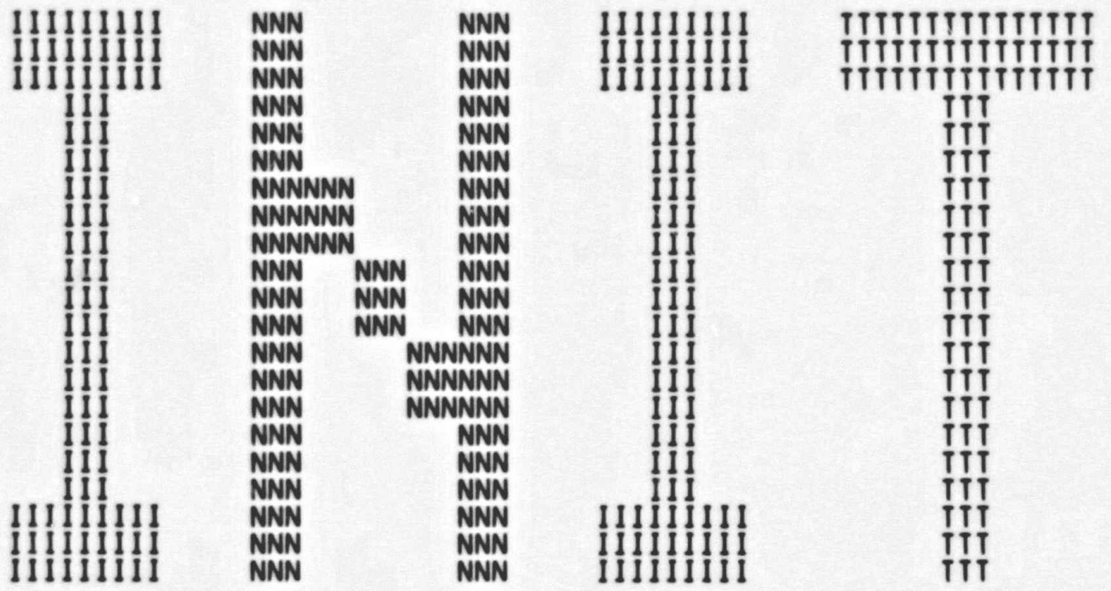
IIIIIIIII NNN NNN IIIIIIIII TTTTTTTTTTTTTTT
IIIIIIIII NNN NNN IIIIIIIII TTTTTTTTTTTTTTT
IIIIIIIII NNN NNN IIIIIIIII TTTTTTTTTTTTTTT
III NNN NNN III TTT
III NNN NNN III TTT
III NNN NNN III TTT
III NNNNNN NNN III TTT
III NNNNNN NNN III TTT
III NNNNNN NNN III TTT
III NNN NNN NNN III TTT
III NNN NNN NNN III TTT
III NNN NNNNNN III TTT
III NNN NNNNNN III TTT
III NNN NNNNNN III TTT
III NNN NNN III TTT
III NNN NNN III TTT
III NNN NNN III TTT
IIIIIIIII NNN NNN IIIIIIIII TTT
IIIIIIIII NNN NNN IIIIIIIII TTT
IIIIIIIII NNN NNN IIIIIIIII TTT

FILEID**INIBIT

```
IIIIII    NN      NN    IIIIII    BBBBBBBB     IIIIII    TTTTTTTTTT
IIIIII    NN      NN    IIIIII    BBBBBBBB     IIIIII    TTTTTTTTTT
  II      NN      NN      II      BB      BB     II          TT
  II      NN      NN      II      BB      BB     II          TT
  II      NNNN    NN      II      BB      BB     II          TT
  II      NNNN    NN      II      BB      BB     II          TT
  II      NN  NN  NN      II      BBBBBBBB       II          TT
  II      NN  NN  NN      II      BBBBBBBB       II          TT
  II      NN    NNNN      II      BB      BB     II          TT
  II      NN    NNNN      II      BB      BB     II          TT
  II      NN      NN      II      BB      BB     II          TT          ....
  II      NN      NN      II      BB      BB     II          TT          ....
IIIIII    NN      NN    IIIIII    BBBBBBBB     IIIIII        TT        ....
IIIIII    NN      NN    IIIIII    BBBBBBBB     IIIIII        TT        ....

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

```
;     1          0001  0 MODULE INIBIT (
;     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:  INIT Utility Structure Level 1
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This routine initializes the contents of the volume storage bitmap.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !
;    41          0041  1 !       STARLET operating system, including privileged system services
;    42          0042  1 !       and internal exec routines.
;    43          0043  1 !
;    44          0044  1 !--
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  13-Nov-1977  14:37
;    48          0048  1 !
;    49          0049  1 ! MODIFIED BY:
;    50          0050  1 !
;    51          0051  1 !       V02-003 LMP0002         L. Mark Pilant          11-Nov-1981  13:15
;    52          0052  1 !               Fix bitmap allocation when a large number of blocks are
;    53          0053  1 !               to be allocated.
;    54          0054  1 !
;    55          0055  1 !       V02-002 ACG0191         Andrew C. Goldstein,    1-Apr-1981  10:33
;    56          0056  1 !               Fix index file allocation at end of volume
;    57          0057  1 !
```

```
;    58          0058  1 !       V0101   ACG0069         Andrew C. Goldstein,    9-Oct-1979  16:58
;    59          0059  1 !               Remove device data table
;    60          0060  1 !
;    61          0061  1 !       V0100   ACG00001        Andrew C. Goldstein,  10-Oct-1978  21:27
;    62          0062  1 !       Previous revision history moved to [INIT.SRC]INIT.REV
;    63          0063  1 !**
;    64          0064  1
;    65          0065  1
;    66          0066  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    67          0067  1 REQUIRE 'SRC$:INIDEF.B32';
;    68          0358  1 REQUIRE 'LIBD$:[VMSLIB.OBJ]INITMSG.B32';
```

```
;    70          0490  1 GLOBAL ROUTINE INIT_BITMAP : NOVALUE =
;    71          0491  1
;    72          0492  1 !++
;    73          0493  1 !
;    74          0494  1 ! FUNCTIONAL DESCRIPTION:
;    75          0495  1 !
;    76          0496  1 !       This routine initializes the contents of the volume storage bitmap.
;    77          0497  1 !
;    78          0498  1 !
;    79          0499  1 ! CALLING SEQUENCE:
;    80          0500  1 !       INIT_BITMAP ()
;    81          0501  1 !
;    82          0502  1 ! INPUT PARAMETERS:
;    83          0503  1 !       NONE
;    84          0504  1 !
;    85          0505  1 ! IMPLICIT INPUTS:
;    86          0506  1 !       device data table
;    87          0507  1 !       allocation table
;    88          0508  1 !
;    89          0509  1 ! OUTPUT PARAMETERS:
;    90          0510  1 !       NONE
;    91          0511  1 !
;    92          0512  1 ! IMPLICIT OUTPUTS:
;    93          0513  1 !       NONE
;    94          0514  1 !
;    95          0515  1 ! ROUTINE VALUE:
;    96          0516  1 !       NONE
;    97          0517  1 !
;    98          0518  1 ! SIDE EFFECTS:
;    99          0519  1 !       storage bitmap file written
;   100          0520  1 !
;   101          0521  1 !--
;   102          0522  1
;   103          0523  2 BEGIN
;   104          0524  2
;   105          0525  2 BUILTIN
;   106          0526  2         ROT;
;   107          0527  2
;   108          0528  2 LOCAL
;   109          0529  2         BLOCK_COUNT,                    ! number of blocks in storage map
;   110          0530  2         MAP_LBN,                        ! LBN of current bitmap block
;   111          0531  2         PREV_LBN,                       ! start LBN + 1 of last entry processed
;   112          0532  2         NEXT_LBN,                       ! start LBN of current allocation table entry
;   113          0533  2         INDEX,                          ! table index of current entry
;   114          0534  2         BIT_COUNT,                      ! number of bits to clear in storage map
;   115          0535  2         MAP_VBN,                        ! relative block in storage map to use
;   116          0536  2         BIT_POS,                        ! bit position of start of area
;   117          0537  2         BIT_IDX;                        ! index into bitmap buffer
;   118          0538  2
;   119          0539  2 EXTERNAL
;   120          0540  2         INIT_OPTIONS    : BITVECTOR,    ! command option flags
;   121          0541  2         ALLOC_TABLE_CNT : VECTOR,       ! allocation block count table
;   122          0542  2         ALLOC_TABLE_LBN : VECTOR,       ! allocation LBN table
;   123          0543  2         BITMAP_CNT,                     ! block count of storage map file
;   124          0544  2         BITMAP_LBN,                     ! starting LBN of storage map file
;   125          0545  2         VOLUME_SIZE,                    ! size of volume rounded to next cluster
;   126          0546  2         CLUSTER,                        ! volume cluster factor
```

```
;  127     0547  2          BUFFER          : BBLOCK,       ! I/O buffer
;  128     0548  2          DEVICE_CHAR     : BBLOCK;       ! device characteristics
;  129     0549  2
;  130     0550  2 EXTERNAL LITERAL
;  131     0551  2          ALLOC_MAX       : UNSIGNED (16); ! total number of entries in allocation table
;  132     0552  2
;  133     0553  2 EXTERNAL ROUTINE
;  134     0554  2          CHECKSUM2,                      ! compute block checksum
;  135     0555  2          WRITE_BLOCK;                    ! write block on volume
;  136     0556  2
;  137     0557  2
;  138     0558  2 ! Build the storage control block and write it out.
;  139     0559  2 !
;  140     0560  2
;  141     0561  2 CH$FILL (0, 512, BUFFER);
;  142     0562  2
;  143     0563  2 IF .INIT_OPTIONS[OPT_STRUCTURE1]
;  144     0564  2 THEN
;  145     0565  3     BEGIN
;  146     0566  3     MAP BUFFER : VECTOR;
;  147     0567  3     BLOCK_COUNT = .BITMAP_CNT - 1;
;  148     0568  3     IF .BLOCK_COUNT GTRU 126
;  149     0569  3     THEN BLOCK_COUNT = 0;
;  150     0570  3
;  151     0571  3     (BUFFER+3)<0,8> = .BLOCK_COUNT;
;  152     0572  3     INCR J FROM 0 TO .BLOCK_COUNT - 1
;  153     0573  3     DO BUFFER[.J+1] = 4096;
;  154     0574  3     BUFFER[.BLOCK_COUNT+1] = ROT (.VOLUME_SIZE, 16);
;  155     0575  3     END
;  156     0576  3
;  157     0577  2 ELSE
;  158     0578  3     BEGIN
;  159     0579  3     BUFFER[SCB$W_STRUCLEV]      = SCB$C_LEVEL2 + 1;
;  160     0580  3     BUFFER[SCB$W_CLUSTER]       = .CLUSTER;
;  161     0581  3     BUFFER[SCB$L_VOLSIZE]       = .DEVICE_CHAR[DIB$L_MAXBLOCK];
;  162     0582  4     BUFFER[SCB$L_BLKSIZE]       = (.DEVICE_CHAR[DIB$B_SECTORS]
;  163     0583  4                                  * .DEVICE_CHAR[DIB$B_TRACKS]
;  164     0584  4                                  * .DEVICE_CHAR[DIB$W_CYLINDERS])
;  165     0585  3                                  / .DEVICE_CHAR[DIB$L_MAXBLOCK];
;  166     0586  3     BUFFER[SCB$L_SECTORS]       = .DEVICE_CHAR[DIB$B_SECTORS];
;  167     0587  3     BUFFER[SCB$L_TRACKS]        = .DEVICE_CHAR[DIB$B_TRACKS];
;  168     0588  3     BUFFER[SCB$L_CYLINDER]      = .DEVICE_CHAR[DIB$W_CYLINDERS];
;  169     0589  3
;  170     0590  3     CHECKSUM2 (BUFFER, $BYTEOFFSET (SCB$W_CHECKSUM));
;  171     0591  2     END;
;  172     0592  2
;  173     0593  2 WRITE_BLOCK (.BITMAP_LBN, BUFFER);
;  174     0594  2
;  175     0595  2 ! Now write the contents of the bitmap, marking off the areas listed in the
;  176     0596  2 ! allocation table. To save disk thrashing, we process the table entries
;  177     0597  2 ! in LBN order.
;  178     0598  2 !
;  179     0599  2
;  180     0600  2 MAP_LBN = .BITMAP_LBN + 1;
;  181     0601  2 CH$FILL (-1, 512, BUFFER);
;  182     0602  2 PREV_LBN = 0;
;  183     0603  2 WHILE 1 DO
```

```
;  184          0604  3     BEGIN
;  185          0605  3
;  186          0606  3     ! Search the allocation table for the lowest LBN which is greater than the
;  187          0607  3     ! one previously processed.
;  188          0608  3     !
;  189          0609  3
;  190          0610  3         NEXT_LBN = -1;
;  191          0611  3         INCR J FROM 0 TO ALLOC_MAX-1 DO
;  192          0612  4             BEGIN
;  193          0613  4             IF  .ALLOC_TABLE_LBN[.J] GEQU .PREV_LBN
;  194          0614  4             AND .ALLOC_TABLE_LBN[.J] LSSU .NEXT_LBN
;  195          0615  4             THEN
;  196          0616  5                 BEGIN
;  197          0617  5                 NEXT_LBN = .ALLOC_TABLE_LBN[.J];
;  198          0618  5                 INDEX = .J;
;  199          0619  4                 END;
;  200          0620  3             END;
;  201          0621  3         IF .NEXT_LBN EQL -1 THEN EXITLOOP;  ! done all entries
;  202          0622  3         PREV_LBN = .NEXT_LBN + 1;
;  203          0623  3
;  204          0624  3     ! For this group of blocks, compute the bit count and block and bit offset
;  205          0625  3     ! from the current block in the storage map.
;  206          0626  3     !
;  207          0627  3
;  208          0628  3         BIT_COUNT = .ALLOC_TABLE_CNT[.INDEX] / .CLUSTER;
;  209          0629  3         BIT_POS = .NEXT_LBN / .CLUSTER - (.MAP_LBN - .BITMAP_LBN - 1) * 4096;
;  210          0630  3
;  211          0631  3     ! Now mark off the blocks represented by the allocated entry. This is coded
;  212          0632  3     ! one bit at a time to keep the code simple; the areas are not large enough
;  213          0633  3     ! in general to warrant more intelligent code. If the bit position points
;  214          0634  3     ! off the end of the current block, pass blocks until its doesn't.
;  215          0635  3     !
;  216          0636  3
;  217          0637  3         UNTIL .BIT_POS LSSU 4096 DO
;  218          0638  4             BEGIN
;  219          0639  4             WRITE_BLOCK(.MAP_LBN, BUFFER);
;  220          0640  4             CH$FILL (-1, 512, BUFFER);
;  221          0641  4             MAP_LBN = .MAP_LBN + 1;
;  222          0642  4             BIT_POS = .BIT_POS - 4096;
;  223          0643  3             END;
;  224          0644  3
;  225          0645  3         DECR J FROM .BIT_COUNT TO 1 DO
;  226          0646  4             BEGIN
;  227          0647  4             MAP BUFFER : BITVECTOR;
;  228          0648  4
;  229          0649  4             BUFFER[.BIT_POS] = 0;
;  230          0650  4             BIT_POS = .BIT_POS + 1;
;  231          0651  4
;  232          0652  4             IF .BIT_POS GEQU 4096
;  233          0653  4             THEN
;  234          0654  5                 BEGIN
;  235          0655  5                 WRITE_BLOCK(.MAP_LBN, BUFFER);
;  236          0656  5                 CH$FILL (-1, 512, BUFFER);
;  237          0657  5                 MAP_LBN = .MAP_LBN + 1;
;  238          0658  5                 BIT_POS = 0;
;  239          0659  4                 END;
;  240          0660  3             END;
```

```
;  241          0661  3
;  242          0662  2      END;                                    ! end of allocation scan loop
;  243          0663  2
;  244          0664  2  ! Finally flush the last buffer and zero any additional blocks present
;  245          0665  2  ! in the storage map due to cluster roundup.
;  246          0666  2  !
;  247          0667  2
;  248          0668  2  WRITE_BLOCK (.MAP_LBN, BUFFER);
;  249          0669  2  MAP_LBN = .MAP_LBN + 1;
;  250          0670  2  CH$FILL (0, 512, BUFFER);
;  251          0671  2
;  252          0672  2  UNTIL .MAP_LBN GEQU .BITMAP_LBN + .BITMAP_CNT DO
;  253          0673  3      BEGIN
;  254          0674  3      WRITE_BLOCK (.MAP_LBN, BUFFER);
;  255          0675  3      MAP_LBN = .MAP_LBN + 1;
;  256          0676  3      END;
;  257          0677  2
;  258          0678  1  END;                                        ! end of routine INIT_BITMAP

                                                        .TITLE  INIBIT
                                                        .IDENT  \V04-000\

                                                        .EXTRN  INIT_OPTIONS, ALLOC_TABLE_CNT
                                                        .EXTRN  ALLOC_TABLE_LBN
                                                        .EXTRN  BITMAP_CNT, BITMAP_LBN
                                                        .EXTRN  VOLUME_SIZE, CLUSTER
                                                        .EXTRN  BUFFER, DEVICE_CHAR
                                                        .EXTRN  ALLOC_MAX, CHECKSUM2
                                                        .EXTRN  WRITE_BLOCK

                                                        .PSECT  $CODE$,NOWRT,2

                                                OFFC 00000      .ENTRY  INIT_BITMAP, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 0490
                                                                        R10,R11
                                                        5E      04  C2 00002      SUBL2   #4, SP                          ;
0200  8F                00                      6E      00  2C 00005      MOVC5   #0, (SP), #0, #512, BUFFER              ; 0561
                                                0000G   CF      0000C
                                                0000G   CF  95  0000F      TSTB    INIT_OPTIONS+3                          ; 0563
                                                            32  18 00013      BGEQ    4$                                      ;
                        50      0000G   CF                  01  C3 00015      SUBL3   #1, BITMAP_CNT, BLOCK_COUNT             ; 0567
                            0000007E    8F                  50  D1 0001B      CMPL    BLOCK_COUNT, #126                       ; 0568
                                                            02  1B 00022      BLEQU   1$                                      ;
                                                            50  D4 00024      CLRL    BLOCK_COUNT                             ; 0569
                        0000G   CF                          50  90 00026  1$:  MOVB    BLOCK_COUNT, BUFFER+3                   ; 0571
                                51                          01  CE 0002B      MNEGL   #1, J                                   ; 0572
                                                            08  11 0002E      BRB     3$                                      ;
                        0000GCF41       1000                8F  3C 00030  2$:  MOVZWL  #4096, BUFFER+4[J]                  ; 0573
                    F4          51                          50  F2 00038  3$:  AOBLSS  BLOCK_COUNT, J, 2$                  ;
                0000GCF40       0000G   CF                  10  9C 0003C      ROTL    #16, VOLUME_SIZE, BUFFER+4[BLOCK_COUNT] ; 0574
                                                            55  11 00045      BRB     5$                                      ; 0563
                        0000G   CF      0201                8F  B0 00047  4$:  MOVW    #513, BUFFER                            ; 0579
                        0000G   CF      0000G               CF  B0 0004E      MOVW    CLUSTER, BUFFER+2                       ; 0580
                        0000G   CF      0000G               CF  D0 00055      MOVL    DEVICE_CHAR+112, BUFFER+4               ; 0581
                                50      0000G               CF  9A 0005C      MOVZBL  DEVICE_CHAR+8, R0                       ; 0583
                                51      0000G               CF  9A 00061      MOVZBL  DEVICE_CHAR+9, R1                       ;
                                50                          51  C4 00066      MULL2   R1, R0                                  ;
```

```
                                        52      0000G   CF   3C 00069          MOVZWL  DEVICE_CHAR+10, R2                                  ; 0584
                                        50          52   C4 0006E          MULL2   R2, R0
                0000G   CF                      50  0000G   CF   C7 00071          DIVL3   DEVICE_CHAR+112, R0, BUFFER+8                       ; 0585
                                0000G   CF      0000G   CF   9A 00079          MOVZBL  DEVICE_CHAR+8, BUFFER+12                            ; 0586
                                0000G   CF      0000G   CF   9A 00080          MOVZBL  DEVICE_CHAR+9, BUFFER+16                            ; 0587
                                0000G   CF      0000G   CF   3C 00087          MOVZWL  DEVICE_CHAR+10, BUFFER+20                           ; 0588
                                        7E      01FE    8F   3C 0008E          MOVZWL  #510, -(SP)                                         ; 0590
                                                0000G   CF   9F 00093          PUSHAB  BUFFER
                                0000G   CF              02   FB 00097          CALLS   #2, CHECKSUM2
                                                0000G   CF   9F 0009C  5$:      PUSHAB  BUFFER                                              ; 0593
                                                0000G   CF   DD 000A0          PUSHL   BITMAP_LBN
                                0000G   CF              02   FB 000A4          CALLS   #2, WRITE_BLOCK
                        56      0000G   CF              01   C1 000A9          ADDL3   #1, BITMAP_LBN, MAP_LBN                             ; 0600
    0200    8F      FF      8F      6E              00   2C 000AF          MOVC5   #0, (SP), #-1, #512, BUFFER                         ; 0601
                                                0000G   CF      000B7
                                                        6E   D4 000BA          CLRL    PREV_LBN                                            ; 0602
                                        59              01   CE 000BC  6$:      MNEGL   #1, NEXT_LBN                                        ; 0610
                                        50              01   CE 000BF          MNEGL   #1, J                                               ; 0613
                                                        16   11 000C2          BRB     8$
                                        51      0000GCF 40   D0 000C4  7$:      MOVL    ALLOC_TABLE_LBN[J], R1
                                        6E              51   D1 000CA          CMPL    R1, PREV_LBN
                                                        0B   1F 000CD          BLSSU   8$
                                        59              51   D1 000CF          CMPL    R1, NEXT_LBN                                        ; 0614
                                                        06   1E 000D2          BGEQU   8$
                                        59              51   D0 000D4          MOVL    R1, NEXT_LBN                                        ; 0617
                                        5B              50   D0 000D7          MOVL    J, INDEX                                            ; 0618
                        E2                      50 00000000G   8F   F3 000DA  8$:      AOBLEQ  #ALLOC_MAX-1, J, 7$                         ; 0611
                            FFFFFFFF    8F                  59   D1 000E2          CMPL    NEXT_LBN, #-1                                       ; 0621
                                                            03   12 000E9          BNEQ    9$
                                                        0084    31 000EB          BRW     15$
                                        6E      01      A9   9E 000EE  9$:      MOVAB   1(R9), PREV_LBN                                     ; 0622
                        5A      0000GCF 4B      0000G   CF   C7 000F2          DIVL3   CLUSTER, ALLOC_TABLE_CNT[INDEX], BIT_COUNT           ; 0628
                        51              59      0000G   CF   C7 000FB          DIVL3   CLUSTER, NEXT_LBN, R1                               ; 0629
                        50              56      0000G   CF   C3 00101          SUBL3   BITMAP_LBN, MAP_LBN, R0
                        50              50              0C   78 00107          ASHL    #12, R0, R0
                                        51              50   C2 0010B          SUBL2   R0, R1
                                        58      1000    C1   9E 0010E          MOVAB   4096(R1), BIT_POS
                            00001000    8F              58   D1 00113  10$:      CMPL    BIT_POS, #4096                                      ; 0637
                                                        1F   1F 0011A          BLSSU   11$
                                                0000G   CF   9F 0011C          PUSHAB  BUFFER                                              ; 0639
                                                        56   DD 00120          PUSHL   MAP_LBN
                                0000G   CF              02   FB 00122          CALLS   #2, WRITE_BLOCK
    0200    8F      FF      8F      6E              00   2C 00127          MOVC5   #0, (SP), #-1, #512, BUFFER                         ; 0640
                                                0000G   CF      0012F
                                                        56   D6 00132          INCL    MAP_LBN                                             ; 0641
                                        58      F000    C8   9E 00134          MOVAB   -4096(R8), BIT_POS                                  ; 0642
                                                        D8   11 00139          BRB     10$                                                 ; 0637
                                        57      01      AA   9E 0013B  11$:      MOVAB   1(R10), J                                           ; 0645
                                                        2B   11 0013F          BRB     14$
                        00      0000G   CF              58   E5 00141  12$:      BBCC    BIT_POS, BUFFER, 13$                                ; 0649
                                                        58   D6 00147  13$:      INCL    BIT_POS                                             ; 0650
                            00001000    8F              58   D1 00149          CMPL    BIT_POS, #4096                                      ; 0652
                                                        1A   1F 00150          BLSSU   14$
                                                0000G   CF   9F 00152          PUSHAB  BUFFER                                              ; 0655
                                                        56   DD 00156          PUSHL   MAP_LBN
                                0000G   CF              02   FB 00158          CALLS   #2, WRITE_BLOCK
    0200    8F      FF      8F      6E              00   2C 0015D          MOVC5   #0, (SP), #-1, #512, BUFFER                         ; 0656
```

```
                                        0000G  CF      00165
                                               56  D6  00168          INCL    MAP_LBN                                                 ;  0657
                                               58  D4  0016A          CLRL    BIT_POS                                                 ;  0658
                                    D2         57  F5  0016C 14$:      SOBGTR  J, T2$                                                  ;  0645
                                      FF4A     31  0016F              BRW     6$                                                      ;  0603
                                        0000G  CF  9F  00172 15$:      PUSHAB  BUFFER                                                  ;  0668
                                               56  DD  00176          PUSHL   MAP_LBN                                                 ;
                                0000G  CF      02  FB  00178          CALLS   #2, WRITE_BLOCK                                         ;
                                               56  D6  0017D          INCL    MAP_LBN                                                 ;  0669
    0200    8F          00                 6E  00  2C  0017F          MOVC5   #0, (SP), #0, #512, BUFFER                              ;  0670
                                        0000G  CF      00186                                                                          ;
                    50  0000G  CF      0000G  CF  C1  00189 16$:      ADDL3   BITMAP_CNT, BITMAP_LBN, R0                              ;  0672
                                    50         56  D1  00191          CMPL    MAP_LBN, R0                                             ;
                                               0F  1E  00194          BGEQU   17$                                                     ;
                                        0000G  CF  9F  00196          PUSHAB  BUFFER                                                  ;  0674
                                               56  DD  0019A          PUSHL   MAP_LBN                                                 ;
                                0000G  CF      02  FB  0019C          CALLS   #2, WRITE_BLOCK                                         ;
                                               56  D6  001A1          INCL    MAP_LBN                                                 ;  0675
                                               E4  11  001A3          BRB     16$                                                     ;  0672
                                                   04  001A5 17$:      RET                                                             ;  0678

; Routine Size:  422 bytes,    Routine Base:  $CODE$ + 0000

;  259          0679  1
;  260          0680  1 END
;  261          0681  0 ELUDOM



;                                PSECT SUMMARY
;
;      Name                      Bytes                         Attributes
;
;  $CODE$                          422  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         18         0        1000        00:02.0


;                                COMMAND QUALIFIERS
```

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

; Size:         422 code + 0 data bytes
; Run Time:       00:14.5
; Elapsed Time:   00:29.3
; Lines/CPU Min:   2827
; Lexemes/CPU-Min: 34999
; Memory Used:  150 pages
; Compilation Complete
```