B 13

```
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   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
```

```
**FILE**ID**INIBAD
```

```
  IIIIII    NN      NN    IIIIII    BBBBBBBB      AAAAAA    DDDDDDDD
  IIIIII    NN      NN    IIIIII    BBBBBBBB      AAAAAA    DDDDDDDD
    II      NN      NN      II      BB      BB  AA      AA  DD      DD
    II      NN      NN      II      BB      BB  AA      AA  DD      DD
    II      NNNN    NN      II      BB      BB  AA      AA  DD      DD
    II      NNNN    NN      II      BB      BB  AA      AA  DD      DD
    II      NN  NN  NN      II      BBBBBBBB    AA      AA  DD      DD
    II      NN  NN  NN      II      BBBBBBBB    AA      AA  DD      DD
    II      NN    NNNN      II      BB      BB  AAAAAAAAAA  DD      DD
    II      NN    NNNN      II      BB      BB  AAAAAAAAAA  DD      DD
    II      NN      NN      II      BB      BB  AA      AA  DD      DD      ....
    II      NN      NN      II      BB      BB  AA      AA  DD      DD      ....
  IIIIII    NN      NN    IIIIII    BBBBBBBB    AA      AA  DDDDDDDD        ....
  IIIIII    NN      NN    IIIIII    BBBBBBBB    AA      AA  DDDDDDDD        ....

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

```
0 MODULE INIBAD (
0                 LANGUAGE (BLISS32),
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  INIT Utility Structure Level 1
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains the routines that do the bad block processing.
1 !
1 ! ENVIRONMENT:
1 !
1 !       STARLET operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  10-Nov-1977  19:21
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-004 ACG0361         Andrew C. Goldstein,    21-Sep-1983  16:45
1 !               Eliminate use of physical read operations
1 !
1 !       V03-003 LMP0060         L. Mark Pilant,         24-Nov-1982  14:39
1 !               Correct a problem that caused the software badblock information
1 !               to be ignored on a device whose sector size was not 512 bytes
1 !               (RL02's).
```

```
:    58          0058  1 !
:    59          0059  1 !       V03-002 LMP50629        L. Mark Pilant,         1-Nov-1982  15:52
:    60          0060  1 !               Correct a problem that caused an invalid media address error
:    61          0061  1 !               to be returned when the number of sectors in a cylinder was
:    62          0062  1 !               not a multiple of the device blocking factor.
:    63          0063  1 !
:    64          0064  1 !       V03-001 ACG0283         Andrew C. Goldstein,    8-Apr-1982  10:15
:    65          0065  1 !               Clean up use of VERIFIED options
:    66          0066  1 !
:    67          0067  1 !       V0102   ACG0075         Andrew C. Goldstein,    19-Oct-1979  17:48
:    68          0068  1 !               Add pack serial number to home block
:    69          0069  1 !
:    70          0070  1 !       V0101   ACG0069         Andrew C. Goldstein,    9-Oct-1979  16:44
:    71          0071  1 !               Remove device data table; always look for DEC-144 data
:    72          0072  1 !
:    73          0073  1 !       V0100   ACG00001        Andrew C. Goldstein, 10-Oct-1978  21:27
:    74          0074  1 !       Previous revision history moved to [INIT.SRC]INIT.REV
:    75          0075  1 !**
:    76          0076  1
:    77          0077  1
:    78          0078  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
:    79          0079  1 REQUIRE 'SRC$:INIDEF.B32';
:    80          0370  1 REQUIRE 'LIBD$:[VMSLIB.OBJ]INITMSG.B32';
:    81          0502  1
:    82          0503  1
:    83          0504  1 FORWARD ROUTINE
:    84          0505  1         INIT_BADBLOCKS  : NOVALUE,      ! main level bad block processing
:    85          0506  1         GET_FACTBAD,                    ! process factory bad block data
:    86          0507  1         GET_SOFTBAD     : NOVALUE,      ! process bad block scan program data
:    87          0508  1         GET_USERBAD     : NOVALUE,      ! process user specified data
:    88          0509  1         MARK_BAD        : NOVALUE;      ! enter bad block in allocation table
:   90      0510  1 GLOBAL ROUTINE INIT_BADBLOCKS : NOVALUE =
:   91      0511  1
:   92      0512  1 !++
:   93      0513  1 !
:   94      0514  1 ! FUNCTIONAL DESCRIPTION:
:   95      0515  1 !
:   96      0516  1 !       This is the main bad block processing routine. It calls the software
:   97      0517  1 !       data, factory data, and manually entered bad block routines as
:   98      0518  1 !       is appropriate.
:   99      0519  1 !
:  100      0520  1 !
:  101      0521  1 ! CALLING SEQUENCE:
:  102      0522  1 !       INIT_BADBLOCKS ()
:  103      0523  1 !
:  104      0524  1 ! INPUT PARAMETERS:
:  105      0525  1 !       NONE
:  106      0526  1 !
:  107      0527  1 ! IMPLICIT INPUTS:
:  108      0528  1 !       parser data base
:  109      0529  1 !       data base in INIT_DISK
:  110      0530  1 !
:  111      0531  1 ! OUTPUT PARAMETERS:
:  112      0532  1 !       NONE
:  113      0533  1 !
:  114      0534  1 ! IMPLICIT OUTPUTS:
:  115      0535  1 !       bad block area in allocation table
:  116      0536  1 !
:  117      0537  1 ! ROUTINE VALUE:
:  118      0538  1 !       NONE
:  119      0539  1 !
:  120      0540  1 ! SIDE EFFECTS:
:  121      0541  1 !       disk bad block data read
:  122      0542  1 !
:  123      0543  1 !--
:  124      0544  1
:  125      0545  2 BEGIN
:  126      0546  2
:  127      0547  2 EXTERNAL
:  128      0548  2         INIT_OPTIONS    : BITVECTOR,    ! command options
:  129      0549  2         DEVICE_CHAR     : BBLOCK,       ! device characteristics
:  130      0550  2         VOLUME_SIZE,                    ! size of volume rounded to cluster
:  131      0551  2         SMALL_DISK;                     ! maximum size of a "small" disk
:  132      0552  2
:  133      0553  2
:  134      0554  2 ! Establish whether the volume has factory bad block data or not and
:  135      0555  2 ! call the appropriate routine. Then, if user data has been entered,
:  136      0556  2 ! call the routine to process it.
:  137      0557  2 !
:  138      0558  2
:  139      0559  2 IF .INIT_OPTIONS[OPT_VERIFIED]
:  140      0560  2 THEN
:  141      0561  3     BEGIN
:  142      0562  3     IF NOT  GET_FACTBAD ()
:  143      0563  3     THEN GET_SOFTBAD ();
:  144      0564  3     END
:  145      0565  2 ELSE
:  146      0566  3     BEGIN
```

```
;   147         0567  3      IF .DEVICE_CHAR[DIB$L_MAXBLOCK] LSSU .VOLUME_SIZE
;   148         0568  3      THEN MARK_BAD (1, .DEVICE_CHAR[DIB$L_MAXBLOCK]);
;   149         0569  2      END;
;   150         0570  2
;   151         0571  2  IF .INIT_OPTIONS[OPT_BADBLOCKS]
;   152         0572  2  THEN GET_USERBAD ();
;   153         0573  2
;   154         0574  1  END;                                      ! end of routine INIT_BADBLOCKS


                                                                 .TITLE   INIBAD
                                                                 .IDENT   \V04-000\

                                                                 .EXTRN   INIT_OPTIONS, DEVICE_CHAR
                                                                 .EXTRN   VOLUME_SIZE, SMALL_DISK

                                                                 .PSECT   $CODE$,NOWRT,2

                                             0000 00000          .ENTRY   INIT_BADBLOCKS, Save nothing              ; 0510
                  0F     0000G  CF             06  E1 00002      BBC      #6, INIT_OPTIONS, 1$                      ; 0559
                         0000V  CF             00  FB 00008      CALLS    #0, GET_FACTBAD                           ; 0562
                                1B             50  E8 0000D      BLBS     R0, 2$                                    ;
                         0000V  CF             00  FB 00010      CALLS    #0, GET_SOFTBAD                           ; 0563
                                               14  11 00015      BRB      2$                                        ; 0559
                         0000G  CF     0000G   CF  D1 00017 1$:  CMPL     DEVICE_CHAR+112, VOLUME_SIZE              ; 0567
                                               0B  1E 0001E      BGEQU    2$                                        ;
                                       0000G   CF  DD 00020      PUSHL    DEVICE_CHAR+112                           ; 0568
                                               01  DD 00024      PUSHL    #1                                        ;
                         0000V  CF             02  FB 00026      CALLS    #2, MARK_BAD                              ;
                  05     0000G  CF             01  E1 0002B 2$:  BBC      #1, INIT_OPTIONS+1, 3$                    ; 0571
                         0000V  CF             00  FB 00031      CALLS    #0, GET_USERBAD                           ; 0572
                                                   04 00036 3$:  RET                                                ; 0574

; Routine Size:  55 bytes,    Routine Base:  $CODE$ + 0000
```

```
1 ROUTINE GET_FACTBAD =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine processes the factory bad block data found on the last
1 !       track of the disk.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       GET_FACTBAD ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       device table in INIT_DISK
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       allocation table in INIT_DISK
1 !       SERIAL_NUMBER: pack serial number from bad block data
1 !
1 ! ROUTINE VALUE:
1 !       LBS if factory data found
1 !       LBC if factory data not found
1 !
1 ! SIDE EFFECTS:
1 !       disk blocks read
1 !
1 !--
1
2 BEGIN
2
2 LABEL
2         SEARCH_TRACK;                   ! main loop to search last track of disk
2
2 LOCAL
2         LBN,                            ! LBN to mark bad
2         BLOCKFACT,                      ! blocking factor of disk
2         FIRST_TIME,                     ! first time through flag
2         FIRST_BUFFER,                   ! first buffer flag
2         NOGOOD,                         ! no blocks read without errors
2         STATUS,                         ! return status
2         P               : REF BBLOCK,   ! pointer into bad block descriptors
2         DATA_LBN;                       ! LBN of current block in last track
2
2 OWN
2         BUFFER2         : BBLOCK [512]; ! buffer for second copy of data
2
2 EXTERNAL
2         INIT_OPTIONS    : BITVECTOR,    ! command options
2         SERIAL_NUMBER,                  ! pack serial number
2         DEVICE_CHAR     : BBLOCK,       ! device characteristics
```

```
;  213          0632  2          BUFFER          : BBLOCK;          ! I/O buffer
;  214          0633  2
;  215          0634  2  EXTERNAL ROUTINE
;  216          0635  2          READ_BLOCK;                        ! read disk block
;  217          0636  2
;  218          0637  2
;  219          0638  2  ! First mark the entire last track of the disk bad to prevent its use.
;  220          0639  2  !
;  221          0640  2
;  222          0641  3  BLOCKFACT = (.DEVICE_CHAR[DIB$B_SECTORS]
;  223          0642  3              * .DEVICE_CHAR[DIB$B_TRACKS]
;  224          0643  3              * .DEVICE_CHAR[DIB$W_CYLINDERS])
;  225          0644  2              / .DEVICE_CHAR[DIB$L_MAXBLOCK];
;  226          0645  2
;  227          0646  2  DATA_LBN = .DEVICE_CHAR[DIB$L_MAXBLOCK] - .DEVICE_CHAR[DIB$B_SECTORS]/.BLOCKFACT;
;  228          0647  2  MARK_BAD (.DEVICE_CHAR[DIB$B_SECTORS]/.BLOCKFACT, .DATA_LBN);
;  229          0648  2
;  230          0649  2  ! Now, if automatic bad block processing is not inhibited, find a good
;  231          0650  2  ! block on the last track and process the bad block list in it. Do this
;  232          0651  2  ! twice, once on the first good block and once on the first good block
;  233          0652  2  ! after sector 10 (if not redundant) to get both factory and software
;  234          0653  2  ! detected bad block data.
;  235          0654  2  !
;  236          0655  2
;  237          0656  2  IF NOT .INIT_OPTIONS[OPT_VERIFIED] THEN RETURN 1;
;  238          0657  2
;  239          0658  2  FIRST_TIME = 1;
;  240          0659  2  NOGOOD = 1;
;  241          0660  2
;  242          0661  3  SEARCH_TRACK: BEGIN
;  243          0662  3  WHILE 1 DO
;  244          0663  4      BEGIN
;  245          0664  4
;  246          0665  4      FIRST_BUFFER = 1;
;  247          0666  4      WHILE 1 DO
;  248          0667  5          BEGIN
;  249          0668  5          STATUS = READ_BLOCK (.DATA_LBN, (IF .FIRST_BUFFER THEN BUFFER ELSE BUFFER2));
;  250          0669  5          IF .STATUS
;  251          0670  5          THEN
;  252          0671  6              BEGIN
;  253          0672  6              NOGOOD = 0;
;  254          0673  6              IF .FIRST_BUFFER
;  255          0674  6              THEN
;  256          0675  7                  BEGIN
;  257          0676  7                  IF .BUFFER[BBD$L_LASTWORD] EQL -1
;  258          0677  7                  THEN FIRST_BUFFER = 0;
;  259          0678  7                  END
;  260          0679  6              ELSE
;  261          0680  7                  BEGIN
;  262          0681  7                  IF CH$EQL (512, BUFFER, 512, BUFFER2, 0)
;  263          0682  7                  THEN EXITLOOP;
;  264          0683  6                  END;
;  265          0684  6              END
;  266          0685  5          ELSE IF .STATUS NEQ SS$_PARITY
;  267          0686  5          THEN ERR_EXIT (.STATUS);
;  268          0687  5
;  269          0688  5          DATA_LBN = .DATA_LBN + 1;
```

```
5                IF .DATA_LBN GEQ .DEVICE_CHAR[DIB$L_MAXBLOCK]
5                THEN LEAVE SEARCH_TRACK;
4                END;                             ! end of block search loop
4
4 ! We have a good bad block list. Process its entries.
4 !
4
4       IF .FIRST_TIME
4       THEN SERIAL_NUMBER = .BUFFER[BBD$L_SERIAL];
4
4       IF .BUFFER[BBD$W_FLAGS] EQL 65535
4       THEN ERR_EXIT (INIT$_DIAGPACK);
4
4       P = BUFFER + BBD$C_DESCRIPT;
4       DO
5           BEGIN
5           IF .P[BBD$V_CYLINDER] EQL 32767
5           THEN EXITLOOP;
7           LBN = ((.P[BBD$V_CYLINDER] * .DEVICE_CHAR[DIB$B_TRACKS]
6                 + .P[BBD$V_TRACK]) * .DEVICE_CHAR[DIB$B_SECTORS]
5                 + .P[BBD$V_SECTOR]) / .BLOCKFACT;
5           MARK_BAD (1, .LBN);
5           P = .P + BBD$C_ENTRY;
5           END
4       UNTIL .P GEQA BUFFER+512;
4
4 ! If we are not yet into the user data, position to it and try again.
4 !
4
4       FIRST_TIME = 0;
4       IF DATA_LBN GEQU .DEVICE_CHAR[DIB$L_MAXBLOCK]
4               - .DEVICE_CHAR[DIB$B_SECTORS]/.BLOCKFACT + 10 THEN EXITLOOP;
4       DATA_LBN = .DEVICE_CHAR[DIB$L_MAXBLOCK]
4               - .DEVICE_CHAR[DIB$B_SECTORS]/.BLOCKFACT + 10;
3       END;                                      ! end of outer loop
2 END;                                            ! end of block SEARCH_TRACK
2
2 ! If we found no good data at all, complain.
2 !
2
2 IF .NOGOOD
2 THEN ERR_EXIT (INIT$_FACTBAD);
2
2 RETURN NOT .FIRST_TIME;
2
1 END;                                            ! end of routine GET_FACTBAD


                                                                 .PSECT  $OWN$,NOEXE,2

                                                 00000 BUFFER2:.BLKB     512

                                                                 .EXTRN  SERIAL_NUMBER, BUFFER
                                                                 .EXTRN  READ_BLOCK

                                                                 .PSECT  $CODE$,NOWRT,2
```

```
                                                       0FFC 00000 GET_FACTBAD:
                                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0575
                                       50     0000G  CF  9A 00002         MOVZBL  DEVICE_CHAR+8, R0                           ; 0642
                                       51     0000G  CF  9A 00007         MOVZBL  DEVICE_CHAR+9, R1                           ;
                                       50            51  C4 0000C         MULL2   R1, R0                                      ;
                                       52     0000G  CF  3C 0000F         MOVZWL  DEVICE_CHAR+10, R2                          ; 0643
                                       50            52  C4 00014         MULL2   R2, R0                                      ;
                         57            50     0000G  CF  C7 00017         DIVL3   DEVICE_CHAR+112, R0, BLOCKFACT              ; 0644
                                       50     0000G  CF  9A 0001D         MOVZBL  DEVICE_CHAR+8, R0                           ; 0646
                                       50            57  C6 00022         DIVL2   BLOCKFACT, R0                               ;
                         7E     0000G  CF            50  C3 00025         SUBL3   R0, DEVICE_CHAR+112, DATA_LBN               ;
                                                     6E  DD 0002B         PUSHL   DATA_LBN                                    ; 0647
                                                     50  DD 0002D         PUSHL   R0                                          ;
                                0000V  CF            02  FB 0002F         CALLS   #2, MARK_BAD                                ;
                         04     0000G  CF            06  E0 00034         BBS     #6, INIT_OPTIONS, 1$                        ; 0656
                                       50            01  D0 0003A         MOVL    #1, R0                                      ;
                                                         04 0003D         RET                                                 ;
                                       55            01  D0 0003E 1$:     MOVL    #1, FIRST_TIME                              ; 0658
                                       59            01  D0 00041         MOVL    #1, NOGOOD                                  ; 0659
                                       5A            01  D0 00044 2$:     MOVL    #1, FIRST_BUFFER                            ; 0665
                                       56            6E  D0 00047         MOVL    DATA_LBN, R6                                ; 0668
                                       07            5A  E9 0004A 3$:     BLBC    FIRST_BUFFER, 4$                            ;
                                       50     0000G  CF  9E 0004D         MOVAB   BUFFER, R0                                  ;
                                                     05  11 00052         BRB     5$                                          ;
                                       50     0000'  CF  9E 00054 4$:     MOVAB   BUFFER2, R0                                 ;
                                                     50  DD 00059 5$:     PUSHL   R0                                          ;
                                                     56  DD 0005B         PUSHL   R6                                          ;
                                0000G  CF            02  FB 0005D         CALLS   #2, READ_BLOCK                              ;
                                       58            50  D0 0C062         MOVL    R0, STATUS                                  ;
                                       22            58  E9 00065         BLBC    STATUS, 7$                                  ; 0669
                                                     59  D4 00068         CLRL    NOGOOD                                      ; 0672
                                       0F            5A  E9 0006A         BLBC    FIRST_BUFFER, 6$                            ; 0673
                            FFFFFFFF   8F     0000G  CF  D1 0006D         CMPL    BUFFER+508, #-1                             ; 0676
                                                     24  12 00076         BNEQ    8$                                          ;
                                                     5A  D4 00078         CLRL    FIRST_BUFFER                                ; 0677
                                                     20  11 0007A         BRB     3$                                          ; 0673
                  0000'  CF     0000G  CF     0200   8F  29 0007C 6$:     CMPC3   #512, BUFFER, BUFFER2                       ; 0681
                                                     14  12 00086         BNEQ    8$                                          ;
                                                     21  11 00088         BRB     9$                                          ; 0682
                            000001F4   8F            58  D1 0008A 7$:     CMPL    STATUS, #500                                ; 0685
                                                     09  13 00091         BEQL    8$                                          ;
                                                     58  DD 00093         PUSHL   STATUS                                      ; 0686
                            00000000G  00            01  FB 00095         CALLS   #1, LIB$STOP                                ;
                                                     6E  D6 0009C 8$:     INCL    DATA_LBN                                    ; 0688
                                       56            6E  D0 0009E         MOVL    DATA_LBN, R6                                ; 0689
                                0000G  CF            56  D1 000A1         CMPL    R6, DEVICE_CHAR+112                         ;
                                                     A2  19 000A6         BLSS    3$                                          ;
                                                   0092  31 000A8         BRW     14$                                         ; 0690
                                       07            55  E9 000AB 9$:     BLBC    FIRST_TIME, 10$                             ; 0696
                                0000G  CF     0000G  CF  D0 000AE         MOVL    BUFFER, SERIAL_NUMBER                       ; 0697
                                FFFF   8F     0000G  CF  B1 000B5 10$:    CMPW    BUFFER+6, #65535                            ; 0699
                                                     0D  12 000BC         BNEQ    11$                                         ;
                                          007580A4   8F  DD 000BE         PUSHL   #7700644                                    ; 0700
                            00000000G  00            01  FB 000C4         CALLS   #1, LIB$STOP                                ;
                                       54     0000G  CF  9E 000CB 11$:    MOVAB   BUFFER+8, P                                 ; 0702
00007FFF   8F            64            0F            00  ED 000D0 12$:    CMPZV   #0, #15, (P), #32767                        ; 0705
```

```
                                                      3F  13 000D9         BEQL     13$                                         ;
            50            64            0F            00  EF 000DB         EXTZV    #0, #15, (P), R0                            ; 0707
                                        51     0000G  CF  9A 000E0         MOVZBL   DEVICE_CHAR+9, R1                           ;
                                        50            51  C4 000E5         MULL2    R1, R0                                      ;
            52       03   A4            07            00  EF 000E8         EXTZV    #0, #7, 3(P), R2                            ; 0708
                                        50            52  C0 000EE         ADDL2    R2, R0                                      ;
                                        51     0000G  CF  9A 000F1         MOVZBL   DEVICE_CHAR+8, R1                           ;
                                        50            51  C4 000F6         MULL2    R1, R0                                      ;
                                        52        02  A4  9A 000F9         MOVZBL   2(P), R2                                    ; 0709
                                        50            52  C0 000FD         ADDL2    R2, R0                                      ;
                          5B            50            57  C7 00100         DIVL3    BLOCKFACT, R0, LBN                          ;
                                                      5B  DD 00104         PUSHL    LBN                                         ; 0710
                                                      01  DD 00106         PUSHL    #1                                          ;
                                  0000V CF            02  FB 00108         CALLS    #2, MARK_BAD                                ;
                                        54            04  C0 0010D         ADDL2    #4, P                                       ; 0711
                                        50     0000G  CF  9E 00110         MOVAB    BUFFER+512, R0                              ; 0713
                                        50            54  D1 00115         CMPL     P, R0                                       ;
                                                      B6  1F 00118         BLSSU    12$                                         ;
                                                      55  D4 0011A 13$:    CLRL     FIRST_TIME                                  ; 0718
                                        50     0000G  CF  9A 0011C         MOVZBL   DEVICE_CHAR+8, R0                           ; 0720
                                        50            57  C6 00121         DIVL2    BLOCKFACT, R0                               ;
                                        50     0000G  CF  C2 00124         SUBL2    DEVICE_CHAR+112, R0                         ;
                                        50            0A  C2 00129         SUBL2    #10, R0                                     ;
                                        52            50  CE 0012C         MNEGL    R0, R2                                      ;
                                        51            6E  9E 0012F         MOVAB    DATA_LBN, R1                                ; 0719
                                        51            52  D1 00132         CMPL     R2, R1                                      ;
                                                      06  1B 00135         BLEQU    14$                                         ;
                                        6E            50  CE 00137         MNEGL    R0, DATA_LBN                                ; 0722
                                                    FF07  31 0013A         BRW      2$                                          ; 0662
                                        0D            59  E9 0013D 14$:    BLBC     NOGOOD, 15$                                 ; 0729
                                            007580AC  8F  DD 00140         PUSHL    #7700652                                    ; 0730
                              00000000G 00            01  FB 00146         CALLS    #1, LIB$STOP                                ;
                                        55            55  D2 0014D 15$:    MCOML    FIRST_TIME, R5                              ; 0732
                                        50            55  D0 00150         MOVL     R5, R0                                      ;
                                                          04 00153         RET                                                  ; 0734

; Routine Size:  340 bytes,    Routine Base:  $CODE$ + 0037
```

```
1 ROUTINE GET_SOFTBAD : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine processes the data left by the bad block scan program
1 !       somewhere near the end of the disk.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       GET_SOFTBAD ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       device table in INIT_DISK
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       allocation table in INIT_DISK
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       disk blocks read
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       LBN,                            ! LBN to mark bad
2       STATUS,                         ! return status
2       P               : REF BBLOCK;   ! pointer into bad block map
2
2 EXTERNAL
2       INIT_OPTIONS    : BITVECTOR,    ! command options
2       DEVICE_CHAR     : BBLOCK,       ! device characteristics
2       BUFFER          : BBLOCK;       ! I/O buffer
2
2 EXTERNAL ROUTINE
2       READ_BLOCK,                     ! read block by LBN
2       CHECKSUM2;                      ! compute block checksum
2
2
2 ! Scan from the end of the volume forward to find the bad block data.
2 ! If none is found, output a warning and proceed.
2 !
2
2 LBN = .DEVICE_CHAR[DIB$L_MAXBLOCK];
2 IF
3 BEGIN
```

```
;  374          0792 3 DECR J FROM 32 TO 1 DO
;  375          0793 4     BEGIN
;  376          0794 4     LBN = .LBN - 1;
;  377          0795 4     STATUS = READ_BLOCK (.LBN, BUFFER);
;  378          0796 4
;  379          0797 4     IF .STATUS
;  380          0798 4     THEN
;  381          0799 5         BEGIN
;  382          0800 5         IF  CHECKSUM2 (BUFFER, $BYTEOFFSET (BBM$W_CHECKSUM))
;  383          0801 5         AND .BUFFER[BBM$B_COUNTSIZE] EQL 1
;  384          0802 5         AND .BUFFER[BBM$B_LBNSIZE] EQL 3
;  385          0803 5         AND .BUFFER[BBM$B_INUSE] LEQ (512 - BBM$C_POINTERS - 2) / 2
;  386          0804 5         THEN EXITLOOP 0;
;  387          0805 5         END
;  388          0806 4     ELSE IF .STATUS NEQ SS$_PARITY
;  389          0807 4     THEN ERR_EXIT (.STATUS);
;  390          0808 4     END
;  391          0809 3 END
;  392          0810 2 THEN
;  393          0811 3     BEGIN
;  394          0812 3     ERR_MESSAGE (INIT$_NOBADDATA);
;  395          0813 3     RETURN;
;  396          0814 2     END;
;  397          0815 2
;  398          0816 2 ! Found a gooo bad block descriptor. Enter it in the bad block map and
;  399          0817 2 ! then process its contents.
;  400          0818 2 !
;  401          0819 2
;  402          0820 2 MARK_BAD (.DEVICE_CHAR[DIB$L_MAXBLOCK] - .LBN, .LBN);
;  403          0821 2
;  404          0822 2 P = BUFFER + BBM$C_POINTERS;
;  405          0823 2 DECR J FROM .BUFFER[BBM$B_INUSE]/2 TO 1 DO
;  406          0824 3     BEGIN
;  407          0825 3     LBN = .P[BBM$W_LOWLBN];
;  408          0826 3     LBN<16,8> = .P[BBM$B_HIGHLBN];
;  409          0827 3     MARK_BAD (.P[BBM$B_COUNT]+1, .LBN);
;  410          0828 3     P = .P + 4;
;  411          0829 2     END;
;  412          0830 2
;  413          0831 1 END;                                    ! end of routine GET_SOFTBAD


                                                      .EXTRN  CHECKSUM2

                                    003C 00000 GET_SOFTBAD:
                                                      .WORD   Save R2,R3,R4,R5                                 ; 0735
                    55      0000G  CF  9E 00002       MOVAB   BUFFER, R5                                       ;
                    54      0000G  CF  D0 00007       MOVL    DEVICE_CHAR+112, LBN                             ; 0789
                    52             20  D0 0000C       MOVL    #32, J                                           ; 0792
                                   55  DD 0000F 1$:   PUSHL   R5                                               ; 0795
                                   74  9F 00011       PUSHAB  -(LBN)                                           ;
              0000G  CF            02  FB 00013       CALLS   #2, READ_BLOCK                                   ;
                    53             50  D0 00018       MOVL    R0, STATUS                                       ;
                    23             53  E9 0001B       BLBC    STATUS, 2$                                       ; 0797
                    7E      01FE   8F  3C 0001E       MOVZWL  #510, -(SP)                                      ; 0800
                                   55  DD 00023       PUSHL   R5                                               ;
```

```
                                 0000G  CF            02  FB 00025         CALLS   #2, CHECKSUM2                               ;
                                        26            50  E9 0002A         BLBC    R0, 3$                                      ;
                                        01            65  91 0002D         CMPB    BUFFER, #1                                  ; 0801
                                                      21  12 00030         BNEQ    3$                                          ;
                                        03       01   A5  91 00032         CMPB    BUFFER+1, #3                                ; 0802
                                                      1B  12 00036         BNEQ    3$                                          ;
                                    FD  8F       02   A5  91 00038         CMPB    BUFFER+2, #253                              ; 0803
                                                      14  1A 0003D         BGTRU   3$                                          ;
                                                      23  11 0003F         BRB     4$                                          ; 0804
                             000001F4   8F            53  D1 00041 2$:     CMPL    STATUS, #500                                ; 0806
                                                      09  13 00048         BEQL    3$                                          ;
                                                      53  DD 0004A         PUSHL   STATUS                                      ; 0807
                             00000000G  00            01  FB 0004C         CALLS   #1, LIB$STOP                                ;
                                        B9            52  F5 00053 3$:     SOBGTR  J, 1$                                       ; 0792
                                           00759008   8F  DD 00056         PUSHL   #7704584                                    ; 0812
                             00000000G  00            01  FB 0005C         CALLS   #1, LIB$SIGNAL                              ;
                                                          04 00063         RET                                                 ; 0811
                                                      54  DD 00064 4$:     PUSHL   LBN                                         ; 0820
                         7E      0000G  CF            54  C3 00066         SUBL3   LBN, DEVICE_CHAR+112, -(SP)                 ;
                                 0000V  CF            02  FB 0006C         CALLS   #2, MARK_BAD                                ;
                                        52       04   A5  9E 00071         MOVAB   BUFFER+4, P                                 ; 0822
                                        53       02   A5  9A 00075         MOVZBL  BUFFER+2, R3                                ; 0823
                                        53            02  C6 00079         DIVL2   #2, R3                                      ;
                                                      53  D6 0007C         INCL    J                                           ;
                                                      19  11 0007E         BRB     6$                                          ;
                                        54       02   A2  3C 00080 5$:     MOVZWL  2(P), LBN                                   ; 0825
           54            08             10            62  F0 00084         INSV    (P), #16, #8, LBN                           ; 0826
                                                      54  DD 00089         PUSHL   LBN                                         ; 0827
                                        7E       01   A2  9A 0008B         MOVZBL  1(P), -(SP)                                 ;
                                                      6E  D6 0008F         INCL    (SP)                                        ;
                                 0000V  CF            02  FB 00091         CALLS   #2, MARK_BAD                                ;
                                        52            04  C0 00096         ADDL2   #4, P                                       ; 0828
                                        E4            53  F5 00099 6$:     SOBGTR  J, 5$                                       ; 0823
                                                          04 0009C         RET                                                 ; 0831

; Routine Size:  157 bytes,    Routine Base:  $CODE$ + 018B
```

```
1 ROUTINE GET_USERBAD : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine processes the bad block data entered by the user in the
1 !       command line.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       GET_USERBAD ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       device table in INIT_DISK
1 !       parser output database
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       allocation table in INIT_DISK
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       disk blocks read
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2         BLOCKFACT,                      ! blocking factor of disk
2         LBN;                            ! LBN to mark bad
2
2 EXTERNAL
2         DEVICE_CHAR     : BBLOCK,       ! device characteristics
2         BADBLOCK_TABLE  : BBLOCKVECTOR [,BAD_LENGTH],
2                                         ! user entered bad block table
2         BADBLOCK_COUNT;                 ! count of entries
2
2 ! Pick up each entry in the bad block table. If it was entered in
2 ! sector - track - cylinder form, convert it to LBN. Enter it in the
2 ! allocation table.
2 !
2
3 BLOCKFACT = (.DEVICE_CHAR[DIB$B_SECTORS]
3            * .DEVICE_CHAR[DIB$B_TRACKS]
3            * .DEVICE_CHAR[DIB$W_CYLINDERS])
2            / .DEVICE_CHAR[DIB$L_MAXBLOCK];
2
```

```
;  472         0889  2 INCR J FROM 0 TO .BADBLOCK_COUNT-1 DO
;  473         0890  3     BEGIN
;  474         0891  3     IF .BADBLOCK_TABLE[.J, BAD_STC_FORM]
;  475         0892  3     THEN
;  476         0893  5         LBN = ((.BADBLOCK_TABLE[.J, BAD_CYLINDER] * .DEVICE_CHAR[DIB$B_TRACKS]
;  477         C894  4               + .BADBLOCK_TABLE[.J, BAD_TPACK]) * .DEVICE_CHAR[DIB$B_SECTORS]
;  478         0895  3               + .BADBLOCK_TABLE[.J, BAD_SECTOR]) / .BLOCKFACT
;  479         0896  3     ELSE
;  480         0897  3         LBN = .BADBLOCK_TABLE[.J, BAD_LBN];
;  481         0898  3     MARK_BAD (.BADBLOCK_TABLE[.J, BAD_COUNT], .LBN);
;  482         0899  2     END;
;  483         0900  2
;  484         0901  1 END;                                        ! end of routine GET_USERBAD


                                                      .EXTRN  BADBLOCK_TABLE, BADBLOCK_COUNT

                                          01FC 00000 GET_USERBAD:
                                                      .WORD   Save R2,R3,R4,R5,R6,R7,R8                 ; 0832
                      58      0000G  CF   9E 00002      MOVAB   DEVICE_CHAR+8, R8                         ;
                      57      0000G  CF   9E 00007      MOVAB   BADBLOCK_TABLE, R7                        ;
                      50             68   9A 0000C      MOVZBL  DEVICE_CHAR+8, R0                         ; 0885
                      51        01   A8   9A 0000F      MOVZBL  DEVICE_CHAR+9, R1                         ;
                      50             51   C4 00013      MULL2   R1, R0                                    ;
                      52        02   A8   3C 00016      MOVZWL  DEVICE_CHAR+10, R2                        ; 0886
                      50             52   C4 0001A      MULL2   R2, R0                                    ;
          55          50        68   A8   C7 0001D      DIVL3   DEVIC_CHAR+112, R0, BLOCKFACT             ; 0887
                      54      0000G  CF   D0 00022      MOVL    BADBLOCK_COUNT, R4                        ; 0889
                      52             01   CE 00027      MNEGL   #1, J                                     ;
                                     49   11 0002A      BRB     4$                                        ;
                                06 A742   7F 0002C 1$:  PUSHAQ  BADBLOCK_TABLE+6[J]                       ; 0891
          2D          9E             00   E1 00030      BBC     #0, @(SP)+, 2$                            ;
                                02 A742   7F 00034      PUSHAQ  BADBLOCK_TABLE+2[J]                       ; 0893
                      50             9E   3C 00038      MOVZWL  @(SP)+, R0                                ;
                      51        01   A8   9A 0003B      MOVZBL  DEVICE_CHAR+9, R1                         ;
                      50             51   C4 0003F      MULL2   R1, R0                                    ;
                                01 A742   7F 00042      PUSHAQ  BADBLOCK_TABLE+1[J]                       ; 0894
                      56             9E   9A 00046      MOVZBL  @(SP)+, R6                                ;
                      50             56   C0 00049      ADDL2   R6, R0                                    ;
                      51             68   9A 0004C      MOVZBL  DEVICE_CHAR+8, R1                         ;
                      50             51   C4 0004F      MULL2   R1, R0                                    ;
                                   6742   7F 00052      PUSHAQ  BADBLOCK_TABLE[J]                         ; 0895
                      56             9E   9A 00055      MOVZBL  @(SP)+, R6                                ;
                      50             56   C0 00058      ADDL2   R6, R0                                    ;
          53          50             55   C7 0005B      DIVL3   BLOCKFACT, R0, LBN                        ;
                                     06   11 0005F      BRB     3$                                        ; 0893
                                   6742   7F 00061 2$:  PUSHAQ  BADBLOCK_TABLE[J]                         ; 0897
                      53             9E   D0 00064      MOVL    @(SP)+, LBN                               ;
                                     53   DD 00067 3$:  PUSHL   LBN                                       ; 0898
                                04 A742   7F 00069      PUSHAQ  BADBLOCK_TABLE+4[J]                       ;
                      7E             9E   3C 0006D      MOVZWL  @(SP)+, -(SP)                             ;
               0000V  CF             02   FB 00070      CALLS   #2, MARK_BAD                              ;
          B3          52             54   F2 00075 4$:  AOBLSS  R4, J, 1$                                 ; 0889
                                          04 00079      RET                                               ; 0901

; Routine Size:  122 bytes,    Routine Base:  $CODE$ + 0228
```

```
1 ROUTINE MARK_BAD (BLOCK_COUNT, START_LBN) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine enters the indicated block(s) into the bad block part
1 !       of the allocation table. The table is maintained in reverse order
1 !       by LBN, and adjacent or overlapping areas are merged. Reverse order
1 !       is used to make the bad block data appear at the front of the volume's
1 !       bad block file.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       MARK_BAD (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: count of blocks to mark bad
1 !       ARG2: start LBN of blocks
1 !
1 ! IMPLICIT INPUTS:
1 !       allocation table
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       allocation table altered
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       LBN,                            ! start LBN of new bad cluster
2       COUNT,                          ! block count of new bad cluster
2       J,                              ! index into bad block allocation table
2       C;                              ! merge loop counter
2
2 EXTERNAL
2       CLUSTER,                        ! volume cluster factor
2       VOLUME_SIZE,                    ! volume size rounded tc next cluster
2       BADBLOCK_TOTAL,                 ! count of bad areas so far
2       BADBLOCK_LBN    : VECTOR,       ! bad block LBN table
2       BADBLOCK_CNT    : VECTOR;       ! bad block count table
2
2 EXTERNAL LITERAL
2       BADBLOCK_MAX    : UNSIGNED (16); ! length of bad block table
2
2
2 ! Round the start LBN and count out to the cluster boundaries surrounding
```

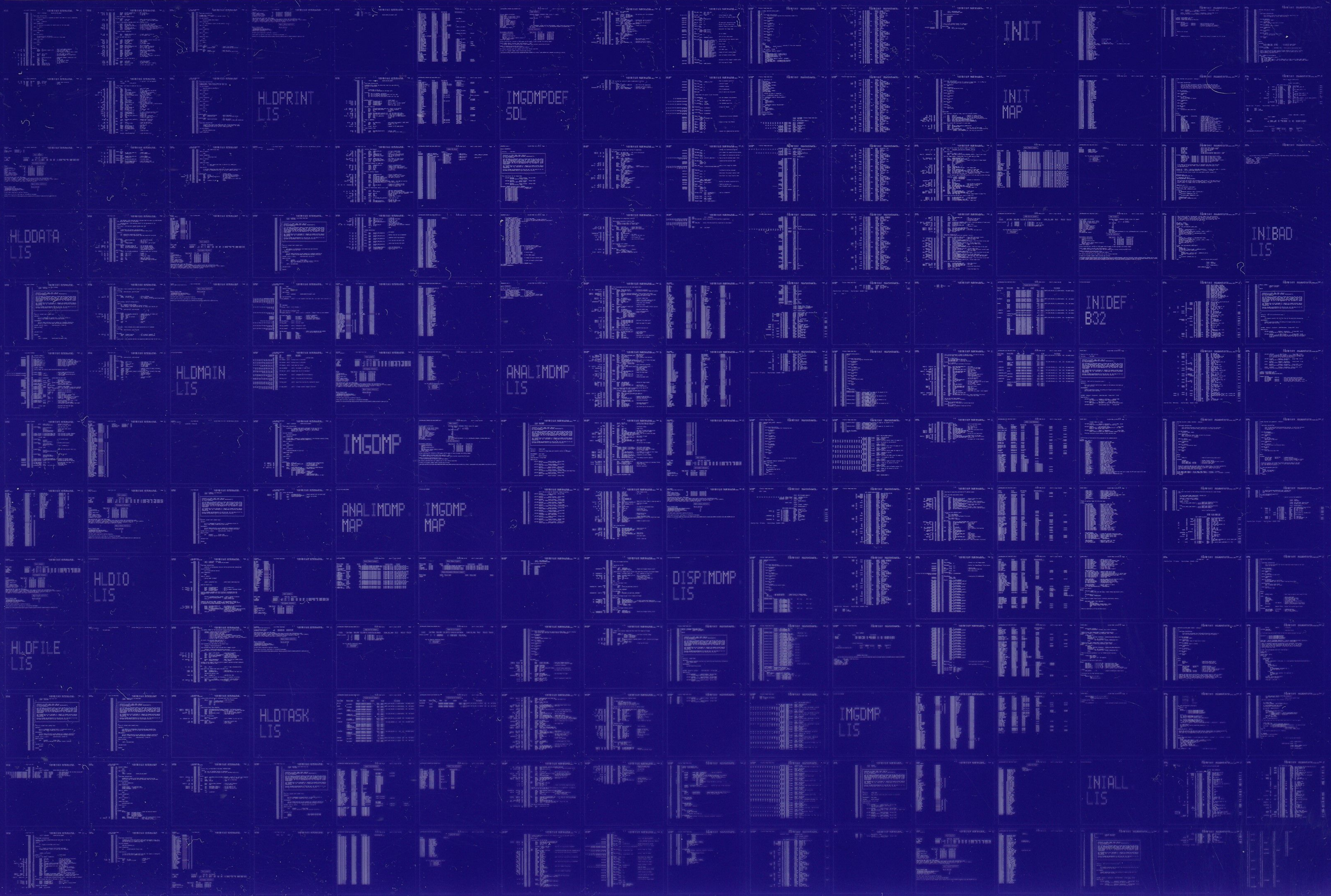
```
2 ! the bad area.
2 !
2
2 IF .BADBLOCK_TOTAL GEQ BADBLOCK_MAX
2 THEN ERR_EXIT (INIT$_MAXBAD);
2
2 LBN = .START_LBN / .CLUSTER * .CLUSTER;
2 COUNT = (.START_LBN + .BLOCK_COUNT + .CLUSTER - 1) / .CLUSTER * .CLUSTER - .LBN;
2
2 IF .LBN GEQU .VOLUME_SIZE
2 THEN ERR_EXIT (INIT$_BADRANGE);
2
2 ! Search the allocation table until an entry is found with a start LBN lower
2 ! than the new LBN. Shuffle the table down at this point and insert the
2 ! new entry.
2 !
2
2 J = 0;
2 UNTIL .J GEQ .BADBLOCK_TOTAL DO
3     BEGIN
3     IF .BADBLOCK_LBN[.J] LSSU .LBN THEN EXITLOOP;
3     J = .J + 1;
2     END;
2
2 CH$MOVE ((.BADBLOCK_TOTAL-.J)*4, BADBLOCK_LBN[.J], BADBLOCK_LBN[.J+1]);
2 CH$MOVE ((.BADBLOCK_TOTAL-.J)*4, BADBLOCK_CNT[.J], BADBLOCK_CNT[.J+1]);
2 BADBLOCK_TOTAL = .BADBLOCK_TOTAL + 1;
2 BADBLOCK_CNT[.J] = .COUNT;
2 BADBLOCK_LBN[.J] = .LBN;
2
2 ! Now check for adjacencies and merge if they exist. Start with the previous
2 ! table entry and compare pairs.
2 !
2
2 IF .J NEQ 0 THEN J = .J-1;
2 C = 0;
2
2 UNTIL .J+1 GEQ .BADBLOCK_TOTAL DO
3     BEGIN
3     IF .BADBLOCK_LBN[.J] LEQ .BADBLOCK_LBN[.J+1] + .BADBLOCK_CNT[.J+1]
3     THEN
4         BEGIN
4         BADBLOCK_CNT[.J+1] = MAXU (.BADBLOCK_LBN[.J] + .BADBLOCK_CNT[.J],
4                                    .BADBLOCK_LBN[.J+1] + .BADBLOCK_CNT[.J+1])
4                              - .BADBLOCK_LBN[.J+1];
4         BADBLOCK_TOTAL = .BADBLOCK_TOTAL - 1;
4         CH$MOVE ((.BADBLOCK_TOTAL-.J)*4, BADBLOCK_LBN[.J+1], BADBLOCK_LBN[.J]);
4         CH$MOVE ((.BADBLOCK_TOTAL-.J)*4, BADBLOCK_CNT[.J+1], BADBLOCK_CNT[.J]);
4         BADBLOCK_CNT[.BADBLOCK_TOTAL] = 0;
4         END
4
3     ELSE
4         BEGIN
4         J = .J + 1;
4         C = .C + 1;
4         IF .C GEQ 2 THEN EXITLOOP
3         END;
```

```
;  600         1016  2     END;                                   ! end of merge loop
;  601         1017  2
;  602         1018  1 END;                                       ! end of routine MARK_BAD



                                                       .EXTRN  CLUSTER, BADBLOCK_TOTAL
                                                       .EXTRN  BADBLOCK_LBN, BADBLOCK_CNT
                                                       .EXTRN  BADBLOCK_MAX

                                    0FFC 00000 MARK_BAD:
                                                       .WORD   Save P2,R3,R4,R5,R6,R7,R8,R9,R10,R11     ; 0902
                  5B      0000G    CF 9E 00002         MOVAB   BADBLuCK_LBN, R11                        ;
      00000000G   8F      0000G    CF D1 00007         CMPL    BADBLOCK_TOTAL, #BADBLOCK_MAX            ; 0962
                                   0D 19 00010         BLSS    1$                                       ;
                       007580BC    8F DD 00012         PUSHL   #7700668                                 ; 0963
      00000000G   00               01 FB 00018         CALLS   #1, LIB$STOP                             ;
                  51      0000G    CF D0 0001F 1$:     MOVL    CLUSTER, R1                              ; 0965
  50         08   AC               51 C7 00024         DIVL3   R1, START_LBN, R0                        ;
  58              50               51 C5 00029         MULL3   R1, R0, LBN                              ;
  50         08   AC         04    AC C1 0002D         ADDL3   BLOCK_COUNT, START_LBN, R0               ; 0966
                  50         FF  A140 9E 00033         MOVAB   -1(R1)[R0], R0                           ;
                  50               51 C6 00038         DIVL2   R1, R0                                   ;
                  50               51 C4 0003B         MULL2   R1, R0                                   ;
  59              50               58 C3 0003E         SUBL3   LBN, R0, COUNT                           ;
          0000G   CF               58 D1 00042         CMPL    LBN, VOLUME_SIZE                         ; 0968
                                   0D 1F 00047         BLSSU   2$                                       ;
                       00758094    8F DD 00049         PUSHL   #7700628                                 ; 0969
      00000000G   00               01 FB 0004F         CALLS   #1, LIB$STOP                             ;
                                   56 D4 00056 2$:     CLRL    J                                        ; 0976
          0000G   CF               56 D1 00058 3$:     CMPL    J, BADBLOCK_TOTAL                        ; 0977
                                   0A 18 0005D         BGEQ    4$                                       ;
                  58             6B46 D1 0005F         CMPL    BADBLOCK_LBN[J], LBN                     ; 0979
                                   04 1F 00063         BLSSU   4$                                       ;
                                   56 D6 00065         INCL    J                                        ; 0980
                                   EF 11 00067         BRB     3$                                       ; 0977
  57      0000G   CF               56 C3 00069 4$:     SUBL3   J, BADBLOCK_TOTAL, R7                    ; 0983
                  57               04 C4 0006F         MULL2   #4, R7                                   ;
                             04  AB46 DF 00072         PUSHAL  BADBLOCK_LBN+4[J]                        ;
                                 6B46 DF 00076         PUSHAL  BADBLOCK_LBN[J]                          ;
  9E              9E               57 28 00079         MOVC3   R7, @(SP)+, @(SP)+                       ;
                            0000GCF46 DF 0007D         PUSHAL  BADBLOCK_CNT+4[J]                        ; 0984
                            0000GCF46 DF 00082         PUSHAL  BADBLOCK_CNT[J]                          ;
  9E              9E               57 28 00087         MOVC3   R7, @(SP)+, @(SP)+                       ;
                          0000G    CF D6 0008B         INCL    BADBLOCK_TOTAL                           ; 0985
          0000GCF46                59 D0 0008F         MOVL    COUNT, BADBLOCK_CNT[J]                   ; 0986
                6B46               58 D0 00095         MOVL    LBN, BADBLOCK_LBN[J]                     ; 0987
                                   56 D5 00099         TSTL    J                                        ; 0993
                                   02 13 0009B         BEQL    5$                                       ;
                                   56 D7 0009D         DECL    J                                        ;
                                   5A D4 0009F 5$:     CLRL    C                                        ; 0994
                  50         01    A6 9E 000A1 6$:     MOVAB   1(R6), R0                                ; 0996
          0000G   CF               50 D1 000A5         CMPL    R0, BADBLOCK_TOTAL                       ;
                                   59 18 000AA         BGEQ    9$                                       ;
                  59        0000GCF40 DE 000AC         MOVAL   BADBLOCK_CNT[R0], R9                     ; 0998
  52            6B40               69 C1 000B2         ADDL3   (R9), BADBLOCK_LBN[R0], R2               ;
                  52             6B46 D1 000B7         CMPL    BADBLOCK_LBN[J], R2                      ;
```

```
                                            3F  14 000BB          BGTR    8$                                          ;
                51         6B46     0000GCF46  C1 000BD          ADDL3   BADBLOCK_CNT[J], BADBLOCK_LBN[J], R1        ; 1001
                             52            51  D1 000C5          CMPL    R1, R2                                      ; 1002
                                           03  1E 000C8          BGEQU   7$                                          ;
                             51            52  D0 000CA          MOVL    R2, R1                                      ;
                69           51          6B40  C3 000CD 7$:      SUBL3   BADBLOCK_LBN[R0], R1, (R9)                  ; 1003
                                    0000G  CF  D7 000D2          DECL    BADBLOCK_TOTAL                              ; 1004
                             57     0000G  CF  D0 000D6          MOVL    BADBLOCK_TOTAL, R7                          ; 1005
                58           57            56  C3 000DB          SUBL3   J, R7, R8                                   ;
                             58            04  C4 000DF          MULL2   #4, R8                                      ;
                                         6B46  DF 000E2          PUSHAL  BADBLOCK_LBN[J]                             ;
                                         6B40  DF 000E5          PUSHAL  BADBLOCK_LBN[R0]                            ;
                9E           9E            58  28 000E8          MOVC3   R8, @(SP)+, @(SP)+                          ;
                                    0000GCF46  DF 000EC          PUSHAL  BADBLOCK_CNT[J]                             ; 1006
                9E           69            58  28 000F1          MOVC3   R8, (R9), @(SP)+                            ;
                                    0000GCF47  D4 000F5          CLRL    BADBLOCK_CNT[R7]                            ; 1007
                                           A5  11 000FA          BRB     6$                                          ; 0998
                                           56  D6 000FC 8$:      INCL    J                                           ; 1012
                                           5A  D6 000FE          INCL    C                                           ; 1013
                             02            5A  D1 00100          CMPL    C, #2                                       ; 1014
                                           9C  19 00103          BLSS    6$                                          ;
                                           04  00105 9$:         RET                                                 ; 1018

; Routine Size:  262 bytes,    Routine Base:  $CODE$ + 02A2


;  603          1019  1
;  604          1020  1 END
;  605          1021  0 ELUDOM



                                                                   .EXTRN  LIB$SIGNAL, LIB$STOP
;                                PSECT SUMMARY
;
;       Name                      Bytes                      Attributes
;
;  $CODE$                           936  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                            512  NOVEC,  WRT,  RD ,NOEXE,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        26         0      1000          00:01.9
;                                            COMMAND QUALIFIERS

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

; Size:            936 code + 512 data bytes
; Run Time:           00:22.3
; Elapsed Time:       00:46.1
; Lines/CPU Min:        2749
; Lexemes/CPU-Min: 29270
; Memory Used:   151 pages
; Compilation Complete
```

0186 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0187 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY