00146        BUGW                                    ; 1678
                             0000* 00148        .WORD   <BUG$_MAPCNTZER!4>              ;
                       F8 A64B 7F 0014A 16$:    PUSHAQ  -8(EXTENT_LIST)[J]              ; 1680
              9E             58 C2 0014E        SUBL2   COUNT, @(SP)+                   ;
              50       F8 A64B 7E 00151         MOVAQ   -8(EXTENT_LIST)[J], R0          ; 1681
                             60 D5 00156        TSTL    (R0)                            ;
                             16 12 00158        BNEQ    17$                             ;
              51       02    A7 3C 0015A        MOVZWL  2(EXTENT_CACHE), R1             ; 1684
              51             5B C2 0015E        SUBL2   J, R1                           ;
              51             08 C4 00161        MULL2   #8, R1                          ;
                           664B 7F 00164        PUSHAQ  (EXTENT_LIST)[J]                ; 1686
60            9E             51 28 00167        MOVC3   R1, @(SP)+, (R0)                ;
                       02    A7 B7 0016B        DECW    2(EXTENT_CACHE)                 ; 1687
                             51 11 0016E        BRB     19$                             ; 1681
              59       04    A0 D1 00170 17$:   CMPL    4(R0), LBN                      ; 1694
                             06 12 00174        BNEQ    18$                             ;
        04    A0             58 C0 00176        ADDL2   COUNT, 4(R0)                    ; 1696
                             45 11 0017A        BRB     19$                             ;
50      04    A0             60 C1 0017C 18$:   ADDL3   (R0), 4(R0), R0                 ; 1704
              59             50 D1 00181        CMPL    R0, LBN                         ;
                             3B 13 00184        BEQL    19$                             ;
              50       02    A7 3C 00186        MOVZWL  2(EXTENT_CACHE), R0             ; 1707
              50             5B C2 0018A        SUBL2   J, R0                           ;
              50             08 C4 0018D        MULL2   #8, R0                          ;
                       08 A64B 7F 00190         PUSHAQ  8(EXTENT_LIST)[J]               ; 1709
                           664B 7F 00194        PUSHAQ  (EXTENT_LIST)[J]                ;
9E            9E             50 28 00197        MOVC3   R0, @(SP)+, @(SP)+              ;
                       02    A7 B6 0019B        INCW    2(EXTENT_CACHE)                 ; 1710
                       04 A64B 7F 0019E         PUSHAQ  4(EXTENT_LIST)[J]               ; 1711
9E            58             59 C1 001A2        ADDL3   LBN, COUNT, @(SP)+              ;
              50       F8 A64B 7E 001A6         MOVAQ   -8(EXTENT_LIST)[J], R0          ; 1712
50            60       04    A0 C1 001AB        ADDL3   4(R0), (R0), R0                 ; 1713
                           664B 7F 001B0        PUSHAQ  (EXTENT_LIST)[J]                ; 1714
9E            50             59 C3 001B3        SUBL3   LBN, R0, @(SP)+                 ;
                       F8 A64B 7F 001B7         PUSHAQ  -8(EXTENT_LIST)[J]              ; 1716
                           664B 7F 001BB        PUSHAQ  (EXTENT_LIST)[J]                ;
              9E             9E C2 001BE        SUBL2   @(SP)+, @(SP)+                  ;
        0C    BC             59 D0 001C1 19$:   MOVL    LBN, @START_LBN                 ; 1719
        10    BC             58 D0 001C5        MOVL    COUNT, @BLOCKS_ALLOC            ; 1720
        04    A7             58 C2 001C9        SUBL2   COUNT, 4(EXTENT_CACHE)          ; 1721
              50             01 D0 001CD        MOVL    #1, R0                          ; 1723
                                04 001D0        RET                                     ;
```

```
                                               50  D4 001D1 20$:     CLRL     R0                                        ; 1725
                                                   04 001D3          RET                                                ;

; Routine Size:  468 bytes,    Routine Base:  $CODE$ + 02A4
```

```
1 ROUTINE RETURN_EXTENT (START_LBN, BLOCK_COUNT) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine returns the indicated extent to the extent cache.
1 !       It searches the cache to insert the entry in LBN order, and merges
1 !       it with any adjacent entries. If the extent overlaps existing
1 !       entries, an error return is made.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       RETURN_EXTENT (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: starting LBN of extent
1 !       ARG2: block count
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1 if successful
1 !       0 if blocks overlap
1 !
1 ! SIDE EFFECTS:
1 !       extent cache modified
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       EXTENT_CACHE    : REF BBLOCK,   ! pointer to extent cache
2       EXTENT_LIST     : REF BBLOCKVECTOR [,8], ! pointer to extent list
2       J;                              ! extent list index
2
2 BIND_COMMON;
2
2 ! Search the extent cache until we find an entry whose start LBN is
2 ! higher than the end LBN of the extent being returned.
2 !
2
2 IF .BLOCK_COUNT LEQ 0
2 THEN BUG_CHECK (MAPCNTZER, FATAL, 'Attempted to return zero extent to cache');
2
2 EXTENT_CACHE = .BBLOCK [.CURRENT_VCB[VCB$L_CACHE], VCA$L_EXTCACHE];
2 EXTENT_LIST = EXTENT_CACHE[VCA$Q_EXTLIST];
2 J = 1;
2 UNTIL .J GTRU .EXTENT_CACHE[VCA$W_EXTCOUNT]
```

```
2 DO
3     BEGIN
3     IF .EXTENT_LIST[.J-1, VCA$L_EXTLBN] GEQU .START_LBN + .BLOCK_COUNT
3     THEN EXITLOOP;
3     J = .J + 1;
2     END;
2
2 ! If there is a preceding entry, check it for overlap.
2 !
2
2 IF .J GTRU 1
2 THEN
3     BEGIN
3     IF .EXTENT_LIST[.J-2, VCA$L_EXTLBN] + .EXTENT_LIST[.J-2, VCA$L_EXTBLOCKS]
3     GTRU .START_LBN
3     THEN RETURN 0;
2     END;
2
2 ! Check for adjacency with the preceding and current extents; if so, do
2 ! a merge.
2 !
2
2 IF .J GTRU 1
2 AND .EXTENT_LIST[.J-2, VCA$L_EXTLBN] + .EXTENT_LIST[.J-2, VCA$L_EXTBLOCKS]
2     EQL .START_LBN
2 THEN
3     BEGIN
3     EXTENT_LIST[.J-2, VCA$L_EXTBLOCKS] = .EXTENT_LIST[.J-2, VCA$L_EXTBLOCKS] + .BLOCK_COUNT;
3
3     IF .J LEQU .EXTENT_CACHE[VCA$W_EXTCOUNT]
3     AND .EXTENT_LIST[.J-1, VCA$L_EXTLBN] EQL .START_LBN + .BLOCK_COUNT
3     THEN
4         BEGIN
4         EXTENT_LIST[.J-2, VCA$L_EXTBLOCKS] =
4             .EXTENT_LIST[.J-2, VCA$L_EXTBLOCKS]
4           + .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS];
4         CH$MOVE ((.EXTENT_CACHE[VCA$W_EXTCOUNT]-.J)*8,
4                  EXTENT_LIST[.J, VCA$L_EXTBLOCKS],
4                  EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS]);
4         EXTENT_CACHE[VCA$W_EXTCOUNT] = .EXTENT_CACHE[VCA$W_EXTCOUNT] - 1;
3         END;
3     END
3
2 ELSE IF .J LEQU .EXTENT_CACHE[VCA$W_EXTCOUNT]
2      AND .EXTENT_LIST[.J-1, VCA$L_EXTLBN] EQL .START_LBN + .BLOCK_COUNT
2 THEN
3     BEGIN
3     EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] = .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] + .BLOCK_COUNT;
3     EXTENT_LIST[.J-1, VCA$L_EXTLBN] = .START_LBN;
3     END
3
2 ELSE
3     BEGIN
3     CH$MOVE ((.EXTENT_CACHE[VCA$W_EXTCOUNT]-.J+1)*8,
3              EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS],
3              EXTENT_LIST[.J, VCA$L_EXTBLOCKS]);
3     EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] = .BLOCK_COUNT;
```

```
;  855        1840 3      EXTENT_LIST[.J-1, VCA$L_EXTLBN] = .START_LBN;
;  856        1841 3      EXTENT_CACHE[VCA$W_EXTCOUNT] = .EXTENT_CACHE[VCA$W_EXTCOUNT] + 1;
;  857        1842 2      END;
;  858        1843 2
;  859        1844 2  EXTENT_CACHE[VCA$L_EXTTOTAL] = .EXTENT_CACHE[VCA$L_EXTTOTAL] + .BLOCK_COUNT;
;  860        1845 2
;  861        1846 2  RETURN 1;
;  862        1847 2
;  863        1848 1  END;                                   ! end of routine RETURN_EXTENT


                                          01FC 00000 RETURN_EXTENT:
                                                          .WORD    Save R2,R3,R4,R5,R6,R7,R8                  ; 1726
                                    08    AC  D5 00002    TSTL     BLOCK_COUNT                                ; 1776
                                          04  14 00005    BGTR     1$
                                          FEFF 00007      BUGW                                                ; 1777
                                          0000* 00009     .WORD    <BUG$_MAPCNTZER!4>
                          50        98    AA  D0 0000B 1$: MOVL    -104(BASE), R0                             ; 1779
                          50        58    A0  D0 0000F    MOVL     88(R0), R0
                          57        04    A0  D0 00013    MOVL     4(R0), EXTENT_CACHE
                          56        2C    A7  9E 00017    MOVAB    44(R7), EXTENT_LIST                        ; 1780
                          58              01  D0 0001B    MOVL     #1, J                                      ; 1781
                53     04 AC        08    AC  C1 0001E    ADDL3    BLOCK_COUNT, START_LBN, R3                 ; 1785
      58     02 A7        10              00  ED 00024 2$: CMPZV   #0, #16, 2(EXTENT_CACHE), J                ; 1782
                                          0D  1F 0002A    BLSSU    3$
                                    FC  A648  7F 0002C    PUSHAQ   -4(EXTENT_LIST)[J]                         ; 1785
                          53              9E  D1 00030    CMPL     @(SP)+, R3
                                          04  1E 00033    BGEQU    3$
                                          58  D6 00035    INCL     J                                          ; 1787
                                          EB  11 00037    BRB      2$                                         ; 1782
                                          52  D4 00039 3$: CLRL    R2                                         ; 1793
                          01              58  D1 0003B    CMPL     J, #1
                                          18  1B 0003E    BLEQU    4$
                                          52  D6 00040    INCL     R2
                          51            6648  7E 00042    MOVAQ    (EXTENT_LIST)[J], R1                       ; 1796
                          50              51  D0 00046    MOVL     R1, R0
                51     F4 A1        F0    A0  C1 00049    ADDL3    -16(R0), -12(R1), R1
                       04 AC              51  D1 0004F    CMPL     R1, START_LBN                              ; 1797
                                          03  1B 00053    BLEQU    4$
                                        00AE  31 00055    BRW      8$
                          57              52  E9 00058 4$: BLBC    R2, 5$                                     ; 1805
                          51            6648  7E 0005B    MOVAQ    (EXTENT_LIST)[J], R1                       ; 1806
                          50              51  D0 0005F    MOVL     R1, R0
                51     F4 A1        F0    A0  C1 00062    ADDL3    -16(R0), -12(R1), R1
                       04 AC              51  D1 00068    CMPL     R1, START_LBN                              ; 1807
                                          44  12 0006C    BNEQ     5$
                                    F0  A648  7F 0006E    PUSHAQ   -16(EXTENT_LIST)[J]                        ; 1810
                          9E        08    AC  C0 00072    ADDL2    BLOCK_COUNT, @(SP)+
      58     02 A7        10              00  ED 00076    CMPZV    #0, #16, 2(EXTENT_CACHE), J                ; 1812
                                          7F  1F 0007C    BLSSU    7$
                50     04 AC        08    AC  C1 0007E    ADDL3    BLOCK_COUNT, START_LBN, R0                 ; 1813
                                    FC  A648  7F 00084    PUSHAQ   -4(EXTENT_LIST)[J]
                          50              9E  D1 00088    CMPL     @(SP)+, R0
                                          70  12 0008B    BNEQ     7$
```

```
                                         F0 A648  7F 0008D          PUSHAQ  -16(EXTENT_LIST)[J]                                  ; 1818
                                         F8 A648  7F 00091          PUSHAQ  -8(EXTENT_LIST)[J]                                   ;
                                9E           9E  C0 00095           ADDL2   @(SP)+, @(SP)+                                       ;
                                50       02  A7  3C 00098           MOVZWL  2(EXTENT_CACHE), R0                                  ; 1819
                                50           58  C2 0009C           SUBL2   J, R0                                                ;
                                50           08  C4 0009F           MULL2   #8, R0                                               ;
                                         F8 A648  7F 000A2          PUSHAQ  -8(EXTENT_LIST)[J]                                   ; 1821
                                            6648  7F 000A6          PUSHAQ  (EXTENT_LIST)[J]                                     ;
                   9E           9F           50  28 000A9           MOVC3   R0, @(SP)+, @(SP)+                                   ;
                                         02  A7  B7 000AD           DECW    2(EXTENT_CACHE)                                      ; 1822
                                             4B  11 000B0           BRB     7$                                                   ; 1805
                                51       F8 A648  7E 000B2 5$:      MOVAQ   -8(EXTENT_LIST)[J], R1                               ; 1830
   58        02    A7           10           00  ED 000B7           CMPZV   #0, #16, 2(EXTENT_CACHE), J                          ; 1826
                                             17  1F 000BD           BLSSU   6$                                                   ;
                                         FC A648  7F 000BF          PUSHAQ  -4(EXTENT_LIST)[J]                                   ; 1827
                                53           9E  D1 000C3           CMPL    @(SP)+, R3                                           ;
                                             0E  12 000C6           BNEQ    6$                                                   ;
                                61       08  AC  C0 000C8           ADDL2   BLOCK_COUNT, (R1)                                    ; 1830
                                         FC A648  7F 000CC          PUSHAQ  -4(EXTENT_LIST)[J]                                   ; 1831
                                9E       04  AC  D0 000D0           MOVL    START_LBN, @(SP)+                                    ;
                                             27  11 000D4           BRB     7$                                                   ; 1826
                                50       02  A7  3C 000D6 6$:       MOVZWL  2(EXTENT_CACHE), R0                                  ; 1836
                                50           58  C2 000DA           SUBL2   J, R0                                                ;
                                50           08  C4 000DD           MULL2   #8, R0                                               ;
                                50           08  C0 000E0           ADDL2   #8, R0                                               ;
                                            6648  7F 000E3          PUSHAQ  (EXTENT_LIST)[J]                                     ; 1838
                   9E           61           50  28 000E6           MOVC3   R0, (R1), @(SP)+                                     ;
                                         F8 A648  7F 000EA          PUSHAQ  -8(EXTENT_LIST)[J]                                   ; 1839
                                9E       08  AC  D0 000EE           MOVL    BLOCK_COUNT, @(SP)+                                  ;
                                         FC A648  7F 000F2          PUSHAQ  -4(EXTENT_LIST)[J]                                   ; 1840
                                9E       04  AC  D0 000F6           MOVL    START_LBN, @(SP)+                                    ;
                                         02  A7  B6 000FA           INCW    2(EXTENT_CACHE)                                      ; 1841
                          04    A7       08  AC  C0 000FD 7$:       ADDL2   BLOCK_COUNT, 4(EXTENT_CACHE)                         ; 1844
                                50           01  D0 00102           MOVL    #1, R0                                               ; 1846
                                                 04 00105           RET                                                          ;
                                             50  D4 00106 8$:       CLRL    R0                                                   ; 1848
                                                 04 00108           RET                                                          ;

; Routine Size:  265 bytes,    Routine Base:  $CODE$ + 0478
```

```
1 GLOBAL ROUTINE PURGE_EXTENT (ENTRY_COUNT, CACHE_LIMIT) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine removes the specified number of entries from the
1 !       extent cache and returns the blocks to the storage bitmap.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       PURGE_EXTENT (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: number of entries to retain
1 !       ARG2: total number of blocks to retain in cache
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       extent cache and storage bitmap modified
1 !
1 !--
1
2 BEGIN
2
2 BUILTIN FP;
2
2 LOCAL
2       EXTENT_CACHE    : REF BBLOCK,   ! pointer to extent cache
2       EXTENT_LIST     : REF BBLOCKVECTOR [,8], ! pointer to extent list
2       BLOCK,                          ! bitmap block number of current extent
2       VBN,                            ! bitmap block number of best group
2       COUNT,                          ! count of entries in current group
2       BLOCKS,                         ! block count in current group
2       BASE_J,                         ! cache index of start of current map block
2       BEST_COUNT,                     ! count of entries in best group
2       BEST_BLOCKS,                    ! count of blocks in best group
2       BEST_J,                         ! index of start of best group
2       MOST_BLOCKS,                    ! count of blocks in largest group
2       MOST_J,                         ! starting index on largest group
2       BLOCKS_TO_REM,                  ! number of blocks to remove from cache
2       LBN,                            ! starting LBN of extent
2       BLOCK_COUNT,                    ! count of extent
2       LOCK_STATUS     : VECTOR [2];   ! lock status block
2
2 BIND_COMMON;
```

```
2
2 EXTERNAL ROUTINE
2         ALLOCATION_LOCK : L_NORM,
2         ZERO_ON_ERROR;                    ! return zero on error signal (handler)
2
2 ! Serialize processing against other storage/header allocation/deallocation.
2 !
2
2 ALLOCATION_LOCK ();
2
2 ! If we are not removing all the entries, scan the extent cache for the
2 ! desired number of entries that reside in the same bitmap block.
2 !
2
2 EXTENT_CACHE = .BBLOCK [.CURRENT_VCB[VCB$L_CACHE], VCA$L_EXTCACHE];
2 EXTENT_LIST = EXTENT_CACHE[VCA$Q_EXTLIST];
2
2 IF .ENTRY_COUNT NEQ 0
2 THEN
3     BEGIN
3     BEST_COUNT = 0;
3     BEST_BLOCKS = 0;
3     MOST_BLOCKS = 0;
3     VBN = -1;
3
3     INCR J FROM 1 TO .EXTENT_CACHE[VCA$W_EXTCOUNT]
3     DO
4         BEGIN
5         BLOCK = (.EXTENT_LIST[.J-1, VCA$L_EXTLBN] / 4096)
4                 / .CURRENT_VCB[VCB$W_CLUSTER];
4         IF .BLOCK NEQ .VBN
4         THEN
5             BEGIN
5             VBN = .BLOCK;
5             COUNT = 0;
5             BLOCKS = 0;
5             BASE_J = .J;
4             END;
4         COUNT = .COUNT + 1;
4         BLOCKS = .BLOCKS + .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS];
4
4         IF .COUNT GTRU .BEST_COUNT
4         THEN
5             BEGIN
5             BEST_COUNT = .COUNT;
5             BEST_BLOCKS = .BLOCKS;
5             BEST_J = .BASE_J;
4             END;
4
4         IF .BLOCKS GTRU .MOST_BLOCKS
4         THEN
5             BEGIN
5             MOST_BLOCKS = .BLOCKS;
5             MOST_J = .BASE_J;
4             END;
3         END;
3
```

```
3 ! See what we got from scanning the cache. If removing the greatest number
3 ! of entries will satisfy the space reduction, then do that. Otherwise,
3 ! go for the set of entries with the most space. If that isn't sufficient,
3 ! start at the beginning of the cache.
3 !
3
3     BLOCKS_TO_REM = .EXTENT_CACHE[VCA$L_EXTTOTAL] - .CACHE_LIMIT;
3     IF .CACHE_LIMIT GTRU .EXTENT_CACHE[VCA$L_EXTTOTAL]
3     THEN BLOCKS_TO_REM = 0;
3
3     IF .BEST_BLOCKS LSSU .BLOCKS_TO_REM
3     THEN
4         BEGIN
4         BEST_J = .MOST_J;
4         IF .MOST_BLOCKS LSSU .BLOCKS_TO_REM
4         THEN BEST_J = 1;
3         END;
3
3     VBN = (.EXTENT_LIST[.BEST_J-1, VCA$L_EXTLBN] / 4096) / .CURRENT_VCB[VCB$W_CLUSTER];
3
3 ! Now scan the extent cache, remove the called for entries, and return
3 ! the blocks to the storage bitmap.
3 !
3
3     UNTIL .BEST_J GTRU .EXTENT_CACHE[VCA$W_EXTCOUNT]
3     DO
4         BEGIN
4         LBN = .EXTENT_LIST[.BEST_J-1, VCA$L_EXTLBN];
4         IF .EXTENT_CACHE[VCA$L_EXTTOTAL] LEQU .CACHE_LIMIT
5         AND (.EXTENT_CACHE[VCA$W_EXTCOUNT] LEQU .ENTRY_COUNT
6              OR (.VBN NEQ (.LBN / 4096) / .CURRENT_VCB[VCB$W_CLUSTER]
6                  AND .ENTRY_COUNT NEQ 0)
5             )
4         THEN EXITLOOP;
4
4         BLOCK_COUNT = .EXTENT_LIST[.BEST_J-1, VCA$L_EXTBLOCKS];
4         IF .EXTENT_CACHE[VCA$L_EXTTOTAL] - .BLOCK_COUNT LSSU .CACHE_LIMIT
4         AND .EXTENT_CACHE[VCA$W_EXTCOUNT] LEQU .ENTRY_COUNT
4         THEN
5             BEGIN
5             BLOCK_COUNT = .EXTENT_CACHE[VCA$L_EXTTOTAL] - .CACHE_LIMIT;
7             BLOCK_COUNT = ((.BLOCK_COUNT + .CURRENT_VCB[VCB$W_CLUSTER]-1)
5                           / .CURRENT_VCB[VCB$W_CLUSTER]) * .CURRENT_VCB[VCB$W_CLUSTER];
4             END;
4         REMOVE_EXTENT (.LBN, .BLOCK_COUNT);
4         RETURN_BITMAP (.LBN, .BLOCK_COUNT);
3         END;
3     END
3
3 ! For a full purge of the extent cache, just sweep through it, releasing
3 ! the entries. This is done under a handler so that I/O errors do not
3 ! terminate the operation. At the end, we release the cache lock.
3 !
3
2 ELSE
3     BEGIN
3     .FP = ZERO_ON_ERROR;
```

```
                                                      C  6
SMALOC                                                16-Sep-1984 01:11:44    VAX-11 Bliss-32 V4.0-742                 Page  30
V04-000                                               14-Sep-1984 12:30.47    DISK$VMSMASTER:[F11X.SRC]SMALOC.B32;1      (7)

; 1036           2020  3       UNTIL .EXTENT_CACHE[VCA$W_EXTCOUNT] EQL 0
; 1037           2021  3       DO
; 1038           2022  4           BEGIN
; 1039           2023  4           LBN = .EXTENT_LIST[0, VCA$L_EXTLBN];
; 1040           2024  4           BLOCK_COUNT = .EXTENT_LIST[0, VCA$L_EXTBLOCKS];
; 1041           2025  4           REMOVE_EXTENT (.LBN, .BLOCK_COUNT);
; 1042           2026  4           RETURN_BITMAP (.LBN, .BLOCK_COUNT);
; 1043           2027  3           END;
; 1044           2028  3
; 1045           2029  3       IF .EXTENT_CACHE[VCA$L_EXTCLKID] NEQ 0
; 1046           2030  3       THEN
; 1047           2031  4           BEGIN
; 1048           2032  4           LOCK_STATUS[1] = .EXTENT_CACHE[VCA$L_EXTCLKID];
; 1049         P 2033  4           IF NOT $ENQW (EFN     = EFN,
; 1050         P 2034  4                         LKMODE = LCK$K_NLMODE,
; 1051         P 2035  4                         FLAGS  = LCK$M_NOQUEUE OR LCK$M_SYNCSTS OR LCK$M_CONVERT OR LCK$M_CVTSYS,
; 1052         P 2036  4                         LKSB   = LOCK_STATUS
; 1053           2037  5                         )
; 1054           2038  4           THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error');
; 1055           2039  3           END;
; 1056           2040  3       BBLOCK [.CURRENT_VCB[VCB$L_CACHE], VCA$V_EXTC_VALID] = 0;
; 1057           2041  2       END;
; 1058           2042  2
; 1059           2043  1   END;                                   ! end of routine PURGE_EXTENT


                                                                      .EXTRN  ZERO_ON_ERROR, SYS$ENQW
                                                                      .EXTRN  BUG$_XQPERR

                                                    0BFC 00000        .ENTRY  PURGE_EXTENT, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 1849
                                                                              R11                                          ;
                         5E                 28   C2 00002             SUBL2   #40, SP                                      ;
                   1C    AE         98      AA   9E 00005             MOVAB   -104(BASE), 28(SP)                           ; 1903
                0000G    CF                 00   FB 0000A             CALLS   #0, ALLOCATION_LOCK                          ; 1914
                         50         1C      BE   D0 0000F             MOVL    @28(SP), R0                                  ; 1920
                         50         58      A0   D0 00013             MOVL    88(R0), R0                                   ;
                         52         04      A0   D0 00017             MOVL    4(R0), EXTENT_CACHE                          ;
                         53         2C      A2   9E 0001B             MOVAB   44(R2), EXTENT_LIST                          ; 1921
                                    04      AC   D5 0001F             TSTL    ENTRY_COUNT                                  ; 1923
                                            03   12 00022             BNEQ    1$                                           ;
                                          012C   31 00024             BRW     13$                                          ;
                                            5B   D4 00027 1$:         CLRL    BEST_COUNT                                   ; 1926
                                    18      AE   D4 00029             CLRL    BEST_BLOCKS                                  ; 1927
                                            59   D4 0002C             CLRL    MOST_BLOCKS                                  ; 1928
                   14    AE                 01   CE 0002E             MNEGL   #1, VBN                                      ; 1929
                   10    AE         02      A2   3C 00032             MOVZWL  2(EXTENT_CACHE), 16(SP)                      ; 1931
                                            51   D4 00037             CLRL    J                                            ; 1934
                                            4F   11 00039             BRB     5$                                           ;
                         50               6341   7E 0003B 2$:         MOVAQ   (EXTENT_LIST)[J], R0                         ;
         58        FC    A0 00001000        8F   C7 0003F             DIVL3   #4096, -4(R0), R8                            ;
                         54         1C      BE   D0 00048             MOVL    @28(SP), R4                                  ; 1935
                         6E         3C      A4   3C 0004C             MOVZWL  60(R4), (SP)                                 ;
     08       AE         58                 6E   C7 00050             DIVL3   (SP), R8, BLOCK                              ;
                   14    AE         08      AE   D1 00055             CMPL    BLOCK, VBN                                   ; 1936
                                            0B   13 0005A             BEQL    3$                                           ;
                   14    AE         08      AE   D0 0005C             MOVL    BLOCK, VBN                                   ; 1939
```

```
                                     56  7C 00061        CLRQ    BLOCKS                                       ; 1941
                    0C  AE           51  D0 00063        MOVL    J, BASE_J                                    ; 1942
                                     57  D6 00067 3$:    INCL    COUNT                                        ; 1944
                        56     F8    A0  C0 00069        ADDL2   -8(R0), BLOCKS                               ; 1945
                        5B           57  D1 0006D        CMPL    COUNT, BEST_COUNT                            ; 1947
                                     0B  1B 00070        BLEQU   4$                                           ;
                        5B           57  D0 00072        MOVL    COUNT, BEST_COUNT                            ; 1950
                    18  AE           56  D0 00075        MOVL    BLOCKS, BEST_BLOCKS                          ; 1951
                        55     0C    AE  D0 00079        MOVL    BASE_J, BEST_J                               ; 1952
                        59           56  D1 0007D 4$:    CMPL    BLOCKS, MOST_BLOCKS                          ; 1955
                                     08  1B 00080        BLEQU   5$                                           ;
                        59           56  D0 00082        MOVL    BLOCKS, MOST_BLOCKS                          ; 1958
                    04  AE     0C    AE  D0 00085        MOVL    BASE_J, MOST_J                               ; 1959
                AC      51     10    AE  F3 0008A 5$:    AOBLEQ  16(SP), J, 2$                                ; 1931
                50  04  A2     08    AC  C3 0008F        SUBL3   CACHE_LIMIT, 4(EXTENT_CACHE), BLOCKS_TO_REM  ; 1969
                    04  A2     08    AC  D1 00095        CMPL    CACHE_LIMIT, 4(EXTENT_CACHE)                 ; 1970
                                     02  1B 0009A        BLEQU   6$                                           ;
                                     50  D4 0009C        CLRL    BLOCKS_TO_REM                                ; 1971
                        50     18    AE  D1 0009E 6$:    CMPL    BEST_BLOCKS, BLOCKS_TO_REM                   ; 1973
                                     0C  1E 000A2        BGEQU   7$                                           ;
                        55     04    AE  D0 000A4        MOVL    MOST_J, BEST_J                               ; 1976
                        50           59  D1 000A8        CMPL    MOST_BLOCKS, BLOCKS_TO_REM                   ; 1977
                                     03  1E 000AB        BGEQU   7$                                           ;
                        55           01  D0 000AD        MOVL    #1, BEST_J                                   ; 1978
                               FC  A345  7F 000B0 7$:    PUSHAQ  -4(EXTENT_LIST)[BEST_J]                      ; 1981
                50      9E 00001000  8F  C7 000B4        DIVL3   #4096, @(SP)+, R0                            ;
                        51     1C    BE  D0 000BC        MOVL    @28(SP), R1                                  ;
                        54     3C    A1  3C 000C0        MOVZWL  60(R1), R4                                   ;
            14  AE      50           54  C7 000C4        DIVL3   R4, R0, VBN                                  ;
     55     02  A2      10           00  ED 000C9 8$:    CMPZV   #0, #16, 2(EXTENT_CACHE), BEST_J             ; 1987
                                     01  1E 000CF        BGEQU   9$                                           ;
                                         04 000D1        RET                                                  ;
                        50         6345  7E 000D2 9$:    MOVAQ   (EXTENT_LIST)[BEST_J], R0                    ; 1990
                        56     FC    A0  D0 000D6        MOVL    -4(R0), LBN                                  ;
                    08  AC     04    A2  D1 000DA        CMPL    4(EXTENT_CACHE), CACHE_LIMIT                 ; 1991
                                     29  1A 000DF        BGTRU   11$                                          ;
  04  AC    02  A2      10           00  ED 000E1        CMPZV   #0, #16, 2(EXTENT_CACHE), ENTRY_COUNT        ; 1992
                                     01  1A 000E8        BGTRU   10$                                          ;
                                         04 000EA        RET                                                  ;
                54      56 00001000  8F  C7 000EB 10$:   DIVL3   #4096, LBN, R4                               ; 1993
                        51     1C    BE  D0 000F3        MOVL    @28(SP), R1                                  ;
                        57     3C    A1  3C 000F7        MOVZWL  60(R1), R7                                   ;
                        54           57  C6 000FB        DIVL2   R7, R4                                       ;
                        54     14    AE  D1 000FE        CMPL    VBN, R4                                      ;
                                     06  13 00102        BEQL    11$                                          ;
                               04    AC  D5 00104        TSTL    ENTRY_COUNT                                  ; 1994
                                     01  13 00107        BEQL    11$                                          ;
                                         04 00109        RET                                                  ;
                        58     F8    A0  D0 0010A 11$:   MOVL    -8(R0), BLOCK_COUNT                          ; 1998
                50  04  A2           58  C3 0010E        SUBL3   BLOCK_COUNT, 4(EXTENT_CACHE), R0             ; 1999
                    08  AC           50  D1 00113        CMPL    R0, CACHE_LIMIT                              ;
                                     25  1E 00117        BGEQU   12$                                          ;
  04  AC    02  A2      10           00  ED 00119        CMPZV   #0, #16, 2(EXTENT_CACHE), ENTRY_COUNT        ; 2000
                                     1C  1A 00120        BGTRU   12$                                          ;
                58  04  A2     08    AC  C3 00122        SUBL3   CACHE_LIMIT, 4(EXTENT_CACHE), BLOCK_COUNT    ; 2003
                        50     1C    BE  D0 00128        MOVL    @28(SP), R0                                  ; 2004
                        50           3C  C0 0012C        ADDL2   #60, R0                                      ;
                    50           60  3C 0012F           MOVZWL  (R0), R0                                  ;
                    51     FF  A048  9E 00132           MOVAB   -1(R0)[BLOCK_COUNT], R1                   ;
                    51           50  C6 00137           DIVL2   R0, R1                                    ; 2005
         58         51           50  C5 0013A           MULL3   R0, R1, BLOCK_COUNT                       ;
                          0140   8F  BB 0013E 12$:      PUSHR   #^M<R6,R8>                                ; 2007
               0000V  CF         02  FB 00142           CALLS   #2, REMOVE_EXTENT                         ;
                          0140   8F  BB 00147           PUSHR   #^M<R6,R8>                                ; 2008
               0000V  CF         02  FB 0014B           CALLS   #2, RETURN_BITMAP                         ;
                               FF76  31 00150           BRW     8$                                        ; 1987
                    6D   0000G   CF  9E 00153 13$:      MOVAB   ZERO_ON_ERROR, (FP)                       ; 2019
                            02   A2  B5 00158 14$:      TSTW    2(EXTENT_CACHE)                           ; 2020
                                 1B  13 0015B           BEQL    15$                                       ;
                    56      04   A3  D0 0015D           MOVL    4(EXTENT_LIST), LBN                       ; 2023
                    58           63  D0 00161           MOVL    (EXTENT_LIST), BLOCK_COUNT                ; 2024
                          0140   8F  BB 00164           PUSHR   #^M<R6,R8>                                ; 2025
               0000V  CF         02  FB 00168           CALLS   #2, REMOVE_EXTENT                         ;
                          0140   8F  BB 0016D           PUSHR   #^M<R6,R8>                                ; 2026
               0000V  CF         02  FB 00171           CALLS   #2, RETURN_BITMAP                         ;
                                 E0  11 00176           BRB     14$                                       ; 2020
                            0C   A2  D5 00178 15$:      TSTL    12(EXTENT_CACHE)                          ; 2029
                                 25  13 0017B           BEQL    16$                                       ;
                 24   AE    0C   A2  D0 0017D           MOVL    12(EXTENT_CACHE), LOCK_STATUS+4           ; 2032
                                 7E  7C 00182           CLRQ    -(SP)                                     ; 2037
                                 7E  7C 00184           CLRQ    -(SP)                                     ;
                                 7E  7C 00186           CLRQ    -(SP)                                     ;
                                 7E  D4 00188           CLRL    -(SP)                                     ;
                    7E      4E   8F  9A 0018A           MOVZBL  #78, -(SP)                                ;
                            40   AE  9F 0018E           PUSHAB  LOCK_STATUS                               ;
                    7E           1E  7D 00191           MOVQ    #30, -(SP)                                ;
           00000000G  00         0B  FB 00194           CALLS   #11, SYS$ENQW                             ;
                    04           50  E8 0019B           BLBS    R0, 16$                                   ;
                               FEFF     0019E           BUGW                                              ; 2038
                              0000*     001A0           .WORD   <BUG$_XQPERR!4>                           ;
                    50      1C   BE  D0 001A2 16$:      MOVL    @28(SP), R0                               ; 2040
                    50      58   A0  D0 001A6           MOVL    88(R0), R0                                ;
                 0B   A0         02  8A 001AA           BICB2   #2, 11(R0)                                ;
                                 04     001AE           RET                                               ; 2043

; Routine Size:  431 bytes,    Routine Base:  $CODE$ + 0581
```

```
1 ROUTINE REMOVE_EXTENT (LBN, COUNT) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine removes the indicated number of blocks from the indicated
1 !       extent in the cache. If the total block count of the extent is removed,
1 !       then the extent is eliminated completely.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       REMOVE_EXTENT (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: LBN of extent to remove
1 !       ARG2: count of blocks to remove
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !       extent cache altered
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       EXTENT_CACHE    : REF BBLOCK,   ! pointer to extent cache
2       EXTENT_LIST     : REF BBLOCKVECTOR [,8]; ! pointer to extent list
2
2 BIND_COMMON;
2
2 ! Get the pointer to the extent cache and search it for the LBN. When
2 ! found, squish out the entry.
2 !
2
2 EXTENT_CACHE = .BBLOCK [.CURRENT_VCB[VCB$L_CACHE], VCA$L_EXTCACHE];
2 EXTENT_LIST = EXTENT_CACHE[VCA$Q_EXTLIST];
2
2 INCR J FROM 1 TO .EXTENT_CACHE[VCA$W_EXTCOUNT]
2 DO
3     BEGIN
3     IF .EXTENT_LIST[.J-1, VCA$L_EXTLBN] EQL .LBN
3     THEN
4         BEGIN
4         EXTENT_LIST[.J-1, VCA$L_EXTLBN] = .EXTENT_LIST[.J-1, VCA$L_EXTLBN] + .COUNT;
```

```
; 1118      2101  4          EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] = .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] - .COUNT;
; 1119      2102  4          IF .EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS] NEQ 0 THEN EXITLOOP;
; 1120      2103  4          CH$MOVE ((.EXTENT_CACHE[VCA$W_EXTCOUNT]-.J)*8,
; 1121      2104  4                   EXTENT_LIST[.J, VCA$C_EXTBLOCKS],
; 1122      2105  4                   EXTENT_LIST[.J-1, VCA$L_EXTBLOCKS]);
; 1123      2106  4          EXTENT_CACHE[VCA$W_EXTCOUNT] = .EXTENT_CACHE[VCA$W_EXTCOUNT] - 1;
; 1124      2107  4          EXITLOOP;
; 1125      2108  3          END;
; 1126      2109  2      END;
; 1127      2110  2
; 1128      2111  2  EXTENT_CACHE[VCA$L_EXTTOTAL] = .EXTENT_CACHE[VCA$L_EXTTOTAL] - .COUNT;
; 1129      2112  2
; 1130      2113  2  1
; 1131      2114  1  END;                                    ! end of routine REMOVE_EXTENT



                                                03FC 00000 REMOVE_EXTENT:
                                                               .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                ; 2044
                       50       98    AA   D0 00002            MOVL    -104(BASE), R0                              ; 2091
                       50       58    A0   D0 00006            MOVL    88(R0), R0                                  ;
                       57       04    A0   D0 0000A            MOVL    4(R0), EXTENT_CACHE                         ;
                       56       2C    A7   9E 0000E            MOVAB   44(R7), EXTENT_LIST                         ; 2092
                       59       02    A7   3C 00012            MOVZWL  2(EXTENT_CACHE), R9                         ; 2094
                                      58   D4 00016            CLRL    J                                           ;
                                      3C   11 00018            BRB     2$                                          ;
                                FC  A648   7F 0001A 1$:        PUSHAQ  -4(EXTENT_LIST)[J]                          ; 2097
                   04   AC            9E   D1 0001E            CMPL    @(SP)+, LBN                                 ;
                                      32   12 00022            BNEQ    2$                                          ;
                                FC  A648   7F 00024            PUSHAQ  -4(EXTENT_LIST)[J]                          ; 2100
                       9E       08    AC   C0 00028            ADDL2   COUNT, @(SP)+                               ;
                                F8  A648   7F 0002C            PUSHAQ  -8(EXTENT_LIST)[J]                          ; 2101
                       9E       08    AC   C2 00030            SUBL2   COUNT, @(SP)+                               ;
                                F8  A648   7F 00034            PUSHAQ  -8(EXTENT_LIST)[J]                          ; 2102
                                      9E   D5 00038            TSTL    @(SP)+                                      ;
                                      1E   12 0003A            BNEQ    3$                                          ;
                       50       02    A7   3C 0003C            MOVZWL  2(EXTENT_CACHE), R0                         ; 2103
                       50             58   C2 00040            SUBL2   J, R0                                       ;
                       50             08   C4 00043            MULL2   #8, R0                                      ;
                                F8  A648   7F 00046            PUSHAQ  -8(EXTENT_LIST)[J]                          ; 2105
                                    6648   7F 0004A            PUSHAQ  (EXTENT_LIST)[J]                            ;
       9E              9E             50   28 0004D            MOVC3   R0, @(SP)+, @(SP)+                          ;
                                02    A7   B7 00051            DECW    2(EXTENT_CACHE)                             ; 2106
                                      04   11 00054            BRB     3$                                          ; 2099
       C0              58             59   F3 00056 2$:        AOBLEQ  R9, J, 1$                                   ; 2094
                   04  A7       08    AC   C2 0005A 3$:        SUBL2   COUNT, 4(EXTENT_CACHE)                      ; 2111
                       50             01   D0 0005F            MOVL    #1, R0                                      ; 2114
                                      04      00062            RET                                                 ;

; Routine Size:  99 bytes,    Routine Base:  $CODE$ + 0730
```

```
1 ROUTINE ALLOC_BITMAP (FIB, BLOCKS_NEEDED, START_LBN, BLOCKS_ALLOC, PARTIAL) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine allocates a single contiguous area of disk.
1 !       Mode of allocation is determined by the allocation control
1 !       in the FIB.
1 !
1 ! CALLING SEQUENCE:
1 !       ALLOC_BITMAP (ARG1, ARG2, ARG3, ARG4, ARG5)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of FIB for this operation
1 !       ARG2: number of blocks to allocate
1 !       ARG5: 0 to scan entire bitmap
1 !             1 to scan only currently resident block
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: ADDRESS OF VCB IN PROCESS
1 !       CURRENT_UCB: ADDRESS OF UCB IN PROCESS
1 !
1 ! OUTPUT PARAMETERS:
1 !       ARG3: address of longword to store starting LBN
1 !       ARG4: address of longword to store block count
1 !
1 ! IMPLICIT OUTPUTS:
1 !       LOC_LBN: placement LBN of allocation or 0
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1 if successful allocation
1 !       0 if failure
1 !
1 ! SIDE EFFECTS:
1 !       storage map and VCB modified
1 !
1 !--
1
2 BEGIN
2
2 BUILTIN
2       EDIV;
2
2 MAP
2       FIB             : REF BBLOCK;   ! FIB of request
2
2 LOCAL
2       CLUSTER,                        ! cluster factor of volume
2       QUAD_BLOCKS_NEEDED      : VECTOR [2],   ! Blocks needed as a quadword
2       BITS_NEEDED,                    ! number of map bits to allocate
2       BEGIN_BIT,                      ! first bitmap bit looked at
2       START_BIT,                      ! bit address in storage map
2       BIT_COUNT,                      ! number of bits to scan
2       FIRST_SET,                      ! start of free area
2       BITS_SCANNED,                   ! number of bits processed by scanner
```

```
2               END_BIT,                        ! last bit processed
2               BEST_STARTBIT,                  ! start of largest free area
2               BEST_BITSFOUND,                 ! size of largest free area
2               CYL_SIZE,                       ! volume cylinder size in clusters
2               CYL_BOUNDARY,                   ! bit address of next cylinder boundary
2               DUMMY;                          ! Throw-away remainder from EDIV
2
2       LABEL
2               MAP_SCAN;                       ! code block to scan the storage map
2
2       BIND_COMMON;
2
2       ! Adjust the desired block count to a bit count through the volume
2       ! cluster factor. Set up the running parameters.
2       !
2
2       CLUSTER = .CURRENT_VCB[VCB$W_CLUSTER];
2       QUAD_BLOCKS_NEEDED[0] = .BLOCKS_NEEDED + .CLUSTER - 1;
2       QUAD_BLOCKS_NEEDED[1] = 0;
2       EDIV (CLUSTER, QUAD_BLOCKS_NEEDED, BITS_NEEDED, DUMMY);
2       BEST_BITSFOUND = 0;
2       START_BIT = BEGIN_BIT = .CURRENT_VCB[VCB$B_SBMAPVBN] * 4096;
2
2       CYL_SIZE = .CURRENT_UCB[UCB$B_SECTORS]
2                * .CURRENT_UCB[UCB$B_TRACKS]
2                / .CURRENT_VCB[VCB$B_BLOCKFACT];
2
2       ! Get placement data if specified. If the placement LBN is garbage, fail if
2       ! exact placement is called for, else forget it.
2       !
2
2       IF .LOC_LBN NEQ 0
2       THEN
3           BEGIN
3           IF .LOC_LBN GEQU .CURRENT_UCB[UCB$L_MAXBLOCK]
3           THEN
4               BEGIN
4               IF .FIB[FIB$V_EXACT]
4               THEN RETURN 0
4               ELSE LOC_LBN = 0;
3               END;
3           START_BIT = BEGIN_BIT = .LOC_LBN / .CLUSTER;
2           END;
2
2       ! The outer loop potentially scans the map twice: once from the given starting
2       ! point through to the end and then from beginning to end, if necessary to
2       ! locate a large contiguous area with a bad start.
2       !
2
2       MAP_SCAN:
3           BEGIN
3           WHILE 1 DO
4               BEGIN
4               BIT_COUNT = .CURRENT_UCB[UCB$L_MAXBLOCK] / .CLUSTER - .START_BIT;
4               IF .PARTIAL
4               THEN BIT_COUNT = MINU (.BIT_COUNT, 4096);
4
```

```
4 ! Now scan the bitmap for the first free block. Having found it, scan
4 ! to see how many free blocks there are there. If it is a non-contiguous
4 ! allocation, accept the blocks regardless. If it is contiguous, and the
4 ! free area is too small, keep looking.
4 !
4
4       WHILE 1 DO
5           BEGIN
5
5           IF .LOC_LBN EQL 0
5           THEN
6               BEGIN
6               IF BITSCAN (FIND_SET, .START_BIT, .BIT_COUNT, FIRST_SET, BITS_SCANNED)
6               THEN EXITLOOP;          ! out if end of map
6
6               BIT_COUNT = .BIT_COUNT - .BITS_SCANNED;
6               END
5           ELSE
5               FIRST_SET = .START_BIT;
5
5 ! If on cylinder allocation is requested, see if sufficient space remains
5 ! between the current point and the next cylinder boundary. If not, nudge
5 ! to the next cylinder boundary if exact is not specified. If exact is
5 ! specified, we allow for a nudge of 1 cluster to allow for the vagaries
5 ! of cluster boundaries.
5 !
5
5           IF .FIB[FIB$V_ONCYL]
5           THEN
6               BEGIN
6               CYL_BOUNDARY = ((.FIRST_SET*.CLUSTER) /.CYL_SIZE + 1) * .CYL_SIZE;
6               IF .CYL_BOUNDARY/.CLUSTER - .FIRST_SET LEQU .BITS_NEEDED
6               THEN
7                   BEGIN
7                   CYL_BOUNDARY = (.CYL_BOUNDARY + .CLUSTER - 1) / .CLUSTER;
7                   IF .FIB[FIB$V_EXACT]
7                   AND .LOC_LBN NEQ 0
7                   AND .CYL_BOUNDARY - .FIRST_SET GTRU 1
7                   THEN RETURN 0;
7
7                   BIT_COUNT = .BIT_COUNT - .CYL_BOUNDARY + .FIRST_SET;
7                   IF .BIT_COUNT LEQ 0 THEN EXITLOOP;
7                   FIRST_SET = .CYL_BOUNDARY;
6                   END;
5               END;
5
5           BITSCAN (FIND_CLEAR, .FIRST_SET, MIN (.BIT_COUNT, .BITS_NEEDED),
5                       START_BIT, BITS_SCANNED);
5
5           BIT_COUNT = .BIT_COUNT - .BITS_SCANNED;
5
5           IF .BITS_SCANNED GTRU .BEST_BITSFOUND
5           THEN
6               BEGIN
6               BEST_STARTBIT = .FIRST_SET;
6               BEST_BITSFOUND = .BITS_SCANNED;
5               END;
```

```
5
5               IF .BEST_BITSFOUND GEQU .BITS_NEEDED
7               OR (NOT (.FIB[FIB$V_ALCON] OR .FIB[FIB$V_ALCONB])
6                   AND .BEST_BITSFOUND NEQ 0)
5               THEN LEAVE MAP_SCAN;            ! found what we were after
5
5               IF .BIT_COUNT EQL 0
5               THEN EXITLOOP;                  ! end of storage map
5
5 ! If an exact placement was asked for and we didn't get it, it's all over.
5 ! Otherwise, forget placement and continue scanning normally.
5 !
5
5               IF .FIB[FIB$V_ALCON]
5               AND .FIB[FIB$V_EXACT]
5               AND .LOC_LBN NEQ 0
5               THEN RETURN 0;
5               LOC_LBN = 0;
5
4               END;                            ! end of map scan loop
4
4 ! We get here when we run into the end of the storage map. If the scan
4 ! started in the middle, do it once more from the top.
4 !
4
4           IF .BEGIN_BIT EQL 0
4           OR .PARTIAL
4           THEN LEAVE MAP_SCAN;
4           BEGIN_BIT = START_BIT = 0;
3           END;                                ! end of outer loop
2       END;                                    ! end of block MAP_SCAN
2
2 ! We have either found a cluster of free blocks suitable to the occasion
2 ! or we have searched the entire map. If nothing was found, or for a
2 ! normal contiguous request, return error if the number of blocks is
2 ! insufficient; otherwise, allocate the blocks.
2 !
2
2 IF .BEST_BITSFOUND EQL 0
3 OR (.FIB[FIB$V_ALCON] AND NOT .FIB[FIB$V_ALCONB]
3     AND .BEST_BITSFOUND LSSU .BITS_NEEDED)
2 THEN
3     BEGIN
3     USER_STATUS[1] = .BEST_BITSFOUND * .CLUSTER;
3     RETURN 0;
2     END;
2
2 BITSCAN (CLEAR_BITS, .BEST_STARTBIT, .BEST_BITSFOUND, END_BIT, BITS_SCANNED);
2
2 CURRENT_VCB[VCB$B_SBMAPVBN] = .END_BIT / 4096;
2
2 .START_LBN = .BEST_STARTBIT * .CLUSTER;
2 .BLOCKS_ALLOC = .BEST_BITSFOUND * .CLUSTER;
2
2 RETURN 1;
2
1 END;                                          ! end of routine ALLOC_BITMAP
```

```
                                              0BFC 00000 ALLOC_BITMAP:
                                                         .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R11                 ; 2115
                          5E              1C   C2 00002   SUBL2   #28, SP                                          ;
                          56      80  AA  9E 00005        MOVAB   -128(BASE), R6                                   ; 2180
                          59      20  AA  9E 00009        MOVAB   32(BASE), R9                                     ;
                          50      98  AA  D0 0000D        MOVL    -104(BASE), R0                                   ; 2188
                          54      3C  A0  3C 00011        MOVZWL  60(R0), CLUSTER                                  ;
              50          54      08  AC  C1 00015        ADDL3   BLOCKS_NEEDED, CLUSTER, R0                       ; 2189
                    14    AE      FF  A0  9E 0001A        MOVAB   -1(R0), QUAD_BLOCKS_NEEDED                       ;
                                  18  AE  D4 0001F        CLRL    QUAD_BLOCKS_NEEDED+4                             ; 2190
     50       58    14    AE          54  7B 00022        EDIV    CLUSTER, QUAD_BLOCKS_NEEDED, BITS_NEEDED, -      ; 2191
                                                                  DUMMY
                                          55  D4 00028    CLRL    BEST_BITSFOUND                                   ; 2192
                          50      98  AA  D0 0002A        MOVL    -104(BASE), R0                                   ; 2193
                          57      3B  A0  9A 0002E        MOVZBL  59(R0), BEGIN_BIT                                ;
              57          57          0C  78 00032        ASHL    #12, BEGIN_BIT, BEGIN_BIT                        ;
                    08    AE          57  D0 00036        MOVL    BEGIN_BIT, START_BIT                             ;
                          50      94  AA  D0 0003A        MOVL    -108(BASE), R0                                   ; 2195
                          51      44  A0  9A 0003E        MOVZBL  68(R0), R1                                       ; 2196
                          50      45  A0  9A 00042        MOVZBL  69(R0), R0                                       ;
                          50          51  C4 00046        MULL2   R1, R0                                           ;
                          51      98  AA  D0 00049        MOVL    -104(BASE), R1                                   ; 2197
                          5B      52  A1  9A 0004D        MOVZBL  82(R1), CYL_SIZE                                 ;
              5B          50          5B  C7 00051        DIVL3   CYL_SIZE, R0, CYL_SIZE                           ;
                                      69  D5 00055        TSTL    (R9)                                             ; 2203
                                      20  13 00057        BEQL    3$                                               ;
                          50      94  AA  D0 00059        MOVL    -108(BASE), R0                                   ; 2206
                  00B0    C0          69  D1 0005D        CMPL    (R9), 176(R0)                                    ;
                                      0D  1F 00062        BLSSU   2$                                               ;
                          50      04  AC  D0 00064        MOVL    FIB, R0                                          ; 2209
                          03      20  A0  E9 00068        BLBC    32(R0), 1$                                       ;
                                    015F  31 0006C        BRW     20$                                              ;
                                      69  D4 0006F  1$:   CLRL    (R9)                                             ; 2211
              57          69          54  C7 00071  2$:   DIVL3   CLUSTER, (R9), BEGIN_BIT                         ; 2213
                    08    AE          57  D0 00075        MOVL    BEGIN_BIT, START_BIT                             ;
                          50      94  AA  D0 00079  3$:   MOVL    -108(BASE), R0                                   ; 2225
              50  00B0    C0          54  C7 0007D        DIVL3   CLUSTER, 176(R0), R0                             ;
              53          50      08  AE  C3 00083        SUBL3   START_BIT, R0, BIT_COUNT                         ;
                          14      14  AC  E9 00088        BLBC    PARTIAL, 5$                                      ; 2226
                          50          53  D0 0008C        MOVL    BIT_COUNT, R0                                    ; 2227
              00001000    8F          50  D1 0008F        CMPL    R0, #4096                                        ;
                                      05  1B 00096        BLEQU   4$                                               ;
                          50    1000  8F  3C 00098        MOVZWL  #4096, R0                                        ;
                          53          50  D0 0009D  4$:   MOVL    R0, BIT_COUNT                                    ;
                                      69  D5 000A0  5$:   TSTL    (R9)                                             ; 2238
                                      1E  12 000A2        BNEQ    7$                                               ;
                                  0C  AE  9F 000A4        PUSHAB  BITS_SCANNED                                     ; 2241
                                  08  AE  9F 000A7        PUSHAB  FIRST_SET                                        ;
                                      53  DD 000AA        PUSHL   BIT_COUNT                                        ;
                                  14  AE  DD 000AC        PUSHL   START_BIT                                        ;
                                      7E  D4 000AF        CLRL    -(SP)                                            ;
                  0000V   CF          05  FB 000B1        CALLS   #5, BITSCAN                                      ;
```

```
            03            50  E9 000B6           BLBC    R0, 6$                                    ;
                        00B4  31 000B9           BRW     16$                                       ;
            53      0C    AE  C2 000BC 6$:       SUBL2   BITS_SCANNED, BIT_COUNT                   ; 2244
                          05  11 000C0           BRB     8$                                        ; 2238
       04   AE      08    AE  D0 000C2 7$:       MOVL    START_BIT, FIRST_SET                      ; 2247
            50      04    AC  D0 000C7 8$:       MOVL    FIB, R0                                   ; 2256
 45    20   A0            01  E1 000CB           BBC     #1, 32(R0), 10$                           ;
 51    04   AE            54  C5 000D0           MULL3   CLUSTER, FIRST_SET, R1                    ; 2259
            51            5B  C6 000D5           DIVL2   CYL_SIZE, R1                              ;
                          51  D6 000D8           INCL    R1                                        ;
 52         51            5B  C5 000DA           MULL3   CYL_SIZE, R1, CYL_BOUNDARY                ;
 51         52            54  C7 000DE           DIVL3   CLUSTER, CYL_BOUNDARY, R1                 ; 2260
            51      04    AE  C2 000E2           SUBL2   FIRST_SET, R1                             ;
            58            51  D1 000E6           CMPL    R1, BITS_NEEDED                           ;
                          2A  1A 000E9           BGTRU   10$                                       ;
            51      FF  A442  9E 000EB           MOVAB   -1(CLUSTER)[CYL_BOUNDARY], R1             ; 2263
 52         51            54  C7 000F0           DIVL3   CLUSTER, R1, CYL_BOUNDARY                 ;
            0E      20    A0  E9 000F4           BLBC    32(R0), 9$                                ; 2264
                          69  D5 000F8           TSTL    (R9)                                      ; 2265
                          0A  13 000FA           BEQL    9$                                        ;
 51         52      04    AE  C3 000FC           SUBL3   FIRST_SET, CYL_BOUNDARY, R1               ; 2266
            01            51  D1 00101           CMPL    R1, #1                                    ;
                          63  1A 00104           BGTRU   14$                                       ;
 50         53            52  C3 00106 9$:       SUBL3   CYL_BOUNDARY, BIT_COUNT, R0               ; 2269
 53         50      04    AE  C1 0010A           ADDL3   FIRST_SET, R0, BIT_COUNT                  ;
                          5F  15 0010F           BLEQ    16$                                       ; 2270
       04   AE            52  D0 00111           MOVL    CYL_BOUNDARY, FIRST_SET                   ; 2271
                    0C    AE  9F 00115 10$:      PUSHAB  BITS_SCANNED                              ; 2275
                    0C    AE  9F 00118           PUSHAB  START_BIT                                 ;
                          53  DD 0011B           PUSHL   BIT_COUNT                                 ;
            58            6E  D1 0011D           CMPL    (SP), BITS_NEEDED                         ;
                          03  15 00120           BLEQ    11$                                       ;
            6E            58  D0 00122           MOVL    BITS_NEEDED, (SP)                         ;
                    10    AE  DD 00125 11$:      PUSHL   FIRST_SET                                 ;
                          01  DD 00128           PUSHL   #1                                        ;
    0000V   CF            05  FB 0012A           CALLS   #5, BITSCAN                               ;
            53      0C    AE  C2 0012F           SUBL2   BITS_SCANNED, BIT_COUNT                   ; 2278
            55      0C    AE  D1 00133           CMPL    BITS_SCANNED, BEST_BITSFOUND              ; 2280
                          08  1B 00137           BLEQU   12$                                       ;
            6E      04    AE  D0 00139           MOVL    FIRST_SET, BEST_STARTBIT                  ; 2283
            55      0C    AE  D0 0013D           MOVL    BITS_SCANNED, BEST_BITSFOUND              ; 2284
            58            55  D1 00141 12$:      CMPL    BEST_BITSFOUND, BITS_NEEDED               ; 2287
                          3A  1E 00144           BGEQU   17$                                       ;
            50      04    AC  D0 00146           MOVL    FIB, R0                                   ; 2288
            09      16    A0  E8 0014A           BLBS    22(R0), 13$                               ;
 04    16   A0            01  E0 0014E           BBS     #1, 22(R0), 13$                           ;
                          55  D5 00153           TSTL    BEST_BITSFOUND                            ; 2289
                          29  12 00155           BNEQ    17$                                       ;
                          53  D5 00157 13$:      TSTL    BIT_COUNT                                 ; 2292
                          15  13 00159           BEQL    16$                                       ;
            50      04    AC  D0 0015B           MOVL    FIB, R0                                   ; 2299
            08      16    A0  E9 0015F           BLBC    22(R0), 15$                               ;
            04      20    A0  E9 00163           BLBC    32(R0), 15$                               ; 2300
                          69  D5 00167           TSTL    (R9)                                      ; 2301
                          63  12 00169 14$:      BNEQ    20$                                       ;
                          69  D4 0016B 15$:      CLRL    (R9)                                      ; 2303
                        FF30  31 0016D           BRW     5$                                        ; 2235
```

```
                                        57  D5 00170 16$:    TSTL    BEGIN_BIT                                ; 2311
                                        0C  13 00172         BEQL    17$                                      ;
                    08          14      AC  E8 00174         BLBS    PARTIAL, 17$                             ; 2312
                                08      AE  D4 00178         CLRL    START_BIT                                ; 2314
                                        57  D4 0017B         CLRL    BEGIN_BIT                                ;
                                      FEF9  31 0017D         BRW     3$                                       ; 2223
                                        55  D5 00180 17$:    TSTL    BEST_BITSFOUND                           ; 2324
                                        12  13 00182         BEQL    18$                                      ;
                    50          04      AC  D0 00184         MOVL    FIB, R0                                  ; 2325
                    11          16      A0  E9 00188         BLBC    22(R0), 19$                              ;
        0C          16          A0      01  E0 0018C         BBS     #1, 22(R0), 19$                          ;
                    58                  55  D1 00191         CMPL    BEST_BITSFOUND, BITS_NEEDED              ; 2326
                                        07  1E 00194         BGEQU   19$                                      ;
  04    A6          55                  54  C5 00196 18$:    MULL3   CLUSTER, BEST_BITSFOUND, 4(R6)           ; 2329
                                        31  11 0019B         BRB     20$                                      ; 2330
                                0C      AE  9F 0019D 19$:    PUSHAB  BITS_SCANNED                             ; 2333
                                14      AE  9F 001A0         PUSHAB  END_BIT                                  ;
                                        55  DD 001A3         PUSHL   BEST_BITSFOUND                           ;
                                0C      AE  DD 001A5         PUSHL   BEST_STARTBIT                            ;
                                        03  DD 001A8         PUSHL   #3                                       ;
               0000V    CF              05  FB 001AA         CALLS   #5, BITSCAN                              ;
                    50          98      AA  D0 001AF         MOVL    -104(BASE), R0                           ; 2335
        51          10  AE 00001000     8F  C7 001B3         DIVL3   #4096, END_BIT, R1                       ;
                    3B          A0      51  90 001BC         MOVB    R1, 59(R0)                               ;
  0C    BC          6E                  54  C5 001C0         MULL3   CLUSTER, BEST_STARTBIT, @START_LBN       ; 2337
  10    BC          55                  54  C5 001C5         MULL3   CLUSTER, BEST_BITSFOUND, @BLOCKS_ALLOC   ; 2338
                    50                  01  D0 001CA         MOVL    #1, R0                                   ; 2340
                                            04 001CD         RET                                              ;
                                        50  D4 001CE 20$:    CLRL    R0                                       ; 2342
                                            04 001D0         RET                                              ;

; Routine Size:  465 bytes,    Routine Base:  $CODE$ + 0793
```

```
1 ROUTINE RETURN_BITMAP (START_LBN, BLOCK_COUNT) : L_NORM NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine returns a single contiguous area to the storage map.
1 !
1 ! CALLING SEQUENCE:
1 !       RETURN_BITMAP (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: starting LBN to free
1 !       ARG2: number of blocks to free
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: VCB of volume
1 !       CURRENT_UCB: UCB of device
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       storage map and VCB modified
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       START_BIT,                      ! starting bit number in storage map
2       BIT_COUNT,                      ! number of bits to set
2       DUMMY1,                         ! dummies to receive return data
2       DUMMY2;                         ! from BITSCAN, which is not used
2
2 BIND_COMMON;
2
2 ! First check the blocks being returned against the volume size.
2 !
2
2 IF .START_LBN + .BLOCK_COUNT GTRU .CURRENT_UCB[UCB$L_MAXBLOCK]
2 THEN BUG_CHECK (EXTCACHIV, FATAL, 'Contents of extent cache is garbage');
2
2 ! Divide down by the volume cluster factor to convert blocks to storage
2 ! map bits. If there are non-zero remainders, reject the operation on grounds
2 ! of a bad file header.
2 !
2
2 IF .START_LBN MOD .CURRENT_VCB[VCB$W_CLUSTER] NEQ 0
2 THEN BUG_CHECK (EXTCACHIV, FATAL, 'Contents of extent cache is garbage');
2 START_BIT = .START_LBN / .CURRENT_VCB[VCB$W_CLUSTER];
```

```
; 1419          2400  2
; 1420          2401  2 IF .BLOCK_COUNT MOD .CURRENT_VCB[VCB$W_CLUSTER] NEQ 0
; 1421          2402  2 THEN BUG_CHECK (EXTCACHIV, FATAL, 'Contents of extent cache is garbage');
; 1422          2403  2 BIT_COUNT = .BLOCK_COUNT / .CURRENT_VCB[VCB$W_CLUSTER];
; 1423          2404  2
; 1424          2405  2 ! Call the bit scanner to set the appropriate
; 1425          2406  2 ! bits. Finally update the volume free block count.
; 1426          2407  2 !
; 1427          2408  2
; 1428          2409  2 BITSCAN (SET_BITS, .START_BIT, .BIT_COUNT, DUMMY1, DUMMY2);
; 1429          2410  2
; 1430          2411  1 END;                                    ! end of routine RETURN_BITMAP


                                                                    .EXTRN  BUGS_EXTCACHIV

                                              0000 00000 RETURN_BITMAP:
                                                                    .WORD   Save nothing                               ; 2343
                                  5E            08  C2 00002        SUBL2   #8, SP                                     ;
                   51       04    AC      08    AC  C1 00005        ADDL3   BLOCK_COUNT, START_LBN, R1                 ; 2389
                                  50      94    AA  D0 0000B        MOVL    -108(BASE), R0                             ;
                          00B0    C0            51  D1 0000F        CMPL    R1, 176(R0)                                ;
                                                04  1B 00014        BLEQU   1$                                         ;
                                                  FEFF 00016        BUGW                                               ; 2390
                                                 0000* 00018        .WORD   <BUG$_EXTCACHIV!4>                         ;
                                  50      98    AA  D0 0001A 1$:    MOVL    -104(BASE), R0                             ; 2397
                                  50      3C    A0  3C 0001E        MOVZWL  60(R0), R0                                 ;
       7E          00       04    AC            01  7A 00022        EMUL    #1, START_LBN, #0, -(SP)                   ;
       50          50             8E            50  7B 00028        EDIV    R0, (SP)+, R0, R0                          ;
                                                50  D5 0002D        TSTL    R0                                         ;
                                                04  13 0002F        BEQL    2$                                         ;
                                                  FEFF 00031        BUGW                                               ; 2398
                                                 0000* 00033        .WORD   <BUG$_EXTCACHIV!4>                         ;
                                  50      98    AA  D0 00035 2$:    MOVL    -104(BASE), R0                             ; 2399
                                  51      3C    A0  3C 00039        MOVZWL  60(R0), START_BIT                          ;
                   51       04    AC            51  C7 0003D        DIVL3   START_BIT, START_LBN, START_BIT            ;
                                  50      98    AA  D0 00042        MOVL    -104(BASE), R0                             ; 2401
                                  50      3C    A0  3C 00046        MOVZWL  60(R0), R0                                 ;
       7E          00       08    AC            01  7A 0004A        EMUL    #1, BLOCK_COUNT, #0, -(SP)                 ;
       50          50             8E            50  7B 00050        EDIV    R0, (SP)+, R0, R0                          ;
                                                50  D5 00055        TSTL    R0                                         ;
                                                04  13 00057        BEQL    3$                                         ;
                                                  FEFF 00059        BUGW                                               ; 2402
                                                 0000* 0005B        .WORD   <BUG$_EXTCACHIV!4>                         ;
                                  50      98    AA  D0 0005D 3$:    MOVL    -104(BASE), R0                             ; 2403
                                  50      3C    A0  3C 00061        MOVZWL  60(R0), BIT_COUNT                          ;
                   50       08    AC            50  C7 00065        DIVL3   BIT_COUNT, BLOCK_COUNT, BIT_COUNT          ;
                                                5E  DD 0006A        PUSHL   SP                                         ; 2409
                                          08    AE  9F 0006C        PUSHAB  DUMMY1                                     ;
                                                50  DD 0006F        PUSHL   BIT_COUNT                                  ;
                                                51  DD 00071        PUSHL   START_BIT                                  ;
                                                02  DD 00073        PUSHL   #2                                         ;
                          0000V   CF            05  FB 00075        CALLS   #5, BITSCAN                                ;
                                                    04 0007A        RET                                                ; 2411

; Routine Size:  123 bytes,    Routine Base:  $CODE$ + 0964
```

```
1 ROUTINE BITSCAN (MODE, STARTBIT, BITCOUNT, STOPBIT, LENGTHFOUND) : L_NORM =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is the basic bitmap scanner. It scans the bitmap
1 !       over the specified number of bits, performing the operation
1 !       specified by the mode.
1 !
1 ! CALLING SEQUENCE:
1 !       BITSCAN (ARG1, ARG2, ARG3, ARG4, ARG5)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: mode of operation - see module preface
1 !       ARG2: starting bit address in bitmap
1 !       ARG3: maximum number of bits to process
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: address of VCB in process
1 !
1 ! OUTPUT PARAMETERS:
1 !       ARG4: address of longword to receive ending bit address
1 !       ARG5: address of longword to receive number of bits scanned
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1 if maximum bit count processed
1 !       0 if not
1 !
1 ! SIDE EFFECTS:
1 !       bitmap blocks may be altered, read, and written
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       COUNT,                          ! number of bits to go
2       BLOCK,                          ! current bitmap block number
2       CBYTE,                          ! current byte offset in block
2       CBIT,                           ! current bit number within byte
2       BYTELIM,                        ! number of bytes to scan
2       BITLIM,                         ! number of bits to scan
2       BUFFER,                         ! address of bitmap buffer
2       ENDBYTE,                        ! end of current byte scan
2       ENDBIT;                         ! end of current bit scan
2
2 BIND_COMMON;
2
2 EXTERNAL ROUTINE
2       MARK_DIRTY      : L_NORM,       ! mark buffer for writeback
2       READ_BLOCK      : L_NORM;       ! read a disk block
2
2
```

```
2 ! Initialize by setting the count and setting up the pointers to
2 ! the starting position. Read the first map block. The case of a
2 ! zero count is handled specially to avoid bitmap edge problems.
2 !
2
2 COUNT = .BITCOUNT;
2 IF .COUNT EQL 0
2 THEN
3     BEGIN
3     .LENGTHFOUND = 0;
3     .STOPBIT = .STARTBIT;
3     RETURN 1;
2     END;
2
2 BLOCK = .STARTBIT<12,20>;
2 IF .BLOCK GEQU .CURRENT_VCB[VCB$B_SBMAPSIZE]
2 THEN BUG_CHECK (BADSBMBLK, FATAL, 'ACP tried to reference off end of bitmap');
2
2 IF .BLOCK+1 EQL .BITMAP_VBN
2 AND .CURRENT_RVN EQL .BITMAP_RVN
2 THEN
2     BUFFER = .BITMAP_BUFFER
2 ELSE
3     BEGIN
3     BITMAP_VBN = 0;
3     BUFFER = READ_BLOCK (.BLOCK+.CURRENT_VCB[VCB$L_SBMAPLBN], 1, BITMAP_TYPE);
3     BITMAP_VBN = .BLOCK+1;
3     BITMAP_RVN = .CURRENT_RVN;
3     BITMAP_BUFFER = .BUFFER;
2     END;
2
2 CBYTE = .BUFFER + .STARTBIT<3,9>;
2 CBIT = .STARTBIT<0,3>;
2
2 ! The outer loop allows us to use the same set of bit processing instructions
2 ! for the odd bits at both the start and end of the scan.
2 !
2
2 WHILE 1 DO
3     BEGIN
3
3 ! Process bits from the starting position up to the first byte boundary.
3 !
3
3     BITLIM = MIN (8 - .CBIT, .COUNT);    ! max number of bits to scan
3     CASE .MODE FROM 0 TO 3 OF
3         SET
3         [FIND_SET]:     FFS (CBIT, BITLIM, .CBYTE, ENDBIT);
3
3         [FIND_CLEAR]:   FFC (CBIT, BITLIM, .CBYTE, ENDBIT);
3
4         [SET_BITS]:     BEGIN
4                         (.CBYTE)<.CBIT, .BITLIM> = -1;
4                         ENDBIT = .CBIT + .BITLIM;
3                         END;
3
4         [CLEAR_BITS]:   BEGIN
```

```
 4                      (.CBYTE)<.CBIT, .BITLIM> = 0;
 4                      ENDBIT = .CBIT + .BITLIM;
 3                      END;
 3
 3          TES;
 3
 3  ! Update the counters and pointers.
 3  !
 3
 3      COUNT = .COUNT - (.ENDBIT - .CBIT);
 3
 3  ! If we are now positioned on a byte boundary, we can process the bitmap
 3  ! on a byte by byte basis. Page through the bitmap until the count runs out.
 3  !
 3
 3      IF .COUNT EQL 0 OR .ENDBIT NEQ 8 THEN EXITLOOP;
 3
 3      CBYTE = .CBYTE + 1;
 3      CBIT = 0;
 3
 3      WHILE 1 DO
 4          BEGIN
 4          BYTELIM = MIN (.COUNT/8, 512 - (.CBYTE-.BUFFER));
 4
 4          CASE .MODE FROM 0 TO 3 OF
 4              SET
 4
 4              [FIND_SET]:     ENDBYTE = CH$FIND_NOT_CH (.BYTELIM, .CBYTE, 0);
 4
 4              [FIND_CLEAR]:   ENDBYTE = CH$FIND_NOT_CH (.BYTELIM, .CBYTE, 255);
 4
 4              [SET_BITS]:     ENDBYTE = CH$FILL (255, .BYTELIM, .CBYTE);
 4
 4              [CLEAR_BITS]:   ENDBYTE = CH$FILL (0, .BYTELIM, .CBYTE);
 4
 4              TES;
 4
 4          IF CH$FAIL (.ENDBYTE) THEN ENDBYTE = .CBYTE + .BYTELIM;
 4
 4  ! If the count runs out or we run into an end condition leave the loop.
 4  ! Otherwise read the next block, wrapping around the end of the bitmap
 4  ! when necessary, and loop.
 4  !
 4
 4          COUNT = .COUNT - (.ENDBYTE - .CBYTE) * 8;
 4          IF .ENDBYTE - .BUFFER NEQ 512 OR .COUNT EQL 0 THEN EXITLOOP;
 4
 4          CASE .MODE FROM MINU (SET_BITS, CLEAR_BITS) TO MAXU (SET_BITS, CLEAR_BITS) OF
 4              SET
 4
 4              [SET_BITS, CLEAR_BITS]: MARK_DIRTY (.BUFFER);
 4
 4              [INRANGE, OUTRANGE]: 0;
 4
 4              TES;
 4
 4          BLOCK = .BLOCK + 1;
```

```
; 1603          2583  4         IF .BLOCK GEQU .CURRENT_VCB[VCB$B_SBMAPSIZE]
; 1604          2584  4         THEN BUG_CHECK (BADSBMBLK, FATAL, 'ACP tried to reference off end of bitmap');
; 1605          2585  4
; 1606          2586  4         BITMAP_VBN = 0;
; 1607          2587  4         BUFFER = READ_BLOCK (.BLOCK+.CURRENT_VCB[VCB$L_SBMAPLBN], 1, BITMAP_TYPE);
; 1608          2588  4         BITMAP_VBN = .BLOCK+1;
; 1609          2589  4         BITMAP_BUFFER = .BUFFER;
; 1610          2590  4         CBYTE = .BUFFER;
; 1611          2591  3         END;                            ! end of block scan loop
; 1612          2592  3
; 1613          2593  3 ! We have either found the desired end condition or the count will run
; 1614          2594  3 ! out within the next byte. Process the final byte bit by bit.
; 1615          2595  3 !
; 1616          2596  3
; 1617          2597  3     IF .COUNT EQL 0 THEN EXITLOOP;
; 1618          2598  3     CBYTE = .ENDBYTE;
; 1619          2599  2     END;                                ! end of major loop
; 1620          2600  2
; 1621          2601  2 ! Scan is completed. Mark the buffer dirty if necessary and return the
; 1622          2602  2 ! output values.
; 1623          2603  2 !
; 1624          2604  2
; 1625          2605  2 CASE .MODE FROM MINU (SET_BITS, CLEAR_BITS) TO MAXU (SET_BITS, CLEAR_BITS) OF
; 1626          2606  2     SET
; 1627          2607  2
; 1628          2608  2     [SET_BITS, CLEAR_BITS]: MARK_DIRTY (.BUFFER);
; 1629          2609  2
; 1630          2610  2     [INRANGE, OUTRANGE]: 0;
; 1631          2611  2
; 1632          2612  2     TES;
; 1633          2613  2
; 1634          2614  2 .LENGTHFOUND = .BITCOUNT - .COUNT;
; 1635          2615  2 .STOPBIT = .STARTBIT + ..LENGTHFOUND;
; 1636          2616  2 RETURN .COUNT EQL 0;
; 1637          2617  2
; 1638          2618  1 END;                                    ! end of routine BITSCAN


                                                              .EXTRN  MARK_DIRTY, READ_BLOCK
                                                              .EXTRN  BUG$_BADSBMBLK

                                      0BFC 00000 BITSCAN:.WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R11          ; 2412
                        5E         0C   C2 00002         SUBL2   #12, SP                                   ;
                             B4    AA   9F 00005         PUSHAB  -76(BASE)                                 ; 2460
                        59   0C    AC   D0 00008         MOVL    BITCOUNT, COUNT                           ; 2474
                                   0C   12 0000C         BNEQ    1$                                        ; 2475
                             14    BC   D4 0000E         CLRL    @LENGTHFOUND                              ; 2478
                   10   BC   08    AC   D0 00011         MOVL    STARTBIT, @STOPBIT                        ; 2479
                        50         01   D0 00016         MOVL    #1, R0                                    ; 2480
                                        04 00019         RET                                               ;
     57      09   AC    14         04   EF 0001A 1$:     EXTZV   #4, #20, STARTBIT+1, BLOCK                ; 2483
                        50   98    AA   D0 00020         MOVL    -104(BASE), R0                            ; 2484
     57      39   A0    08         00   ED 00024         CMPZV   #0, #8, 57(R0), BLOCK                     ;
                                   04   1A 0002A         BGTRU   2$                                        ;
                                      FEFF 0002C         BUGW                                              ; 2485
                                     0000* 0002E         .WORD   <BUG$_BADSBMBLK!4>                        ;
```

```
                          50      01    A7  9E 00030 2$:     MOVAB   1(R7), R0                                    ; 2487
                  00   BE               50  D1 00034         CMPL    R0, @0(SP)                                   ;
                                        0E  12 00038         BNEQ    3$                                           ;
                  B8   AA         A0    AA  D1 0003A         CMPL    -96(BASE), -72(BASE)                         ; 2488
                                        07  12 0003F         BNEQ    3$                                           ;
                  08   AE         BC    AA  D0 00041         MOVL    -68(BASE), BUFFER                            ; 2490
                                        27  11 00046         BRB     4$                                           ;
                                  00    BE  D4 00048 3$:     CLRL    @0(SP)                                       ; 2493
                                        01  DD 0004B         PUSHL   #1                                           ; 2494
                                        01  DD 0004D         PUSHL   #1                                           ;
                          50      98    AA  D0 0004F         MOVL    -104(BASE), R0                               ;
                               34 B047  9F  00053            PUSHAB  @52(R0)[BLOCK]                               ;
              0000G    CF               03  FB 00057         CALLS   #3, READ_BLOCK                               ;
                  08   AE               50  D0 0005C         MOVL    R0, BUFFER                                   ;
                  00   BE         01    A7  9E 00060         MOVAB   1(R7), @0(SP)                                ; 2495
                  B8   AA         A0    AA  D0 00065         MOVL    -96(BASE), -72(BASE)                         ; 2496
                  BC   AA         08    AE  D0 0006A         MOVL    BUFFER, -68(BASE)                            ; 2497
      56      08   AC             09    03  EF 0006F 4$:     EXTZV   #3, #9, STARTBIT, CBYTE                      ; 2500
                          56      08    AE  C0 00075         ADDL2   BUFFER, CBYTE                                ;
      58      08   AC             03    00  EF 00079         EXTZV   #0, #3, STARTBIT, CBIT                       ; 2501
                   50             08    58  C3 0007F 5$:     SUBL3   CBIT, #8, R0                                 ; 2513
                                  59    50  D1 00083         CMPL    R0, COUNT                                    ;
                                        03  15 00086         BLEQ    6$                                           ;
                                  50    59  D0 00088         MOVL    COUNT, R0                                    ;
                                  5B    50  D0 0008B 6$:     MOVL    R0, BITLIM                                   ;
                   03             00 04 AC  CF 0008E         CASEL   MODE, #0, #3                                 ; 2514
    0023         0018          0010   0008     00093 7$:     .WORD   8$-7$,-                                      ;
                                                                     9$-7$,-                                      ;
                                                                     10$-7$,-                                     ;
                                                                     11$-7$                                       ;
 04    AE          66            5B         58  EA 0009B 8$:     FFS     CBIT, BITLIM, (CBYTE), ENDBIT            ; 2516
                                            1D  11 000A1         BRB     13$                                      ;
 04    AE          66            5B         58  EB 000A3 9$:     FFC     CBIT, BITLIM, (CBYTE), ENDBIT            ; 2518
                                            15  11 000A9         BRB     13$                                      ;
       66          5B            58 FFFFFFFF 8F  F0 000AB 10$:   INSV    #-1, CBIT, BITLIM, (CBYTE)               ; 2521
                                            05  11 000B4         BRB     12$                                      ; 2522
       66          5B            58         00  F0 000B6 11$:    INSV    #0, CBIT, BITLIM, (CBYTE)                ; 2526
               04  AE            58         5B  C1 000BB 12$:    ADDL3   BITLIM, CBIT, ENDBIT                     ; 2527
                   50            58    04   AE  C3 000C0 13$:    SUBL3   ENDBIT, CBIT, R0                         ; 2535
                                 59         50  C0 000C5         ADDL2   R0, COUNT                                ;
                                 03         12  000C8            BNEQ    15$                                      ; 2541
                                       00D6 31  000CA 14$:       BRW     33$                                      ;
                                 08    04   AE  D1 000CD 15$:    CMPL    ENDBIT, #8                               ;
                                            F7  12 000D1         BNEQ    14$                                      ;
                                            56  D6 000D3         INCL    CBYTE                                    ; 2543
                                            58  D4 000D5         CLRL    CBIT                                     ; 2544
                   51            59         08  C7 000D7 16$:    DIVL3   #8, COUNT, R1                            ; 2548
                   50      08    AE         56  C3 000DB         SUBL3   CBYTE, BUFFER, R0                        ;
                                 50   0200  C0  9E 000E0         MOVAB   512(R0), R0                              ;
                                 50         51  D1 000E5         CMPL    R1, R0                                   ;
                                 03         15  000E8            BLEQ    17$                                      ;
                                 51         50  D0 000EA         MOVL    R0, R1                                   ;
                           0C    AE         51  D0 000ED 17$:    MOVL    R1, BYTELIM                              ;
                   03            00    04   AC  CF 000F1         CASEL   MODE, #0, #3                             ; 2550
    002A         0020          0011   0008     000F6 18$:        .WORD   19$-18$,-                                ;
                                                                         20$-18$,-                                ;
                                                                         23$-18$,-                                ;
```

```
                                                                       24$-18$                           ;
                 66        0C    AE          00  3B 000FE 19$:   SKPC    #0, BYTELIM, (CBYTE)              ; 2553
                                             0A  13 00103        BEQL    21$                               ;
                                             0A  11 00105        BRB     22$                               ;
                 66        0C    AE      FF  8F  3B 00107 20$:   SKPC    #255, BYTELIM, (CBYTE)            ; 2555
                                             02  12 0010D        BNEQ    22$                               ;
                                             51  D4 0010F 21$:   CLRL    R1                                ;
                           52                51  D0 00111 22$:   MOVL    R1, ENDBYTE                       ;
                                             14  11 00114        BRB     26$                               ;
 0C    AE     FF   8F      6E                00  2C 00116 23$:   MOVC5   #0, (SP), #255, BYTELIM, (CBYTE)  ; 2557
                                             66     0011D                                                  ;
                                             07  11 0011E        BRB     25$                               ;
 0C    AE          00      6E                00  2C 00120 24$:   MOVC5   #0, (SP), #0, BYTELIM, (CBYTE)    ; 2559
                                             66     00126                                                  ;
                           52                53  D0 00127 25$:   MOVL    R3, ENDBYTE                       ;
                                             05  12 0012A 26$:   BNEQ    27$                               ; 2563
                 52        56        0C      AE  C1 0012C        ADDL3   BYTELIM, CBYTE, ENDBYTE           ;
                 50        56                52  C3 00131 27$:   SUBL3   ENDBYTE, CBYTE, R0                ; 2570
                           59              6940  7E 00135        MOVAQ   (COUNT)[R0], COUNT                ;
                 50        08    AE 00000200 8F  C1 00139        ADDL3   #512, BUFFER, R0                  ; 2571
                                 50          52  D1 00142        CMPL    ENDBYTE, R0                       ;
                                             52  12 00145        BNEQ    32$                               ;
                                             59  D5 00147        TSTL    COUNT                             ;
                                             4E  13 00149        BEQL    32$                               ;
                 01        02        04      AC  CF 0014B        CASEL   MODE, #2, #1                      ; 2573
                         0006              0006     00150 28$:   .WORD   29$-28$,-                         ;
                                                                         29$-28$                           ;
                                             08  11 00154        BRB     30$                               ;
                                     08      AE  DD 00156 29$:   PUSHL   BUFFER                            ; 2576
                    0000G  CF                01  FB 00159        CALLS   #1, MARK_DIRTY                    ;
                                             57  D6 0015E 30$:   INCL    BLOCK                             ; 2582
                           50        98      AA  D0 00160        MOVL    -104(BASE), R0                    ; 2583
       57     39   A0      08                00  ED 00164        CMPZV   #0, #8, 57(R0), BLOCK             ;
                                             04  1A 0016A        BGTRU   31$                               ;
                                           FEFF     0016C        BUGW                                      ; 2584
                                          0000*     0016E        .WORD   <BUG$_BADSBMBLK!4>                ;
                                     00      BE  D4 00170 31$:   CLRL    @0(SP)                            ; 2586
                                             01  DD 00173        PUSHL   #1                                ; 2587
                                             01  DD 00175        PUSHL   #1                                ;
                           50        98      AA  D0 00177        MOVL    -104(BASE), R0                    ;
                                     34    B047  9F 0017B        PUSHAB  @52(R0)[BLOCK]                    ;
                    0000G  CF                03  FB 0017F        CALLS   #3, READ_BLOCK                    ;
                       08  AE                50  D0 00184        MOVL    R0, BUFFER                        ;
                       00  BE        01      A7  9E 00188        MOVAB   1(R7), @0(SP)                     ; 2588
                       BC  AA        08      AE  D0 0018D        MOVL    BUFFER, -68(BASE)                 ; 2589
                           56        08      AE  D0 00192        MOVL    BUFFER, CBYTE                     ; 2590
                                           FF3E  31 00196        BRW     16$                               ; 2546
                                             59  D5 00199 32$:   TSTL    COUNT                             ; 2597
                                             06  13 0019B        BEQL    33$                               ;
                           56                52  D0 0019D        MOVL    ENDBYTE, CBYTE                    ; 2598
                                           FEDC  31 001A0        BRW     5$                                ; 2507
                 01        02        04      AC  CF 001A3 33$:   CASEL   MODE, #2, #1                      ; 2605
                         0006              0006     001A8 34$:   .WORD   35$-34$,-                         ;
                                                                         35$-34$                           ;
                                             08  11 001AC        BRB     36$                               ;
                                     08      AE  DD 001AE 35$:   PUSHL   BUFFER                            ; 2608
                    0000G  CF                01  FB 001B1        CALLS   #1, MARK_DIRTY                    ;
```

```
                                                                   K  7
SMALOC                                                       16-Sep-1984 01:11:44     VAX-11 Bliss-32 V4.0-742                Page  51
V04-000                                                      14-Sep-1984 12:30:47     DISK$VMSMASTER:[F11X.SRC]SMALOC.B32;1    (11)

                      14    BC        0C    AC           59   C3 001B6 36$:      SUBL3    COUNT, BITCOUNT, @LENGTHFOUND                     ; 2614
                      10    BC        08    AC      14   BC   C1 001BC           ADDL3    @LENGTHFOUND, STARTBIT, @STOPBIT                  ; 2615
                                                         50   D4 001C3           CLRL     R0                                                ; 2616
                                                         59   D5 001C5           TSTL     COUNT                                             ;
                                                         02   12 001C7           BNEQ     37$                                               ;
                                                         50   D6 001C9           INCL     R0                                                ;
                                                              04 001CB 37$:      RET                                                        ; 2618

; Routine Size:  460 bytes,    Routine Base:  $CODE$ + 09DF


; 1639          2619  1
; 1640          2620  1 END
; 1641          2621  0 ELUDOM




;                                     PSECT SUMMARY
;
;       Name                       Bytes                        Attributes
;
;  $CODE$                              2987  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                              Library Statistics
;
;                                               -------- Symbols --------       Pages       Processing
;       File                                    Total    Loaded   Percent       Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1               18619        59         0       1000          00:01.9




;                                      COMMAND QUALIFIERS

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

; Size:          2955 code + 32 data bytes
; Run Time:         02:09.1
; Elapsed Time:     04:02.8
; Lines/CPU Min:     1218
; Lexemes/CPU-Min: 54555
; Memory Used:  339 pages
; Compilation Complete
```

0173 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY