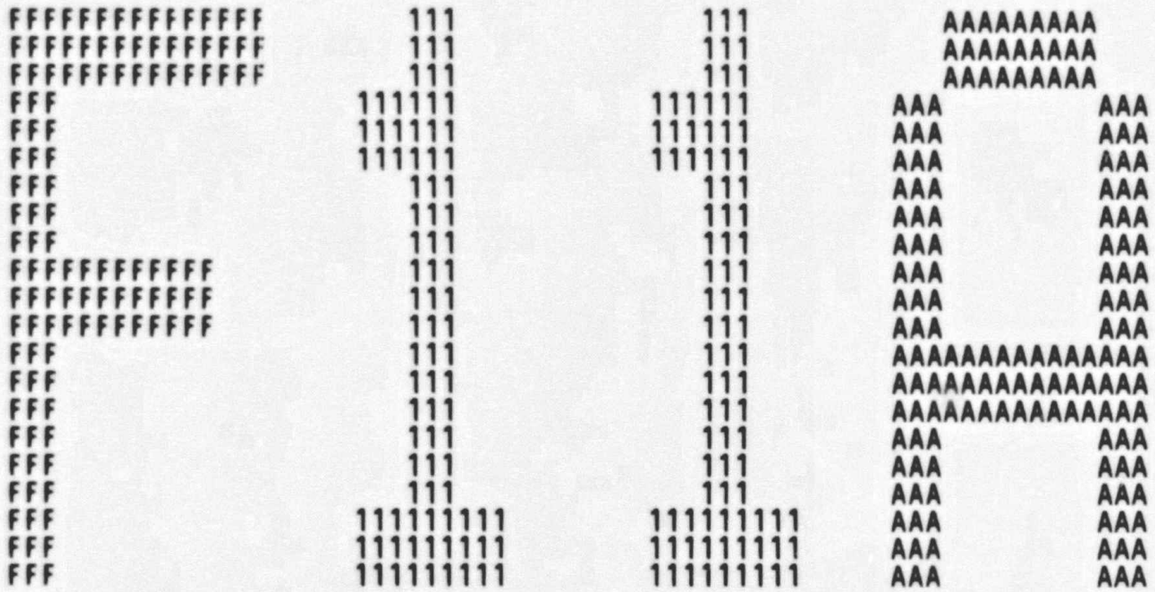
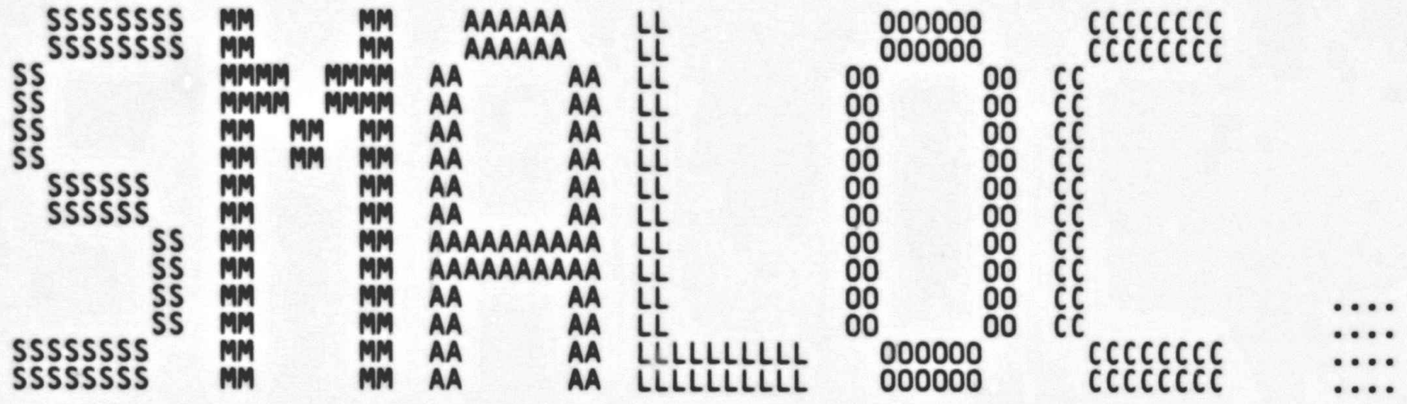

FFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
FFF
FFF
FFF
FFF
FFF
FFF
FFFFFFFFFFFF
FFFFFFFFFFFF
FFFFFFFFFFFF
FFF
FFF
FFF
FFF
FFF
FFF
FFF
FFF
111
111
111
111111
111111
111111
111
111
111
111
111
111
111
111
111
111
111
111111111
111111111
111111111
111
111
111
111111
111111
111111
111
111
111
111
111
111
111
111
111
111
111
111111111
111111111
111111111
AAAAAAAAA
AAAAAAAAA
AAAAAAAAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA

FILEID**SMALOC



```
;  1     0001  0 MODULE SMALOC (
;  2     0002  0                 LANGUAGE (BLISS32),
;  3     0003  0                 IDENT = 'V04-000'
;  4     0004  0                 ) =
;  5     0005  1 BEGIN
;  6     0006  1
;  7     0007  1 !
;  8     0008  1 !*****************************************************************************
;  9     0009  1 !*                                                                           *
; 10     0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
; 11     0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
; 12     0012  1 !*  ALL RIGHTS RESERVED.                                                    *
; 13     0013  1 !*                                                                         *
; 14     0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 15     0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 16     0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 17     0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 18     0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 19     0019  1 !*  TRANSFERRED.                                                            *
; 20     0020  1 !*                                                                         *
; 21     0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 22     0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 23     0023  1 !*  CORPORATION.                                                            *
; 24     0024  1 !*                                                                         *
; 25     0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 26     0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
; 27     0027  1 !*                                                                         *
; 28     0028  1 !*                                                                         *
; 29     0029  1 !*****************************************************************************
; 30     0030  1
; 31     0031  1 !++
; 32     0032  1 !
; 33     0033  1 ! FACILITY:  F11ACP Structure Level 1
; 34     0034  1 !
; 35     0035  1 ! ABSTRACT:
; 36     0036  1 !
; 37     0037  1 !       This module contains the routines that manipulate the volume
; 38     0038  1 !       storage bitmap. These include the routines to allocate a contiguous
; 39     0039  1 !       area, deallocate an area, and the  basic bitmap scanner.
; 40     0040  1 !
; 41     0041  1 ! ENVIRONMENT:
; 42     0042  1 !
; 43     0043  1 !       STARLET operating system, including privileged system services
; 44     0044  1 !       and internal exec routines.
; 45     0045  1 !
; 46     0046  1 !--
; 47     0047  1 !
; 48     0048  1 !
; 49     0049  1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  21-Feb-1977  18:42
; 50     0050  1 !
; 51     0051  1 ! MODIFIED BY:
; 52     0052  1 !
; 53     0053  1 !       V03-001 ACG45949        Andrew C. Goldstein,    8-Jun-1982  16:15
; 54     0054  1 !               Prevent volume free space from going negative
; 55     0055  1 !
; 56     0056  1 !       V02-003 ACG0195         Andrew C. Goldstein,    3-Mar-1981  22:54
; 57     0057  1 !               Fix 4096 block boundary problem by checking zero in BITSCAN
```

```
;    58          0058  1 !
;    59          0059  1 !       V02-002 ACG0167         Andrew C. Goldstein,    7-May-1980  18:52
;    60          0060  1 !               Previous revision history moved to F11A.REV
;    61          0061  1 !**
;    62          0062  1
;    63          0063  1
;    64          0064  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    65          0065  1 REQUIRE 'SRC$:FCPDEF.B32';
;    66          0380  1
;    67          0381  1
;    68          0382  1 !
;    69          0383  1 ! Modes of operation of the bit scanner.
;    70          0384  1 !
;    71          0385  1
;    72          0386  1 LITERAL
;    73          0387  1         FIND_SET        = 0,            ! find first one
;    74          0388  1         FIND_CLEAR      = 1,            ! find first zero
;    75          0389  1         SET_BITS        = 2,            ! set n bits
;    76          0390  1         CLEAR_BITS      = 3;            ! clear n bits
;    77          0391  1
;    78          0392  1
;    79          0393  1 FORWARD ROUTINE
;    80          0394  1         ALLOC_BLOCKS    : NOVALUE,
;    81          0395  1         RETURN_BLOCKS   : NOVALUE,
;    82          0396  1         BITSCAN,
;    83          0397  1         SET_SMVBN       : NOVALUE,
;    84          0398  1         UPDATE_FREE     : NOVALUE;
```

```
;    86          0399  1 GLOBAL ROUTINE ALLOC_BLOCKS (FIB, BLOCKS_NEEDED, START_LBN, BLOCKS_ALLOC) : NOVALUE =
;    87          0400  1
;    88          0401  1 !++
;    89          0402  1 !
;    90          0403  1 ! FUNCTIONAL DESCRIPTION:
;    91          0404  1 !
;    92          0405  1 !       This routine allocates a single contiguous area of disk.
;    93          0406  1 !       Mode of allocation is determined by the allocation control
;    94          0407  1 !       in the FIB.
;    95          0408  1 !
;    96          0409  1 ! CALLING SEQUENCE:
;    97          0410  1 !       ALLOC_BLOCKS (ARG1, ARG2, ARG3, ARG4)
;    98          0411  1 !
;    99          0412  1 ! INPUT PARAMETERS:
;   100          0413  1 !       ARG1: address of FIB for this operation
;   101          0414  1 !       ARG2: number of blocks to allocate
;   102          0415  1 !
;   103          0416  1 ! IMPLICIT INPUTS:
;   104          0417  1 !       CURRENT_VCB: ADDRESS OF VCB IN PROCESS
;   105          0418  1 !
;   106          0419  1 ! OUTPUT PARAMETERS:
;   107          0420  1 !       ARG3: address of longword to store starting LBN
;   108          0421  1 !       ARG4: address of longword to store block count
;   109          0422  1 !
;   110          0423  1 ! IMPLICIT OUTPUTS:
;   111          0424  1 !       NONE
;   112          0425  1 !
;   113          0426  1 ! ROUTINE VALUE:
;   114          0427  1 !       NONE
;   115          0428  1 !
;   116          0429  1 ! SIDE EFFECTS:
;   117          0430  1 !       storage map and VCB modified
;   118          0431  1 !
;   119          0432  1 !--
;   120          0433  1
;   121          0434  2 BEGIN
;   122          0435  2
;   123          0436  2 MAP
;   124          0437  2         FIB             : REF BBLOCK;   ! FIB of request
;   125          0438  2
;   126          0439  2 LOCAL
;   127          0440  2         BITS_NEEDED,                    ! number of map bits to allocate
;   128          0441  2         START_BLOCK,                    ! starting storage map VBN
;   129          0442  2         START_BIT,                      ! bit address in storage map
;   130          0443  2         BIT_COUNT,                      ! number of bits to scan
;   131          0444  2         FIRST_SET,                      ! start of free area
;   132          0445  2         BITS_SCANNED,                   ! number of bits processed by scanner
;   133          0446  2         END_BIT,                        ! last bit processed
;   134          0447  2         BEST_STARTBIT,                  ! start of largest free area
;   135          0448  2         BEST_BITSFOUND;                 ! size of largest free area
;   136          0449  2
;   137          0450  2 LABEL
;   138          0451  2         MAP_SCAN;                       ! code block to scan the storage map
;   139          0452  2
;   140          0453  2 EXTERNAL
;   141          0454  2         USER_STATUS     : VECTOR,       ! user I/O status block
;   142          0455  2         CURRENT_UCB     : REF BBLOCK,   ! UCB of volume
```

```
;  143          0456  2          CURRENT_VCB     : REF BBLOCK;   ! VCB of volume
;  144          0457  2
;  145          0458  2
;  146          0459  2 ! Adjust the desired block count to a bit count through the volume
;  147          0460  2 ! cluster factor. Set up the running parameters.
;  148          0461  2 !
;  149          0462  2
;  150          0463  3 BITS_NEEDED = (.BLOCKS_NEEDED + .CURRENT_VCB[VCB$W_CLUSTER] - 1)
;  151          0464  2                / .CURRENT_VCB[VCB$W_CLUSTER];
;  152          0465  2 BEST_BITSFOUND = 0;
;  153          0466  2 START_BLOCK = .CURRENT_VCB[VCB$B_SBMAPVBN];
;  154          0467  2
;  155          0468  2 ! The outer loop potentially scans the map twice: once from the given starting
;  156          0469  2 ! point through to the end and then from beginning to end, if necessary to
;  157          0470  2 ! locate a large contiguous area with a bad start.
;  158          0471  2 !
;  159          0472  2
;  160          0473  2 MAP_SCAN:
;  161          0474  3     BEGIN
;  162          0475  3     WHILE 1 DO
;  163          0476  4         BEGIN
;  164          0477  4         START_BIT = .START_BLOCK * 4096;
;  165          0478  4         BIT_COUNT = .CURRENT_UCB[UCB$L_MAXBLOCK] - .START_BIT;
;  166          0479  4
;  167          0480  4 ! Now scan the bitmap for the first free block. Having found it, scan
;  168          0481  4 ! to see how many free blocks there are there. If it is a non-contiguous
;  169          0482  4 ! allocation, accept the blocks regardless. If it is contiguous, and the
;  170          0483  4 ! free area is too small, keep looking.
;  171          0484  4 !
;  172          0485  4
;  173          0486  4         WHILE 1 DO
;  174          0487  5             BEGIN
;  175          0488  5             IF BITSCAN (FIND_SET, .START_BIT, .BIT_COUNT, FIRST_SET, BITS_SCANNED)
;  176          0489  5             THEN EXITLOOP;              ! out if end of map
;  177          0490  5
;  178          0491  5             BIT_COUNT = .BIT_COUNT - .BITS_SCANNED;
;  179          0492  5             BITSCAN (FIND_CLEAR, .FIRST_SET, MIN (.BIT_COUNT, .BITS_NEEDED),
;  180          0493  5                     START_BIT, BITS_SCANNED);
;  181          0494  5
;  182          0495  5             BIT_COUNT = .BIT_COUNT - .BITS_SCANNED;
;  183          0496  5
;  184          0497  5             IF .BITS_SCANNED GTRU .BEST_BITSFOUND
;  185          0498  5             THEN
;  186          0499  6                 BEGIN
;  187          0500  6                 BEST_STARTBIT = .FIRST_SET;
;  188          0501  6                 BEST_BITSFOUND = .BITS_SCANNED;
;  189          0502  5                 END;
;  190          0503  5
;  191          0504  5             IF .BEST_BITSFOUND GEQU .BITS_NEEDED
;  192          0505  6             OR NOT (.FIB[FIB$V_ALCON] OR .FIB[FIB$V_ALCONB])
;  193          0506  5             THEN LEAVE MAP_SCAN;        ! found what we were after
;  194          0507  5
;  195          0508  5             IF .BIT_COUNT EQL 0
;  196          0509  5             THEN EXITLOOP;              ! end of storage map
;  197          0510  5
;  198          0511  4             END;                        ! end of map scan loop
;  199          0512  4
```

```
;  200          0513  4 ! We get here when we run into the end of the storage map. If the scan
;  201          0514  4 ! started in the middle, do it once more from the top.
;  202          0515  4 !
;  203          0516  4
;  204          0517  4         IF .START_BLOCK EQL 0 THEN LEAVE MAP_SCAN;
;  205          0518  4         START_BLOCK = 0;
;  206          0519  3         END;                                    ! end of outer loop
;  207          0520  2     END;                                        ! end of block MAP_SCAN
;  208          0521  2
;  209          0522  2 ! We have either found a cluster of free blocks suitable to the occasion
;  210          0523  2 ! or we have searched the entire map. If nothing was found, or for a
;  211          0524  2 ! normal contiguous request, return error if the number of blocks is
;  212          0525  2 ! insufficient; otherwise, allocate the blocks.
;  213          0526  2 !
;  214          0527  2
;  215          0528  2 IF .BEST_BITSFOUND EQL 0
;  216          0529  3 OR (.FIB[FIB$V_ALCON] AND NOT .FIB[FIB$V_ALCONB]
;  217          0530  3     AND .BEST_BITSFOUND LSSU .BITS_NEEDED)
;  218          0531  2 THEN
;  219          0532  3     BEGIN
;  220          0533  3     USER_STATUS[1] = .BEST_BITSFOUND * .CURRENT_VCB[VCB$W_CLUSTER];
;  221          0534  3     ERR_EXIT (SS$_DEVICEFULL);
;  222          0535  2     END;
;  223          0536  2
;  224          0537  2 BITSCAN (CLEAR_BITS, .BEST_STARTBIT, .BEST_BITSFOUND, END_BIT, BITS_SCANNED);
;  225          0538  2
;  226          0539  2 KERNEL_CALL (SET_SMVBN, .END_BIT / 4096);
;  227          0540  2
;  228          0541  2 .START_LBN = .BEST_STARTBIT * .CURRENT_VCB[VCB$W_CLUSTER];
;  229          0542  2 .BLOCKS_ALLOC = .BEST_BITSFOUND * .CURRENT_VCB[VCB$W_CLUSTER];
;  230          0543  2
;  231          0544  2 KERNEL_CALL (UPDATE_FREE, - ..BLOCKS_ALLOC);
;  232          0545  2
;  233          0546  1 END;                                            ! end of routine ALLOC_BLOCKS


                                                .TITLE  SMALOC
                                                .IDENT  \V04-000\

                                                .EXTRN  USER_STATUS, CURRENT_UCB
                                                .EXTRN  CURRENT_VCB, SYS$CMKRNL

                                                .PSECT  $CODE$,NOWRT,2

                                 03FC 00000     .ENTRY  ALLOC_BLOCKS, Save R2,R3,R4,R5,R6,R7,R8,R9 ; 0399
                    59     0000G CF   9E 00002 MOVAB   CURRENT_VCB, R9                            ;
                    58     0000V CF   9E 00007 MOVAB   BITSCAN, R8                                ;
                    57 00000000G 9F   9E 0000C MOVAB   @#SYS$CMKRNL, R7                           ;
                    5E           10   C2 00013 SUBL2   #16, SP                                   ;
                    50           69   D0 00016 MOVL    CURRENT_VCB, R0                           ; 0463
                    51       3C  A0   3C 00019 MOVZWL  60(R0), R1                                ;
                    51       08  AC   C0 0001D ADDL2   BLOCKS_NEEDED, R1                         ;
                                 51   D7 00021 DECL    R1                                        ;
                    54       3C  A0   3C 00023 MOVZWL  60(R0), BITS_NEEDED                       ; 0464
                 54 51           54   C7 00027 DIVL3   BITS_NEEDED, R1, BITS_NEEDED             ;
                                 52   D4 0002B CLRL    BEST_BITSFOUND                            ; 0465
                    55       3B  A0   9A 0002D MOVZBL  59(R0), START_BLOCK                       ; 0466
```

```
                04   AE            55              0C  78 00031 1$:     ASHL    #12, START_BLOCK, START_BIT                     ; 0477
                                   50        0000G CF  D0 00036          MOVL    CURRENT_UCB, R0                                 ; 0478
                     53      00B0  C0          04  AE  C3 0003B          SUBL3   START_BIT, 176(R0), BIT_COUNT                   ;
                                               08  AE  9F 00042 2$:     PUSHAB  BITS_SCANNED                                    ; 0488
                                               04  AE  9F 00045          PUSHAB  FIRST_SET                                       ;
                                                   53  DD 00048          PUSHL   BIT_COUNT                                       ;
                                               10  AE  DD 0004A          PUSHL   START_BIT                                       ;
                                                   7E  D4 0004D          CLRL    -(SP)                                           ;
                                         68            05  FB 0004F          CALLS   #5, BITSCAN                                     ;
                                         43            50  E8 00052          BLBS    R0, 6$                                          ;
                                         53        08  AE  C2 00055          SUBL2   BITS_SCANNED, BIT_COUNT                         ; 0491
                                                   08  AE  9F 00059          PUSHAB  BITS_SCANNED                                    ; 0492
                                                   08  AE  9F 0005C          PUSHAB  START_BIT                                       ;
                                                   53  DD 0005F          PUSHL   BIT_COUNT                                       ;
                                         54            6E  D1 00061          CMPL    (SP), BITS_NEEDED                               ;
                                                       03  15 00064          BLEQ    3$                                              ;
                                         6E            54  D0 00066          MOVL    BITS_NEEDED, (SP)                               ;
                                                   0C  AE  DD 00069 3$:     PUSHL   FIRST_SET                                       ;
                                                       01  DD 0006C          PUSHL   #1                                              ;
                                         68            05  FB 0006E          CALLS   #5, BITSCAN                                     ;
                                         53        08  AE  C2 00071          SUBL2   BITS_SCANNED, BIT_COUNT                         ; 0495
                                         52        08  AE  D1 00075          CMPL    BITS_SCANNED, BEST_BITSFOUND                    ; 0497
                                                       07  1B 00079          BLEQU   4$                                              ;
                                         56            6E  D0 0007B          MOVL    FIRST_SET, BEST_STARTBIT                        ; 0500
                                         52        08  AE  D0 0007E          MOVL    BITS_SCANNED, BEST_BITSFOUND                    ; 0501
                                         54            52  D1 00082 4$:     CMPL    BEST_BITSFOUND, BITS_NEEDED                     ; 0504
                                                       19  1E 00085          BGEQU   7$                                              ;
                                               50  04  AC  D0 00087          MOVL    FIB, R0                                         ; 0505
                                               05  16  A0  E8 0008B          BLBS    22(R0), 5$                                      ;
                     0C        16  A0                 01  E1 0008F          BBC     #1, 22(R0), 7$                                  ;
                                                       53  D5 00094 5$:     TSTL    BIT_COUNT                                       ; 0508
                                                       AA  12 00096          BNEQ    2$                                              ;
                                                       55  D5 00098 6$:     TSTL    START_BLOCK                                     ; 0517
                                                       04  13 0009A          BEQL    7$                                              ;
                                                       55  D4 0009C          CLRL    START_BLOCK                                     ; 0518
                                                       91  11 0009E          BRB     1$                                              ; 0475
                                                       52  D5 000A0 7$:     TSTL    BEST_BITSFOUND                                  ; 0528
                                                       12  13 000A2          BEQL    8$                                              ;
                                               50  04  AC  D0 000A4          MOVL    FIB, R0                                         ; 0529
                                               1C  16  A0  E9 000A8          BLBC    22(R0), 9$                                      ;
                     17        16  A0                 01  E0 000AC          BBS     #1, 22(R0), 9$                                  ;
                                         54            52  D1 000B1          CMPL    BEST_BITSFOUND, BITS_NEEDED                     ; 0530
                                                       12  1E 000B4          BGEQU   9$                                              ;
                                         50            69  D0 000B6 8$:     MOVL    CURRENT_VCB, R0                                 ; 0533
                                         51        3C  A0  3C 000B9          MOVZWL  60(R0), R1                                      ;
                0000G CF                 52            51  C5 000BD          MULL3   R1, BEST_BITSFOUND, USER_STATUS+4               ;
                                                   0850  8F  BF 000C3          CHMU    #2128                                           ; 0534
                                                         04 000C7          RET                                                     ;
                                               08  AE  9F 000C8 9$:     PUSHAB  BITS_SCANNED                                    ; 0537
                                               10  AE  9F 000CB          PUSHAB  END_BIT                                         ;
                                                   52  DD 000CE          PUSHL   BEST_BITSFOUND                                  ;
                                                   56  DD 000D0          PUSHL   BEST_STARTBIT                                   ;
                                                   03  DD 000D2          PUSHL   #3                                              ;
                                         68            05  FB 000D4          CALLS   #5, BITSCAN                                     ;
                     7E        0C  AE  00001000  8F  C7 000D7          DIVL3   #4096, END_BIT, -(SP)                           ; 0539
                                                   01  DD 000E0          PUSHL   #1                                              ;
                                                   5E  DD 000E2          PUSHL   SP                                              ;
```

```
                          0000V  CF  9F  000E4        PUSHAB  SET_SMVBN                                       ;
                              67  04  FB  000E8        CALLS   #4, SYS$CMKRNL                                  ;
                              50  69  D0  000EB        MOVL    CURRENT_VCB, R0                                 ; 0541
                              51    3C  A0  3C  000EE  MOVZWL  60(R0), R1                                      ;
                      0C   BC     56  51  C5  000F2    MULL3   R1, BEST_STARTBIT, @START_LBN                   ;
                              51    3C  A0  3C  000F7  MOVZWL  60(R0), R1                                      ; 0542
                      10   BC     52  51  C5  000FB    MULL3   R1, BEST_BITSFOUND, @BLOCKS_ALLOC               ;
                              7E    10  BC  CE  00100  MNEGL   @BLOCKS_ALLOC, -(SP)                            ; 0544
                                    01  DD  00104      PUSHL   #1                                              ;
                                    5E  DD  00106      PUSHL   SP                                              ;
                          0000V  CF  9F  00108        PUSHAB  UPDATE_FREE                                     ;
                              67  04  FB  0010C        CALLS   #4, SYS$CMKRNL                                  ;
                                  04  0010F          RET                                                     ; 0546

; Routine Size:  272 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  235     0547  1 GLOBAL ROUTINE RETURN_BLOCKS (START_LBN, BLOCK_COUNT) : NOVALUE =
;  236     0548  1
;  237     0549  1 !++
;  238     0550  1 !
;  239     0551  1 ! FUNCTIONAL DESCRIPTION:
;  240     0552  1 !
;  241     0553  1 !       This routine returns a single contiguous area to the storage map.
;  242     0554  1 !
;  243     0555  1 ! CALLING SEQUENCE:
;  244     0556  1 !       RETURN_BLOCKS (ARG1, ARG2)
;  245     0557  1 !
;  246     0558  1 ! INPUT PARAMETERS:
;  247     0559  1 !       ARG1: starting LBN to free
;  248     0560  1 !       ARG2: number of blocks to free
;  249     0561  1 !
;  250     0562  1 ! IMPLICIT INPUTS:
;  251     0563  1 !       CURRENT_VCB: VCB of volume
;  252     0564  1 !       CURRENT_UCB: UCB of device
;  253     0565  1 !
;  254     0566  1 ! OUTPUT PARAMETERS:
;  255     0567  1 !       NONE
;  256     0568  1 !
;  257     0569  1 ! IMPLICIT OUTPUTS:
;  258     0570  1 !       NONE
;  259     0571  1 !
;  260     0572  1 ! ROUTINE VALUE:
;  261     0573  1 !       NONE
;  262     0574  1 !
;  263     0575  1 ! SIDE EFFECTS:
;  264     0576  1 !       storage map and VCB modified
;  265     0577  1 !
;  266     0578  1 !--
;  267     0579  1
;  268     0580  2 BEGIN
;  269     0581  2
;  270     0582  2 LOCAL
;  271     0583  2         VOLUME_SIZE,                    ! size in logical blocks of volume
;  272     0584  2         START_BIT,                      ! starting bit number in storage map
;  273     0585  2         BIT_COUNT,                      ! number of bits to set
;  274     0586  2         DUMMY1,                         ! dummies to receive return data
;  275     0587  2         DUMMY2;                         ! from BITSCAN, which is not used
;  276     0588  2
;  277     0589  2 EXTERNAL
;  278     0590  2         CURRENT_VCB     : REF BBLOCK,   ! VCB of volume in process
;  279     0591  2         CURRENT_UCB     : REF BBLOCK;   ! UCB of device unit
;  280     0592  2
;  281     0593  2
;  282     0594  2 ! First check the blocks being returned against the volume size.
;  283     0595  2 !
;  284     0596  2
;  285     0597  2 VOLUME_SIZE =    .CURRENT_UCB[UCB$B_SECTORS] *
;  286     0598  2                  .CURRENT_UCB[UCB$B_TRACKS] *
;  287     0599  2                  .CURRENT_UCB[UCB$W_CYLINDERS];
;  288     0600  2
;  289     0601  2 IF .START_LBN + .BLOCK_COUNT GTRU .VOLUME_SIZE
;  290     0602  2 THEN ERR_EXIT (SS$_BADFILEHDR);
;  291     0603  2
```

```
;  292          0604  2 ! Divide down by the volume cluster factor to convert blocks to storage
;  293          0605  2 ! map bits. If there are non-zero remainders, reject the operation on grounds
;  294          0606  2 ! of a bad file header.
;  295          0607  2 !
;  296          0608  2
;  297          0609  2 IF .START_LBN MOD .CURRENT_VCB[VCB$W_CLUSTER] NEQ 0
;  298          0610  2 THEN ERR_EXIT (SS$_BADFILEHDR);
;  299          0611  2 START_BIT = .START_LBN / .CURRENT_VCB[VCB$W_CLUSTER];
;  300          0612  2
;  301          0613  2 IF .BLOCK_COUNT MOD .CURRENT_VCB[VCB$W_CLUSTER] NEQ 0
;  302          0614  2 THEN ERR_EXIT (SS$_BADFILEHDR);
;  303          0615  2 BIT_COUNT = .BLOCK_COUNT / .CURRENT_VCB[VCB$W_CLUSTER];
;  304          0616  2
;  305          0617  2 ! Call the bit scanner to set the appropriate
;  306          0618  2 ! bits. Finally update the volume free block count.
;  307          0619  2 !
;  308          0620  2
;  309          0621  2 BITSCAN (SET_BITS, .START_BIT, .BIT_COUNT, DUMMY1, DUMMY2);
;  310          0622  2
;  311          0623  2 KERNEL_CALL (UPDATE_FREE, .BLOCK_COUNT);
;  312          0624  2
;  313          0625  1 END;                                    ! end of routine RETURN_BLOCKS



                                001C 00000          .ENTRY  RETURN_BLOCKS, Save R2,R3,R4                    ; 0547
                        54  0000G  CF  9E 00002      MOVAB   CURRENT_VCB, R4                                 ;
                        5E         08  C2 00007      SUBL2   #8, SP                                          ;
                        50  0000G  CF  D0 0000A      MOVL    CURRENT_UCB, R0                                 ; 0597
                        51    44   A0  9A 0000F      MOVZBL  68(R0), R1                                      ; 0598
                        52    45   A0  9A 00013      MOVZBL  69(R0), R2                                      ;
                        51         52  C4 00017      MULL2   R2, R1                                          ;
                        53    46   A0  3C 0001A      MOVZWL  70(R0), R3                                      ; 0599
                        51         53  C4 0001E      MULL2   R3, VOLUME_SIZE                                 ;
                50      04   AC  51    08   AC  C1 00021      ADDL3   BLOCK_COUNT, START_LBN, R0              ; 0601
                                    50  D1 00027      CMPL    R0, VOLUME_SIZE                                 ;
                                    35  1A 0002A      BGTRU   1$                                              ;
                                    50  D0 0002C      MOVL    CURRENT_VCB, R0                                 ; 0609
                                50  3C  A0  3C 0002F      MOVZWL  60(R0), R0                                  ;
7E              00      04  AC              01  7A 00033      EMUL    #1, START_LBN, #0, -(SP)                ;
50              50          8E              50  7B 00039      EDIV    R0, (SP)+, R0, R0                       ;
                                            50  D5 0003E      TSTL    R0                                      ;
                                            1F  12 00040      BNEQ    1$                                      ;
                                    50      64  D0 00042      MOVL    CURRENT_VCB, R0                         ; 0611
                                51  3C      A0  3C 00045      MOVZWL  60(R0), START_BIT                       ;
                        51      04  AC      51  C7 00049      DIVL3   START_BIT, START_LBN, START_BIT         ;
                                50  3C      A0  3C 0004E      MOVZWL  60(R0), R0                              ; 0613
7E              00      08  AC              01  7A 00052      EMUL    #1, BLOCK_COUNT, #0, -(SP)              ;
50              50          8E              50  7B 00058      EDIV    R0, (SP)+, R0, R0                       ;
                                            50  D5 0005D      TSTL    R0                                      ;
                                            05  13 0005F      BEQL    2$                                      ;
                                0810        8F  BF 00061 1$:  CHMU    #2064                                   ; 0614
                                                04 00065      RET                                             ;
                                    50      64  D0 00066 2$:  MOVL    CURRENT_VCB, R0                         ; 0615
                                50  3C      A0  3C 00069      MOVZWL  60(R0), BIT_COUNT                       ;
```

```
                50      08   AC             50  C7  0006D         DIVL3   BIT_COUNT, BLOCK_COUNT, BIT_COUNT       ;
                                            5E  DD  00072         PUSHL   SP                                      ; 0621
                                       08   AE  9F  00074         PUSHAB  DUMMY1                                  ;
                                            50  DD  00077         PUSHL   BIT_COUNT                               ;
                                            51  DD  00079         PUSHL   START_BIT                               ;
                                            02  DD  0007B         PUSHL   #2                                      ;
                          0000V CF          05  FB  0007D         CALLS   #5, BITSCAN                             ;
                                       08   AC  DD  00082         PUSHL   BLOCK_COUNT                             ; 0623
                                            01  DD  00085         PUSHL   #1                                      ;
                                            5E  DD  00087         PUSHL   SP                                      ;
                                  0000V CF      9F  00089         PUSHAB  UPDATE_FREE                             ;
                    00000000G   9F              04  FB  0008D     CALLS   #4, @#SYS$CMKRNL                        ;
                                                04  00094         RET                                             ; 0625

; Routine Size:  149 bytes,    Routine Base:  $CODE$ + 0110
```

```
;  315          0626  1 GLOBAL ROUTINE BITSCAN (MODE, STARTBIT, BITCOUNT, STOPBIT, LENGTHFOUND) =
;  316          0627  1
;  317          0628  1 !++
;  318          0629  1 !
;  319          0630  1 ! FUNCTIONAL DESCRIPTION:
;  320          0631  1 !
;  321          0632  1 !       This routine is the basic bitmap scanner. It scans the bitmap
;  322          0633  1 !       over the specified number of bits, performing the operation
;  323          0634  1 !       specified by the mode.
;  324          0635  1 !
;  325          0636  1 ! CALLING SEQUENCE:
;  326          0637  1 !       BITSCAN (ARG1, ARG2, ARG3, ARG4, ARG5)
;  327          0638  1 !
;  328          0639  1 ! INPUT PARAMETERS:
;  329          0640  1 !       ARG1: mode of operation - see module preface
;  330          0641  1 !       ARG2: starting bit address in bitmap
;  331          0642  1 !       ARG3: maximum number of bits to process
;  332          0643  1 !
;  333          0644  1 ! IMPLICIT INPUTS:
;  334          0645  1 !       CURRENT_VCB: address of VCB in process
;  335          0646  1 !
;  336          0647  1 ! OUTPUT PARAMETERS:
;  337          0648  1 !       ARG4: address of longword to receive ending bit address
;  338          0649  1 !       ARG5: address of longword to receive number of bits scanned
;  339          0650  1 !
;  340          0651  1 ! IMPLICIT OUTPUTS:
;  341          0652  1 !       NONE
;  342          0653  1 !
;  343          0654  1 ! ROUTINE VALUE:
;  344          0655  1 !       1 if maximum bit count processed
;  345          0656  1 !       0 if not
;  346          0657  1 !
;  347          0658  1 ! SIDE EFFECTS:
;  348          0659  1 !       bitmap blocks may be altered, read, and written
;  349          0660  1 !
;  350          0661  1 !--
;  351          0662  1
;  352          0663  2 BEGIN
;  353          0664  2
;  354          0665  2 LOCAL
;  355          0666  2         COUNT,                          ! number of bits to go
;  356          0667  2         BLOCK,                          ! current bitmap block number
;  357          0668  2         CBYTE,                          ! current byte offset in block
;  358          0669  2         CBIT,                           ! current bit number within byte
;  359          0670  2         BYTELIM,                        ! number of bytes to scan
;  360          0671  2         BITLIM,                         ! number of bits to scan
;  361          0672  2         BUFFER,                         ! address of bitmap buffer
;  362          0673  2         ENDBYTE,                        ! end of current byte scan
;  363          0674  2         ENDBIT;                         ! end of current bit scan
;  364          0675  2
;  365          0676  2 EXTERNAL
;  366          0677  2         BITMAP_VBN,                     ! VBN of current storage map block
;  367          0678  2         BITMAP_BUFFER   : REF BITVECTOR, ! address of current map block
;  368          0679  2         CURRENT_VCB     : REF BBLOCK;   ! VCB in process
;  369          0680  2
;  370          0681  2 EXTERNAL ROUTINE
;  371          0682  2         MARK_DIRTY,                     ! mark buffer for writeback
```

```
;  372     0683  2          READ_BLOCK;                         ! read a disk block
;  373     0684  2
;  374     0685  2
;  375     0686  2 ! Initialize by setting the count and setting up the pointers to
;  376     0687  2 ! the starting position. Read the first map block. The case of a
;  377     0688  2 ! zero count is handled specially to avoid bitmap edge problems.
;  378     0689  2 !
;  379     0690  2
;  380     0691  2 COUNT = .BITCOUNT;
;  381     0692  2 IF .COUNT EQL 0
;  382     0693  2 THEN
;  383     0694  3     BEGIN
;  384     0695  3     .LENGTHFOUND = 0;
;  385     0696  3     .STOPBIT = .STARTBIT;
;  386     0697  3     RETURN 1;
;  387     0698  2     END;
;  388     0699  2
;  389     0700  2 BLOCK = .STARTBIT<12,20>;
;  390     0701  2 IF .BLOCK GEQU .CURRENT_VCB[VCB$B_SBMAPSIZE]
;  391     0702  2 THEN BUG_CHECK (BADSBMBLK, FATAL, 'ACP tried to reference off end of bitmap');
;  392     0703  2
;  393     0704  2 IF .BLOCK+1 EQL .BITMAP_VBN
;  394     0705  2 THEN
;  395     0706  2     BUFFER = .BITMAP_BUFFER
;  396     0707  2 ELSE
;  397     0708  3     BEGIN
;  398     0709  3     BUFFER = READ_BLOCK (.BLOCK+.CURRENT_VCB[VCB$L_SBMAPLBN], 1, BITMAP_TYPE);
;  399     0710  3     BITMAP_VBN = .BLOCK+1;
;  400     0711  3     BITMAP_BUFFER = .BUFFER;
;  401     0712  2     END;
;  402     0713  2
;  403     0714  2 CBYTE = .BUFFER + .STARTBIT<3,9>;
;  404     0715  2 CBIT = .STARTBIT<0,3>;
;  405     0716  2
;  406     0717  2 ! The outer loop allows us to use the same set of bit processing instructions
;  407     0718  2 ! for the odd bits at both the start and end of the scan.
;  408     0719  2 !
;  409     0720  2
;  410     0721  2 WHILE 1 DO
;  411     0722  3     BEGIN
;  412     0723  3
;  413     0724  3 ! Process bits from the starting position up to the first byte boundary.
;  414     0725  3 !
;  415     0726  3
;  416     0727  3     BITLIM = MIN (8 - .CBIT, .COUNT);   ! max number of bits to scan
;  417     0728  3     CASE .MODE FROM 0 TO 3 OF
;  418     0729  3         SET
;  419     0730  3         [FIND_SET]:     FFS (CBIT, BITLIM, .CBYTE, ENDBIT);
;  420     0731  3
;  421     0732  3         [FIND_CLEAR]:   FFC (CBIT, BITLIM, .CBYTE, ENDBIT);
;  422     0733  3
;  423     0734  4         [SET_BITS]:     (IF NOT FFS (CBIT, BITLIM, .CBYTE, ENDBIT)
;  424     0735  5                         THEN ERR_EXIT (SS$_BADFILEHDR)
;  425     0736  4                         ELSE
;  426     0737  5                             BEGIN
;  427     0738  5                             (.CBYTE)<.CBIT, .BITLIM> = -1;
;  428     0739  5                             MARK_DIRTY (.BUFFER);
```

```
;  429     0740  3                          END);
;  430     0741  3
;  431     0742  4          [CLEAR_BITS]:   (IF NOT FFC (CBIT, BITLIM, .CBYTE, ENDBIT)
;  432     0743  5                          THEN ERR_EXIT (SS$_DEVICEFULL)
;  433     0744  4                          ELSE
;  434     0745  5                              BEGIN
;  435     0746  5                              (.CBYTE)<.CBIT, .BITLIM> = 0;
;  436     0747  5                              MARK_DIRTY (.BUFFER);
;  437     0748  3                              END);
;  438     0749  3
;  439     0750  3          TES;
;  440     0751  3
;  441     0752  3      ! Update the counters and pointers.
;  442     0753  3      !
;  443     0754  3
;  444     0755  3          COUNT = .COUNT - (.ENDBIT - .CBIT);
;  445     0756  3
;  446     0757  3      ! If we are now positioned on a byte boundary, we can process the bitmap
;  447     0758  3      ! on a byte by byte basis. Page through the bitmap until the count runs out.
;  448     0759  3      !
;  449     0760  3
;  450     0761  3          IF .COUNT EQL 0 OR .ENDBIT NEQ 8 THEN EXITLOOP;
;  451     0762  3
;  452     0763  3          CBYTE = .CBYTE + 1;
;  453     0764  3          CBIT = 0;
;  454     0765  3
;  455     0766  3          WHILE 1 DO
;  456     0767  4              BEGIN
;  457     0768  4              BYTELIM = MIN (.COUNT/8, 512 - (.CBYTE-.BUFFER));
;  458     0769  4
;  459     0770  4              CASE .MODE FROM 0 TO 3 OF
;  460     0771  4                  SET
;  461     0772  4
;  462     0773  4                  [FIND_SET]:     ENDBYTE = CH$FIND_NOT_CH (.BYTELIM, .CBYTE, 0);
;  463     0774  4
;  464     0775  4                  [FIND_CLEAR]:   ENDBYTE = CH$FIND_NOT_CH (.BYTELIM, .CBYTE, 255);
;  465     0776  4
;  466     0777  5                  [SET_BITS]:     (IF NOT CH$FAIL (ENDBYTE = CH$FIND_NOT_CH (.BYTELIM,
;  467     0778  5                                                  .CBYTE, 0))
;  468     0779  6                                  THEN ERR_EXIT (SS$_BADFILEHDR)
;  469     0780  5                                  ELSE
;  470     0781  6                                      BEGIN
;  471     0782  6                                      CH$FILL (255, .BYTELIM, .CBYTE);
;  472     0783  6                                      MARK_DIRTY (.BUFFER);
;  473     0784  4                                      END);
;  474     0785  4
;  475     0786  5                  [CLEAR_BITS]:   (IF NOT CH$FAIL (ENDBYTE = CH$FIND_NOT_CH (.BYTELIM,
;  476     0787  5                                                  .CBYTE, 255))
;  477     0788  6                                  THEN ERR_EXIT (SS$_DEVICEFULL)
;  478     0789  5                                  ELSE
;  479     0790  6                                      BEGIN
;  480     0791  6                                      CH$FILL (0, .BYTELIM, .CBYTE);
;  481     0792  6                                      MARK_DIRTY (.BUFFER);
;  482     0793  4                                      END);
;  483     0794  4
;  484     0795  4              TES;
;  485     0796  4
```

```
;  486          0797  4          IF CH$FAIL (.ENDBYTE) THEN ENDBYTE = .CBYTE + .BYTELIM;
;  487          0798  4
;  488          0799  4 ! If the count runs out or we run into an end condition leave the loop.
;  489          0800  4 ! Otherwise read the next block, wrapping around the end of the bitmap
;  490          0801  4 ! when necessary, and loop.
;  491          0802  4 !
;  492          0803  4
;  493          0804  4          COUNT = .COUNT - (.ENDBYTE - .CBYTE) * 8;
;  494          0805  4          IF .ENDBYTE - .BUFFER NEQ 512 OR .COUNT EQL 0 THEN EXITLOOP;
;  495          0806  4
;  496          0807  4          BLOCK = .BLOCK + 1;
;  497          0808  4          IF .BLOCK GEQU .CURRENT_VCB[VCB$B_SBMAPSIZE]
;  498          0809  4          THEN BUG_CHECK (BADSBMBLK, FATAL, 'ACP tried to reference off end of bitmap');
;  499          0810  4
;  500          0811  4          BUFFER = READ_BLOCK (.BLOCK+.CURRENT_VCB[VCB$L_SBMAPLBN], 1, BITMAP_TYPE);
;  501          0812  4          BITMAP_VBN = .BLOCK+1;
;  502          0813  4          BITMAP_BUFFER = .BUFFER;
;  503          0814  4          CBYTE = .BUFFER;
;  504          0815  3          END;                                    ! end of block scan loop
;  505          0816  3
;  506          0817  3 ! We have either found the desired end condition or the count will run
;  507          0818  3 ! out within the next byte. Process the final byte bit by bit.
;  508          0819  3 !
;  509          0820  3
;  510          0821  3     IF .COUNT EQL 0 THEN EXITLOOP;
;  511          0822  3     CBYTE = .ENDBYTE;
;  512          0823  2     END;                                    ! end of major loop
;  513          0824  2
;  514          0825  2 ! Scan is completed. Return the output values.
;  515          0826  2 !
;  516          0827  2
;  517          0828  2 .LENGTHFOUND = .BITCOUNT - .COUNT;
;  518          0829  2 .STOPBIT = .STARTBIT + ..LENGTHFOUND;
;  519          0830  2 RETURN .COUNT EQL 0;
;  520          0831  2
;  521          0832  1 END;                                          ! end of routine BITSCAN


                                                        .EXTRN  BITMAP_VBN, BITMAP_BUFFER
                                                        .EXTRN  MARK_DIRTY, READ_BLOCK
                                                        .EXTRN  BUG$_BADSBMBLK

                                        0FFC 00000      .ENTRY  BITSCAN, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-  ; 0626
                                                                R11                                          ;
                                    5E      0C  C2 00002        SUBL2   #12, SP                              ;
                                    59  0C  AC  D0 00005        MOVL    BITCOUNT, COUNT                      ; 0691
                                        0C  12 00009        BNEQ    1$                                       ; 0692
                                    14  BC  D4 0000B        CLRL    @LENGTHFOUND                             ; 0695
                                10  BC  08  AC  D0 0000E        MOVL    STARTBIT, @STOPBIT                   ; 0696
                                    50      01  D0 00013        MOVL    #1, R0                               ; 0697
                                            04  00016        RET                                          ;
58      09  AC          14          04  EF 00017 1$:    EXTZV   #4, #20, STARTBIT+1, BLOCK                   ; 0700
                                    50  0000G   CF  D0 0001D        MOVL    CURRENT_VCB, R0                  ; 0701
58      39  A0          08          00  ED 00022        CMPZV   #0, #8, -57(R0), BLOCK                       ;
                                        04  1A 00028        BGTRU   2$                                   ;
                                          FEFF  0002A        BUGW                                         ; 0702
```

```
                                                0000* 0002C                .WORD   <BUG$_BADSBMBLK!4>                                      ;
                                        52      01    A8  9E 0002E 2$:     MOVAB   1(R8), R2                                               ; 0704
                                0000G   CF            52  D1 00032         CMPL    R2, BITMAP_VBN                                          ;
                                                      08  12 00037         BNEQ    3$                                                      ;
                                  08    AE      0000G CF  D0 00039         MOVL    BITMAP_BUFFER, BUFFER                                   ; 0706
                                                      21  11 0003F         BRB     4$                                                      ;
                                                      01  DD 00041 3$:      PUSHL   #1                                                      ; 0709
                                                      01  DD 00043         PUSHL   #1                                                      ;
                                        50      0000G CF  D0 00045         MOVL    CURRENT_VCB, R0                                         ;
                                              34 B048 9F  0004A            PUSHAB  @52(R0)[BLOCK]                                          ;
                                0000G   CF            03  FB 0004E         CALLS   #3, READ_BLOCK                                          ;
                                  08    AE            50  D0 00053         MOVL    R0, BUFFER                                              ;
                                0000G   CF            52  D0 00057         MOVL    R2, BITMAP_VBN                                          ; 0710
                                0000G   CF      08    AE  D0 0005C         MOVL    BUFFER, BITMAP_BUFFER                                   ; 0711
        56      08    AC                09  03  EF 00062 4$:     EXTZV   #3, #9, STARTBIT, CBYTE                                 ; 0714
                                          56  08  AE C0 00068         ADDL2   BUFFER, CBYTE                                           ;
        57      08    AC                03  00  EF 0006C         EXTZV   #0, #3, STARTBIT, CBIT                                  ; 0715
                      50                08  57  C3 00072 5$:      SUBL3   CBIT, #8, R0                                            ; 0727
                                        59  50  D1 00076         CMPL    R0, COUNT                                               ;
                                            03  15 00079         BLEQ    6$                                                      ;
                                        50  59  D0 0007B         MOVL    COUNT, R0                                               ;
                                        6E  50  D0 0007E 6$:      MOVL    R0, BITLIM                                              ;
                          03            00  04  AC CF 00081         CASEL   MODE, #0, #3                                            ; 0728
                002E            0018        0010        0008        00086 7$:     .WORD   8$-7$,-                                         ;
                                                                                          9$-7$,-                                       ;
                                                                                          10$-7$,-                                      ;
                                                                                          12$-7$                                        ;
    04  AE              66        6E              57  EA 0008E 8$:      FFS     CBIT, BITLIM, (CBYTE), ENDBIT                           ; 0730
                                                  36  11 00094         BRB     15$                                                     ;
    04  AE              66        6E              57  EB 00096 9$:      FFC     CBIT, BITLIM, (CBYTE), ENDBIT                           ; 0732
                                                  2E  11 0009C         BRB     15$                                                     ;
    04  AE              66        6E              57  EA 0009E 10$:     FFS     CBIT, BITLIM, (CBYTE), ENDBIT                           ; 0734
                                                  03  13 000A4         BEQL    11$                                                     ;
                                              0083 31 000A6         BRW     27$                                                     ;
        66              6E        57  FFFFFFFF  8F  F0 000A9 11$:     INSV    #-1, CBIT, BITLIM, (CBYTE)                              ; 0738
                                                  10  11 000B2         BRB     14$                                                     ; 0739
    04  AE              66        6E              57  EB 000B4 12$:     FFC     CBIT, BITLIM, (CBYTE), ENDBIT                           ; 0742
                                                  03  13 000BA         BEQL    13$                                                     ;
                                              0089 31 000BC         BRW     31$                                                     ;
        66              6E        57              00  F0 000BF 13$:     INSV    #0, CBIT, BITLIM, (CBYTE)                               ; 0746
                                            08    AE  DD 000C4 14$:     PUSHL   BUFFER                                                  ; 0747
                                    0000G   CF      01  FB 000C7         CALLS   #1, MARK_DIRTY                                          ;
                        50                  57  04  AE C3 000CC 15$:     SUBL3   ENDBIT, CBIT, R0                                        ; 0755
                                            59  50  C0 000D1         ADDL2   R0, COUNT                                               ;
                                                03  12 000D4         BNEQ    17$                                                     ; 0761
                                            00EA 31 000D6 16$:      BRW     38$                                                     ;
                                            08  04  AE D1 000D9 17$:     CMPL    ENDBIT, #8                                              ;
                                                F7  12 000DD         BNEQ    16$                                                     ;
                                            56  D6 000DF         INCL    CBYTE                                                   ; 0763
                                            57  D4 000E1         CLRL    CBIT                                                    ; 0764
                        51                  59  08  C7 000E3 18$:     DIVL3   #8, COUNT, R1                                           ; 0768
                        50          08      AE  56  C3 000E7         SUBL3   CBYTE, BUFFER, R0                                       ;
                                            50  0200 C0 9E 000EC         MOVAB   512(R0), R0                                             ;
                                            50  51  D1 000F1         CMPL    R1, R0                                                  ;
                                                03  15 000F4         BLEQ    19$                                                     ;
                                            51  50  D0 000F6         MOVL    R0, R1                                                  ;
                                            5A  51  D0 000F9 19$:     MOVL    R1, BYTELIM                                             ;
```

```
                03          00      04  AC  CF 000FC             CASEL   MODE, #0, #3                                    ; 0770
        0039            001E    0010        0008    00101 20$:  .WORD   21$-20$,-
                                                                        22$-20$,-
                                                                        25$-20$,-
                                                                        29$-20$
                66          5A              00  3B 00109 21$:  SKPC    #0, BYTELIM, (CBYTE)                            ; 0773
                                            09  13 0010D         BEQL    23$
                                            09  11 0010F         BRB     24$
                66          5A          FF  8F  3B 00111 22$:  SKPC    #255, BYTELIM, (CBYTE)                          ; 0775
                                            02  12 00116         BNEQ    24$
                                            51  D4 00118 23$:  CLRL    R1
                            5B              51  D0 0011A 24$:  MOVL    R1, ENDBYTE
                                            3C  11 0011D         BRB     34$
                66          5A              00  3B 0011F 25$:  SKPC    #0, BYTELIM, (CBYTE)                            ; 0777
                                            02  12 00123         BNEQ    26$
                                            51  D4 00125         CLRL    R1
                            5B              51  D0 00127 26$:  MOVL    R1, ENDBYTE
                                            05  13 0012A         BEQL    28$                                             ; 0778
                                    0810    8F  BF 0012C 27$:  CHMU    #2064                                           ; 0779
                                            04  00130         RET
        5A      FF  8F          6E          00  2C 00131 28$:  MOVC5   #0, (SP), #255, BYTELIM, (CBYTE)                ; 0782
                                            66      00137
                                            19  11 00138         BRB     33$                                             ; 0783
                66          5A          FF  8F  3B 0013A 29$:  SKPC    #255, BYTELIM, (CBYTE)                          ; 0786
                                            02  12 0013F         BNEQ    30$
                                            51  D4 00141         CLRL    R1
                            5B              51  D0 00143 30$:  MOVL    R1, ENDBYTE
                                            05  13 00146         BEQL    32$                                             ; 0787
                                    0850    8F  BF 00148 31$:  CHMU    #2128                                           ; 0788
                                            04  0014C         RET
        5A          00              6E          00  2C 0014D 32$:  MOVC5   #0, (SP), #0, BYTELIM, (CBYTE)             ; 0791
                                            66      00152
                                        08  AE  DD 00153 33$:  PUSHL   BUFFER                                          ; 0792
                        0000G   CF              01  FB 00156         CALLS   #1, MARK_DIRTY
                                            5B  D5 0015B 34$:  TSTL    ENDBYTE                                         ; 0797
                                            04  12 0015D         BNEQ    35$
                5B              56              5A  C1 0015F         ADDL3   BYTELIM, CBYTE, ENDBYTE
                50              56              5B  C3 00163 35$:  SUBL3   ENDBYTE, CBYTE, R0                         ; 0804
                59                          6940  7E 00167         MOVAQ   (COUNT)[R0], COUNT
                50          08  AE  00000200   8F  C1 0016B         ADDL3   #512, BUFFER, R0                            ; 0805
                50                          5B  D1 00174         CMPL    ENDBYTE, R0
                                            40  12 00177         BNEQ    37$
                                            59  D5 00179         TSTL    COUNT
                                            3C  13 0017B         BEQL    37$
                                            58  D6 0017D         INCL    BLOCK                                           ; 0807
                            50      0000G   CF  D0 0017F         MOVL    CURRENT_VCB, R0                                 ; 0808
        58      39  A0              08          00  ED 00184         CMPZV   #0, #8, 57(R0), BLOCK
                                            04  1A 0018A         BGTRU   36$
                                              FEFF 0018C         BUGW                                                    ; 0809
                                             0000* 0018E         .WORD   <BUG$_BADSBMBLK!4>
                                            01  DD 00190 36$:  PUSHL   #1                                              ; 0811
                                            01  DD 00192         PUSHL   #1
                            50      0000G   CF  D0 00194         MOVL    CURRENT_VCB, R0
                                      34 B048   9F 00199         PUSHAB  @52(R0)[BLOCK]
                        0000G   CF              03  FB 0019D         CALLS   #3, READ_BLOCK
                          08    AE              50  D0 001A2         MOVL    R0, BUFFER
                        0000G   CF          01  A8  9E 001A6         MOVAB   1(R8), BITMAP_VBN                           ; 0812
```

```
                0000G  CF      08   AE  D0 001AC          MOVL    BUFFER, BITMAP_BUFFER                          ; 0813
                       56      08   AE  D0 001B2          MOVL    BUFFER, CBYTE                                  ; 0814
                                 FF2A  31 001B6          BRW     18$                                            ; 0766
                                    59 D5 001B9 37$:     TSTL    COUNT                                          ; 0821
                                    06 13 001BB          BEQL    38$                                            ;
                       56               5B  D0 001BD     MOVL    ENDBYTE, CBYTE                                 ; 0822
                                 FEAF  31 001C0          BRW     5$                                             ; 0721
          14   BC      0C   AC          59  C3 001C3 38$: SUBL3   COUNT, BITCOUNT, @LENGTHFOUND                  ; 0828
          10   BC      08   AC       14  BC  C1 001C9     ADDL3   @LENGTHFOUND, STARTBIT, @STOPBIT              ; 0829
                                       50  D4 001D0     CLRL    R0                                             ; 0830
                                       59  D5 001D2     TSTL    COUNT                                          ;
                                       02  12 001D4     BNEQ    39$                                            ;
                                       50  D6 001D6     INCL    R0                                             ;
                                           04 001D8 39$: RET                                                    ; 0832

; Routine Size:  473 bytes,    Routine Base:  $CODE$ + 01A5
```

SND

```
;   523          0833  1 ROUTINE SET_SMVBN (VBN) : NOVALUE =
;   524          0834  1
;   525          0835  1 !++
;   526          0836  1 !
;   527          0837  1 ! FUNCTIONAL DESCRIPTION:
;   528          0838  1 !
;   529          0839  1 !       This routine updates the current storage map VBN in the VCB.
;   530          0840  1 !       It must be called in kernel mode.
;   531          0841  1 !
;   532          0842  1 ! CALLING SEQUENCE:
;   533          0843  1 !       SET_SMVBN (ARG1)
;   534          0844  1 !
;   535          0845  1 ! INPUT PARAMETERS:
;   536          0846  1 !       ARG1: new storage map VBN
;   537          0847  1 !
;   538          0848  1 ! IMPLICIT INPUTS:
;   539          0849  1 !       NONE
;   540          0850  1 !
;   541          0851  1 ! OUTPUT PARAMETERS:
;   542          0852  1 !       NONE
;   543          0853  1 !
;   544          0854  1 ! IMPLICIT OUTPUTS:
;   545          0855  1 !       CURRENT_VCB: address of volume VCB
;   546          0856  1 !
;   547          0857  1 ! ROUTINE VALUE:
;   548          0858  1 !       NONE
;   549          0859  1 !
;   550          0860  1 ! SIDE EFFECTS:
;   551          0861  1 !       storage map VBN altered
;   552          0862  1 !
;   553          0863  1 !--
;   554          0864  1
;   555          0865  2 BEGIN
;   556          0866  2
;   557          0867  2 EXTERNAL
;   558          0868  2         CURRENT_VCB     : REF BBLOCK;   ! VCB of volume
;   559          0869  2
;   560          0870  2
;   561          0871  2 CURRENT_VCB[VCB$B_SBMAPVBN] = .VBN;
;   562          0872  2
;   563          0873  1 END;                                     ! end of routine SET_SMVBN


                                        0000 00000 SET_SMVBN:
                                                   .WORD   Save nothing                 ; 0833
                        50      0000G   CF  D0 00002       MOVL    CURRENT_VCB, R0     ; 0871
                    3B  A0        04    AC  90 00007       MOVB    VBN, 59(R0)         ;
                                        04 0000C           RET                         ; 0873

; Routine Size:  13 bytes,    Routine Base:  $CODE$ + 037E
```

```
;   565          0874  1 ROUTINE UPDATE_FREE (COUNT) : NOVALUE =
;   566          0875  1
;   567          0876  1 !++
;   568          0877  1 !
;   569          0878  1 ! FUNCTIONAL DESCRIPTION:
;   570          0879  1 !
;   571          0880  1 !       This routine updates the free block count in the volume's VCB.
;   572          0881  1 !       It must be called in kernel mode.
;   573          0882  1 !
;   574          0883  1 ! CALLING SEQUENCE:
;   575          0884  1 !       UPDATE_FREE (ARG1)
;   576          0885  1 !
;   577          0886  1 ! INPUT PARAMETERS:
;   578          0887  1 !       ARG1: value (positive or negative) to alter free count
;   579          0888  1 !
;   580          0889  1 ! IMPLICIT INPUTS:
;   581          0890  1 !       NONE
;   582          0891  1 !
;   583          0892  1 ! OUTPUT PARAMETERS:
;   584          0893  1 !       NONE
;   585          0894  1 !
;   586          0895  1 ! IMPLICIT OUTPUTS:
;   587          0896  1 !       CURRENT_VCB: VCB of volume
;   588          0897  1 !
;   589          0898  1 ! ROUTINE VALUE:
;   590          0899  1 !       NONE
;   591          0900  1 !
;   592          0901  1 ! SIDE EFFECTS:
;   593          0902  1 !       free count altered
;   594          0903  1 !
;   595          0904  1 !--
;   596          0905  1
;   597          0906  2 BEGIN
;   598          0907  2
;   599          0908  2 EXTERNAL
;   600          0909  2         CURRENT_VCB     : REF BBLOCK;   ! VCB of volume
;   601          0910  2
;   602          0911  2
;   603          0912  2 CURRENT_VCB[VCB$L_FREE] = .CURRENT_VCB[VCB$L_FREE] + .COUNT;
;   604          0913  2 IF .CURRENT_VCB[VCB$L_FREE] LSS 0
;   605          0914  2 THEN CURRENT_VCB[VCB$L_FREE] = 0;
;   606          0915  2
;   607          0916  1 END;                                     ! end of routine UPDATE_FREE


                                0000 00000 UPDATE_FREE:
                                           .WORD   Save nothing                         ; 0874
                     50   0000G  CF  D0 00002          MOVL    CURRENT_VCB, R0         ; 0912
                40   A0     04  AC  C0 00007          ADDL2   COUNT, 64(R0)           ;
                                03  18 0000C          BGEQ    1$                      ; 0913
                            40  A0  D4 0000E          CLRL    64(R0)                  ; 0914
                                    04 00011 1$:      RET                             ; 0916

; Routine Size:  18 bytes,    Routine Base:  $CODE$ + 038B
```

```
;  608          0917  1
;  609          0918  1 END
;  610          0919  0 ELUDOM
```

```
;                               PSECT SUMMARY
;
;      Name                     Bytes                          Attributes
;
;  $CODE$                         925  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        14         0        1000        00:01.9
```

```
;                                    COMMAND QUALIFIERS

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

; Size:          925 code + 0 data bytes
; Run Time:          00:19.1
; Elapsed Time:      00:39.2
; Lines/CPU Min:      2886
; Lexemes/CPU-Min:  12273
; Memory Used:   175 pages
; Compilation Complete
```

REQUEU
LIS

RWATTR
LIS

MODIFY
LIS

SCHFCB
LIS

MAKAEC
LIS

MPWIND
LIS

MAPVBN
LIS

PMS
LIS

RDHEDR
LIS

RWVB
LIS

RETDIR
LIS

RDBLOK
LIS

SMALOC
LIS

MOUNT
LIS

MAKNMB
LIS

MAKSTR
LIS

MXTHDR
LIS

FCPDEF
B32

ACLCNTRL
LIS

ACLSUBR
LIS

SNDERL
LIS

ACPCNTRL
LIS

F11X

F11BXQP
MAP

ACCESS
LIS

TRUNC
LIS

FILESERV
MAP

WITURN
LIS

SNDSMB
LIS