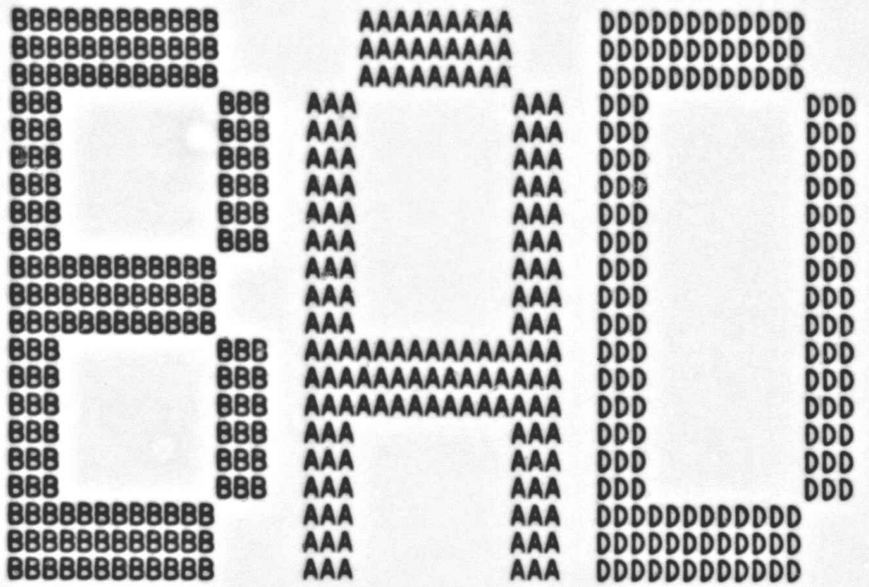
BBBBBBBBBBBB AAAAAAAAA DDDDDDDDDDD
BBBBBBBBBBBB AAAAAAAAA DDDDDDDDDDD
BBBBBBBBBBBB AAAAAAAAA DDDDDDDDDDD
BBB BBB AAA AAA DDD DDD
BBB BBB AAA AAA DDD DDD
BBB BBB AAA AAA DDD DDD
BBB BBB AAA AAA DDD DDD
BBB BBB AAA AAA DDD DDD
BBB BBB AAA AAA DDD DDD
BBBBBBBBBBBB AAA AAA DDD DDD
BBBBBBBBBBBB AAA AAA DDD DDD
BBBBBBBBBBBB AAA AAA DDD DDD
BBB BBB AAAAAAAAAAAAAAA DDD DDD
BBB BBB AAAAAAAAAAAAAAA DDD DDD
BBB BBB AAAAAAAAAAAAAAA DDD DDD
BBB BBB AAA AAA DDD DDD
BBB BBB AAA AAA DDD DDD
BBB BBB AAA AAA DDD DDD
BBBBBBBBBBBB AAA AAA DDDDDDDDDDD
BBBBBBBBBBBB AAA AAA DDDDDDDDDDD
BBBBBBBBBBBB AAA AAA DDDDDDDDDDD

FILEID**BADBLOCKS

```
BBBBBBBB      AAAAAA    DDDDDDDD    BBBBBBBB    LL           OOOOOO      CCCCCCCC  KK      KK    SSSSSSSS
BBBBBBBB      AAAAAA    DDDDDDDD    BBBBBBBB    LL           OOOOOO      CCCCCCCC  KK      KK    SSSSSSSS
BB      BB   AA    AA   DD      DD  BB      BB  LL          OO      OO  CC          KK      KK    SS
BB      BB   AA    AA   DD      DD  BB      BB  LL          OO      OO  CC          KK      KK    SS
BB      BB   AA    AA   DD      DD  BB      BB  LL          OO      OO  CC          KK    KK      SS
BB      BB   AA    AA   DD      DD  BB      BB  LL          OO      OO  CC          KK    KK      SS
BBBBBBBB     AA    AA   DD      DD  BBBBBBBB    LL          OO      OO  CC          KKKKKK          SSSSSS
BBBBBBBB     AA    AA   DD      DD  BBBBBBBB    LL          OO      OO  CC          KKKKKK          SSSSSS
BB      BB   AAAAAAAAAA DD      DD  BB      BB  LL          OO      OO  CC          KK    KK              SS
BB      BB   AAAAAAAAAA DD      DD  BB      BB  LL          OO      OO  CC          KK    KK              SS
BB      BB   AA    AA   DD      DD  BB      BB  LL          OO      OO  CC          KK      KK            SS      ....
BB      BB   AA    AA   DD      DD  BB      BB  LL          OO      OO  CC          KK      KK            SS      ....
BBBBBBBB     AA    AA   DDDDDDDD    BBBBBBBB    LLLLLLLLLL    OOOOOO      CCCCCCCC  KK      KK    SSSSSSSS      ....
BBBBBBBB     AA    AA   DDDDDDDD    BBBBBBBB    LLLLLLLLLL    OOOOOO      CCCCCCCC  KK      KK    SSSSSSSS      ....

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 badblocks(%TITLE 'Analyze/Media Bad Block Manipulation Procedures'
;     2        0002  0                 IDENT = 'V04-000') =
;     3        0003  1 BEGIN
;     4        0004  1 !
;     5        0005  1 !**********************************************************************************
;     6        0006  1 !*                                                                                *
;     7        0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                       *
;     8        0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                         *
;     9        0009  1 !*  ALL RIGHTS RESERVED.                                                           *
;    10        0010  1 !*                                                                                *
;    11        0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED         *
;    12        0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE        *
;    13        0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER        *
;    14        0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY        *
;    15        0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY        *
;    16        0016  1 !*  TRANSFERRED.                                                                   *
;    17        0017  1 !*                                                                                *
;    18        0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE         *
;    19        0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT         *
;    20        0020  1 !*  CORPORATION.                                                                   *
;    21        0021  1 !*                                                                                *
;    22        0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS        *
;    23        0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                        *
;    24        0024  1 !*                                                                                *
;    25        0025  1 !*                                                                                *
;    26        0026  1 !**********************************************************************************
;    27        0027  1 !
;    28        0028  1 !
;    29        0029  1 !++
;    30        0030  1 !
;    31        0031  1 ! Facility:
;    32        0032  1 !
;    33        0033  1 !       Analyze/Media
;    34        0034  1 !
;    35        0035  1 ! Abstract:
;    36        0036  1 !
;    37        0037  1 !       This module contains the procedures required to handle bad block
;    38        0038  1 !       processing for last track and non-last track devices.
;    39        0039  1 !
;    40        0040  1 ! Environment:
;    41        0041  1 !
;    42        0042  1 !       VAX/VMS User Mode, Non-Privileged
;    43        0043  1 !
;    44        0044  1 ! Author:
;    45        0045  1 !
;    46        0046  1 !       Michael T. Rhodes,              Creation Date: July, 1982
;    47        0047  1 !
;    48        0048  1 ! Modified By:
;    49        0049  1 !
;    50        0050  1 !       V03-002 MTR0007         Michael T. Rhodes       19-May-1983
;    51        0051  1 !               Fix 'bad$init_buffers' to protect against boundary condition
;    52        0052  1 !               when three patterns are specified.
;    53        0053  1 !
;    54        0054  1 !       V03-001 MTR0001         Michael T. Rhodes       15-Dec-1982
;    55        0055  1 !               Modify routine 'bad$init_buffers' to use upto an octaword
;    56        0056  1 !               of user supplied test pattern.  The 'pattern' is located
;    57        0057  1 !               in the 'bad$gl_bad_term' data vector, with the number of
```

```
;   58          0058  1 !                    long word elements described in bad$gb_term_count.
;   59          0059  1 !
;   60          0060  1 !--
```

```
;    62          0061  1 %SBTTL 'Bad Block Information Format Specifications'
;    63          0062  1 !++
;    64          0063  1 !
;    65          0064  1 ! This utility supports bad block information in two formats "last track" and
;    66          0065  1 ! "non-last track".  A last track device is one in which the bad block
;    67          0066  1 ! information is located on the last physical track of the device and is the
;    68          0067  1 ! default for ALL devices containing more than 4096 blocks.  Any device which
;    69          0068  1 ! falls outside of this category (ie. less than 4096 blocks) will be deemed a
;    70          0069  1 ! non last track device.  A block is defined as 512 bytes which may actually
;    71          0070  1 ! be made up of one or more physical sectors.  The devices are analyzed in a
;    72          0071  1 ! standard fashion, which is to write one or more test patterns onto the media
;    73          0072  1 ! followed by reading and verifing these patterns against the test pattern
;    74          0073  1 ! buffer.  The bad block information will be written to the software detected
;    75          0074  1 ! bad block area (for last track devices it is written on the last track
;    76          0075  1 ! following the manufacturers detected bad block file and on non last track
;    77          0076  1 ! devices it is located on the last good block on the device) following the
;    78          0077  1 ! analysis.
;    79          0078  1 !
;    80          0079  1 ! NOTES:        1.  In either case (last track or non-last track) the size of
;    81          0080  1 !                  the bad block files are a constant of 512 bytes.
;    82          0081  1 !
;    83          0082  1 !              2.  The MDBSF is redundantly recorded on the first 10 blocks of
;    84          0083  1 !                  the last track, with the SDBSF recorded on the remaining
;    85          0084  1 !                  blocks of the last track.  The MDBSF is not modified in any
;    86          0085  1 !                  way nor are the first ten sectors of the last track.
;    87          0086  1 !
;    88          0087  1 !
;    89          0088  1 ! FORMAT of Last Track Bad Block Information:
;    90          0089  1 !
;    91          0090  1 !{alias: The Manufacturers Detected Bad Sector File (MDBSF)
;    92          0091  1 !         and the  Software Detected Bad Sector File (SDBSF)}
;    93          0092  1 !
;    94          0093  1 !       31                                      0
;    95          0094  1 !       +-------+-------+-------+-------+
;    96          0095  1 !       |    Cartridge Serial Number    | 0
;    97          0096  1 !       +---------------+---------------+
;    98          0097  1 !       | Cartridge     |    Not Used   | 4     The Cartridge Identifier word
;    99          0098  1 !       | Identifier    |               |       contains all:
;   100          0099  1 !       +-------+-------+---------------+         0's - Data Cartridge
;   101          0100  1 !       | Track | Sector|    Cylinder   | 8       1's - Alignment Cartridge
;   102          0101  1 !       |       |       |               |
;   103          0102  1 !       +-------+-------+---------------+
;   104          0103  1 !       |   :   |   :   |       :       |
;   105          0104  1 !       |   :   |   :   |       :       |
;   106          0105  1 !       +-------+-------+---------------+
;   107          0106  1 !       |  - 1  |  - 1  |      - 1      | 251.
;   108          0107  1 !       |       |       |               |
;   109          0108  1 !       +-------+-------+---------------+
;   110          0109  1 !
;   111          0110  1 !                                         Note that this format implies
;   112          0111  1 !                                         that no more than 126 sectors
;   113          0112  1 !                                         can be bad!
;   114          0113  1 !
;   115          0114  1 !--
```

```
;  117     0115  1 !++
;  118     0116  1 !
;  119     0117  1 ! FORMAT of Non-Last Track Bad Block Information:
;  120     0118  1 !
;  121     0119  1 !       31                                      0
;  122     0120  1 !       +-------+-------+-------+-------+
;  123     0121  1 !       | Desc  | Desc  | Addr  | Count | 0
;  124     0122  1 !       | Avail | Used  |Fld Siz|Fld Siz|
;  125     0123  1 !       +-------+-------+-------+-------+
;  126     0124  1 !       | Low Order LBN | Block | High  | 4
;  127     0125  1 !       |               | Count | LBN   |
;  128     0126  1 !       +---------------+-------+-------+
;  129     0127  1 !       |       :       |   :   |   :   | :
;  130     0128  1 !       |       :       |   :   |   :   | :
;  131     0129  1 !       +---------------+-------+-------+
;  132     0130  1 !       |       0       |   0   |   0   | :
;  133     0131  1 !       |               |       |       | :
;  134     0132  1 !       +---------------+-------+-------+
;  135     0133  1 !       |  Checksum of  |    Not Used   | 251.
;  136     0134  1 !       |Bad Block Info |               |
;  137     0135  1 !       +---------------+---------------+
;  138     0136  1 !
;  139     0137  1 !       NOTE:   Only 126 entries are initially available in the buffer due to
;  140     0138  1 !               the overhead of the header and Checksum information.
;  141     0139  1 !
;  142     0140  1 !       Field Definitions:
;  143     0141  1 !
;  144     0142  1 !               Count Fld Siz           - Size of the Block count field in bytes.
;  145     0143  1 !               Addr Fld Siz            - Size of the Address field in bytes.
;  146     0144  1 !               Desc Used               - Number of descriptors used for bad blocks.
;  147     0145  1 !               Desc Avail              - Number of descriptors available.
;  148     0146  1 !               High LBN                - High order byte of the Logical Block Number.
;  149     0147  1 !               Block Count             - Number of bad blocks in this set.
;  150     0148  1 !               Low Order LBN           - Low order portion of the Logical Block Number.
;  151     0149  1 !               Checksum Bad Block Info - Check sum of the bad block information.
;  152     0150  1 !
;  153     0151  1 !--
```

```
;  155     0152  1 %SBTTL 'Declarations'
;  156     0153  1 !
;  157     0154  1 ! Include files:
;  158     0155  1 !
;  159     0156  1 REQUIRE 'lib$:baddef';                          ! Define BAD's structures.
;  160     0276  1 LIBRARY 'SYS$LIBRARY:LIB';                      ! Define VMS structures.
;  161     0277  1
;  162     0278  1 !
;  163     0279  1 ! Table of Contents:
;  164     0280  1 !
;  165     0281  1 FORWARD ROUTINE
;  166     0282  1     bad$check_ltk       : NOVALUE,              ! Check last track device for a user supplied entry.
;  167     0283  1     bad$check_nlt       : NOVALUE,              ! Check non last track device for a user supplied en
;  168     0284  1     bad$cvt_ltk_long    : NOVALUE,              ! Convert last track format to longwords.
;  169     0285  1     bad$cvt_nlt_long    : NOVALUE,              ! Convert non last track format to longwords.
;  170     0286  1     bad$cvt_phy_log,                            ! Convert Physical address to logical.
;  171     0287  1     bad$init_buffers    : NOVALUE,              ! Initialize the contents of the data buffers.
;  172     0288  1     check_for_blk0      : NOVALUE,              ! Check block number for block 0.
;  173     0289  1     cvt_log_phy,                                ! Convert Logical address to Physical.
;  174     0290  1     cvt_long_nlt        : NOVALUE,              ! Convert longwords to non last track format.
;  175     0291  1     insert_blocks       : NOVALUE,              ! Insert the indicated bad blocks into the SDBSF.
;  176     0292  1     lookup_bad;                                 ! Look up specified block in the Bad Sector files.
;  177     0293  1
;  178     0294  1 !
;  179     0295  1 ! Private Storage
;  180     0296  1 !
;  181     0297  1 BIND
;  182     0298  1     mdbsf = $DESCRIPTOR ('MDBSF') : $BBLOCK,
;  183     0299  1     sdbsf = $DESCRIPTOR ('SDBSF') : $BBLOCK;
;  184     0300  1
;  185     0301  1 !
;  186     0302  1 ! External references:
;  187     0303  1 !
;  188     0304  1 EXTERNAL
;  189     0305  1     bad$gl_bad_term     : VECTOR [4, LONG],     ! Information vector.
;  190     0306  1     bad$gb_block_fact   : BYTE,                 ! Number of sectors per block.
;  191     0307  1     bad$gl_chan,                                ! Channel number for device.
;  192     0308  1     bad$gl_context      : BITVECTOR [32],       ! Global control context flags.
;  193     0309  1     bad$ga_bufadr       : VECTOR [2, LONG],     ! Address of data transfer buffers.
;  194     0310  1     bad$ga_device       : $BBLOCK [dsc$c_s_bln], ! Address of device name descriptor.
;  195     0311  1     bad$ga_input_desc   : $BBLOCK [dsc$c_s_bln], ! Address of general purpose input descriptor.
;  196     0312  1     bad$gl_func,                                ! IO function code for $QIO.
;  197     0313  1     bad$ga_mdbsf        : REF $BBLOCK,          ! Address of the Manufacturers Detected Bad Sector F
;  198     0314  1     bad$ga_sdbsf        : REF $BBLOCK,          ! Address of the Software Detected Bad Sector File.
;  199     0315  1     bad$gl_sdbsf_ptr,                           ! Pointer to the first available entry in the SDBSF.
;  200     0316  1     bad$gl_sectors,                             ! Number of sectors per track.
;  201     0317  1     bad$gb_term_count   : BYTE,                 ! Number of elements in bad$gl_bad_term.
;  202     0318  1     bad$ga_tpb,                                 ! Address of the Test Pattern Buffer.
;  203     0319  1     bad$gl_tracks,                              ! Number of tracks per cylinder.
;  204     0320  1     bad$gl_trnsfr_cnt;                          ! Number of bytes per transfer.
;  205     0321  1
;  206     0322  1 !
;  207     0323  1 ! Define message codes...
;  208     0324  1 !
;  209     0325  1 EXTERNAL LITERAL
;  210     0326  1     bad$_bbfovf,                                ! Bad block file overflow.
;  211     0327  1     bad$_dupblknum,                             ! Duplicate block number.
```

```
;  212          0328  1      bad$_readerr,                          ! Read error.
;  213          0329  1      bad$_srclin,                           ! Source line.
;  214          0330  1      bad$_writeerr;                         ! Write error.
;  215          0331  1
```

```
;  217     0332  1 %SBTTL  'bad$check_ltk -- Check Last Track bad block information'
;  218     0333  1 GLOBAL ROUTINE  bad$check_ltk (lobound, hibound) : NOVALUE =
;  219     0334  1 !++
;  220     0335  1 !
;  221     0336  1 ! Functional Description:
;  222     0337  1 !
;  223     0338  1 !    Check last track bad block information files.
;  224     0339  1 !
;  225     0340  1 ! NOTE: "lobound" and "hibound" are specified in lbn form.
;  226     0341  1 !       These values will be converted by "lookup_bad" into
;  227     0342  1 !       the appropriate formats for last track or non last
;  228     0343  1 !       track information.
;  229     0344  1 !
;  230     0345  1 ! Inputs:
;  231     0346  1 !
;  232     0347  1 !    lobound    val     the lower bound block number to look up.
;  233     0348  1 !
;  234     0349  1 !    hibound    val     the upper bound block number to look up.
;  235     0350  1 !
;  236     0351  1 ! Side Effects:
;  237     0352  1 !
;  238     0353  1 !    Any block(s) that are not found in either the MDBSF or the SDBSF,
;  239     0354  1 !    will be added to the SDBSF via a call to "insert_blocks".
;  240     0355  1 !
;  241     0356  1 !--
;  242     0357  2 BEGIN
;  243     0358  2 LOCAL
;  244     0359  2     ltentry;
;  245     0360  2
;  246     0361  2 check_for_blk0 (lobound);                       ! If block 0 is bad, flag it.
;  247     0362  2 INCR entry FROM .lobound TO .hibound DO         ! If the block is not in the
;  248     0363  3     BEGIN                                       ! MDBSF or SDBSF (as they exist)
;  249     0364  3     IF NOT lookup_bad (.entry, .bad$ga_mdbsf)   ! then enter this block into
;  250     0365  3     THEN                                        ! the SDBSF and bump the first
;  251     0366  4         BEGIN                                   ! free entry pointer.
;  252     0367  4         IF NOT lookup_bad (.entry, .bad$ga_sdbsf)
;  253     0368  4         THEN
;  254     0369  4             insert_blocks (.entry, 0)           ! Insert the current block.
;  255     0370  4         ELSE
;  256     0371  4             IF .bad$gl_context [ctx_v_init]     ! Only display duplicate block number xxxxx,
;  257     0372  4             THEN                                ! messages during the initialization phase.
;  258     0373  4                 SIGNAL (bad$_dupblknum, 2,      ! Informational message.
;  259     0374  4                         .entry,
;  260     0375  4                         sdbsf,
;  261     0376  4                         bad$_srclin, 1, bad$ga_input_desc);
;  262     0377  4         END
;  263     0378  3     ELSE
;  264     0379  3         IF .bad$gl_context [ctx_v_init]         ! Only display duplicate block number xxxxx,
;  265     0380  3         THEN                                    ! messages during the initialization phase.
;  266     0381  3             SIGNAL (bad$_dupblknum, 2,          ! Informational message.
;  267     0382  3                     .entry,
;  268     0383  3                     mdbsf,
;  269     0384  3                     bad$_srclin, 1, bad$ga_input_desc);
;  270     0385  2     END;
;  271     0386  1 END;    ! of GLOBAL ROUTINE bad$check_ltk
```

```
                                                                .TITLE  BADBLOCKS Analyze/Media Bad Block Manipulation
;                                                                       Procedures
                                                                .IDENT  \V04-000\

                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                        46  53  42  44  4D  00000 P.AAB:  .ASCII  \MDBSF\                                  ;
                                                            00005       .BLKB   3                                           ;
                                                00000005    00008 P.AAA:  .LONG   5                                         ;
                                                00000000'   0000C         .ADDRESS P.AAB                                    ;
                                        46  53  42  44  53  00010 P.AAD:  .ASCII  \SDBSF\                                  ;
                                                            00015       .BLKB   3                                           ;
                                                00000005    00018 P.AAC:  .LONG   5                                         ;
                                                00000000'   0001C         .ADDRESS P.AAD                                    ;

                                                                MDBSF=              P.AAA
                                                                SDBSF=              P.AAC
                                                                .EXTRN  BAD$GL_BAD_TERM
                                                                .EXTRN  BAD$GB_BLOCK_FACT
                                                                .EXTRN  BAD$GL_CHAN, BAD$GL_CONTEXT
                                                                .EXTRN  BAD$GA_BUFADR, BAD$GA_DEVICE
                                                                .EXTRN  BAD$GA_INPUT_DESC
                                                                .EXTRN  BAD$GL_FUNC, BAD$GA_MDBSF
                                                                .EXTRN  BAD$GA_SDBSF, BAD$GL_SDBSF_PTR
                                                                .EXTRN  BAD$GL_SECTORS, BAD$GB_TERM_COUNT
                                                                .EXTRN  BAD$GA_TPB, BAD$GL_TRACKS
                                                                .EXTRN  BAD$GL_TRNSFR_CNT
                                                                .EXTRN  BAD$_BBFOVF, BAD$_DUPBLKNUM
                                                                .EXTRN  BAD$_READERR, BAD$_SRCLIN
                                                                .EXTRN  BAD$_WRITEERR

                                                                .PSECT  $CODE$,NOWRT,2

                                                    000C 00000  .ENTRY  BAD$CHECK_LTK, Save R2,R3                       ; 0333
                                    53 00000000G    8F   DO 00002       MOVL   #BAD$_SRCLIN, R3                         ;
                                               04   AC   9F 00009       PUSHAB LOBOUND                                 ; 0361
                                0000V  CF           01   FB 0000C       CALLS  #1, CHECK_FOR_BLKO                       ;
                            52    04   AC           01   C3 00011       SUBL3  #1, LOBOUND, ENTRY                       ; 0362
                                                    5E   11 00016       BRB    5$                                       ;
                                           0000G    CF   DD 00018 1$:   PUSHL  BAD$GA_MDBSF                            ; 0364
                                                    52   DD 0001C       PUSHL  ENTRY                                   ;
                                0000V  CF           02   FB 0001E       CALLS  #2, LOOKUP_BAD                          ;
                                       2D           50   E8 00023       BLBS   R0, 3$                                  ;
                                           0000G    CF   DD 00026       PUSHL  BAD$GA_SDBSF                            ; 0367
                                                    52   DD 0002A       PUSHL  ENTRY                                   ;
                                0000V  CF           02   FB 0002C       CALLS  #2, LOOKUP_BAD                          ;
                                       0B           50   E8 00031       BLBS   R0, 2$                                  ;
                                                    7E   D4 00034       CLRL   -(SP)                                   ; 0369
                                                    52   DD 00036       PUSHL  ENTRY                                   ;
                                0000V  CF           02   FB 00038       CALLS  #2, INSERT_BLOCKS                       ;
                                                    37   11 0003D       BRB    5$                                       ;
                            31  0000G  CF           04   E1 0003F 2$:   BBC    #4, BAD$GL_CONTEXT, 5$                   ; 0371
                                           0000G    CF   9F 00045       PUSHAB BAD$GA_INPUT_DESC                       ; 0373
                                                    01   DD 00049       PUSHL  #1                                      ;
                                                    53   DD 0004B       PUSHL  R3                                      ;
                                           0000'    CF   9F 0004D       PUSHAB SDBSF                                   ;
                                                    12   11 00051       BRB    4$                                       ; 0374
```

```
                 1D     0000G  CF              04  E1 00053 3$:    BBC     #4, BAD$GL_CONTEXT, 5$          ; 0379
                                        0000G   CF  9F 00059        PUSHAB  BAD$GA_INPUT_DESC               ; 0381
                                                01  DD 0005D        PUSHL   #1                              ;
                                                53  DD 0005F        PUSHL   R3                              ;
                                        0000'   CF  9F 00061        PUSHAB  MDBSF                           ;
                                                52  DD 00065 4$:    PUSHL   ENTRY                           ; 0382
                                                02  DD 00067        PUSHL   #2                              ; 0381
                                    00000000G   8F  DD 00069        PUSHL   #BAD$_DUPBLKNUM                 ;
              00000000G  00                     07  FB 0006F        CALLS   #7, LIB$SIGNAL                  ;
           9D            52        08          AC  F3 00076 5$:    AOBLEQ  HIBOUND, ENTRY, 1$              ; 0362
                                                04  0007B        RET                                     ; 0386

; Routine Size:  124 bytes,    Routine Base:  $CODE$ + 0000

;  272          0387  1
```

```
;   274     0388  1 %SBTTL  'bad$check_nlt -- Check Non Last Track bad block information'
;   275     0389  1 GLOBAL ROUTINE  bad$check_nlt (lobound, hibound) : NOVALUE =
;   276     0390  1 !++
;   277     0391  1 !
;   278     0392  1 ! Functional Description:
;   279     0393  1 !
;   280     0394  1 ! Check for prior bad block information for non last track devices.
;   281     0395  1 !
;   282     0396  1 ! NOTE: "lobound" and "hibound" are specified in lbn form.
;   283     0397  1 !       These values will be converted by "lookup_bad" into
;   284     0398  1 !       the appropriate formats for last track or non last
;   285     0399  1 !       track information.
;   286     0400  1 !
;   287     0401  1 ! Inputs:
;   288     0402  1 !
;   289     0403  1 !    lobound    val     the lower bound block number to look up.
;   290     0404  1 !
;   291     0405  1 !    hibound    val     the upper bound block number to look up.
;   292     0406  1 !
;   293     0407  1 ! Side Effects:
;   294     0408  1 !
;   295     0409  1 !    Any block(s) that are not found in the SDBSF, will be added via
;   296     0410  1 !    a call to "insert_blocks".
;   297     0411  1 !
;   298     0412  1 !--
;   299     0413  2 BEGIN
;   300     0414  2 LOCAL
;   301     0415  2     count,
;   302     0416  2     next,
;   303     0417  2     nltentry,
;   304     0418  2     start_block;
;   305     0419  2
;   306     0420  2 check_for_blk0 (lobound);                       ! If block 0 is bad, flag it.
;   307     0421  2 INCR entry FROM .lobound TO .hibound DO         ! Look for each block individually.
;   308     0422  3     BEGIN
;   309     0423  3     IF NOT lookup_bad (.entry, .bad$ga_sdbsf)   ! Look for the low bound lbn.
;   310     0424  3     THEN                                        ! If it was NOT found, then save the starting block
;   311     0425  4         BEGIN                                   ! number and count the number of interveining blocks
;   312     0426  4         count = 0;                              ! on file to the next block recorded.
;   313     0427  4         next = start_block = .entry;
;   314     0428  4         INCR next_entry FROM .entry + 1 TO .hibound DO
;   315     0429  5             BEGIN
;   316     0430  5             IF NOT lookup_bad (.next_entry, .bad$ga_sdbsf)
;   317     0431  5             THEN
;   318     0432  5                 count = .count + 1
;   319     0433  5             ELSE
;   320     0434  6                 BEGIN
;   321     0435  6                 IF .bad$gl_context [ctx_v_init]     ! Only display duplicate block number xxxxx,
;   322     0436  6                 THEN                                ! messages during the initialization phase.
;   323     0437  6                     SIGNAL (bad$_dupblknum, 2,      ! Informational message.
;   324     0438  6                             .next_entry,
;   325     0439  6                             sdbsf,
;   326     0440  6                             bad$_srclin, 1, bad$ga_input_desc);
;   327     0441  6                 next = .next_entry;                 ! Remember the "next" block number to check.
;   328     0442  6                 EXITLOOP;
;   329     0443  5                 END;
;   330     0444  5             next = .next_entry;                     ! Remember the "next" block number to check.
```

```
;  331          0445  4            END;
;  332          0446  4        entry = .next;                                  ! Update the "next" block number to check.
;  333          0447  4        insert_blocks (.start_block, .count);           ! Add the new bad blocks to the list.
;  334          0448  4        END
;  335          0449  3    ELSE
;  336          0450  3        IF .bad$gl_context [ctx_v_init]                 ! Only display the dupliacte block number xxxxx,
;  337          0451  3        THEN                                            ! messages during the initialization phase.
;  338          0452  3            SIGNAL (bad$_dupblknum, 2,                  ! Informational message.
;  339          0453  3                    .entry,
;  340          0454  3                    sdbsf,
;  341          0455  3                    bad$_srclin, 1, bad$ga_input_desc);
;  342          0456  2    END;
;  343          0457  1 END;     ! of GLOBAL ROUTINE bad$check_nlt



                                                03FC 00000          .ENTRY  BAD$CHECK_NLT, Save R2,R3,R4,R5,R6,R7,R8,R9 ; 0389
                                    59 00000000G    00   9E 00002      MOVAB   LIB$SIGNAL, R9                                  ;
                                    58 00000000G    8F   D0 00009      MOVL    #BAD$_DUPBLKNUM, R8                             ;
                                    57 00000000G    8F   D0 00010      MOVL    #BAD$_SRCLIN, R7                                ;
                                            04      AC   9F 00017      PUSHAB  LOBOUND                                         ; 0420
                                0000V   CF              01   FB 0001A      CALLS   #1, CHECK_FOR_BLK0                              ;
                                  53      04    AC      01   C3 0001F      SUBL3   #1, LOBOUND, ENTRY                              ; 0421
                                                        7E   11 00024      BRB     9$                                              ;
                                                0000G   CF   DD 00026 1$:  PUSHL   BAD$GA_SDBSF                                    ; 0423
                                                        53   DD 0002A      PUSHL   ENTRY                                           ;
                                0000V   CF              02   FB 0002C      CALLS   #2, LOOKUP_BAD                                  ;
                                        55              50   E8 00031      BLBS    R0, 8$                                          ;
                                                        55   D4 00034      CLRL    COUNT                                           ; 0426
                                        56              53   D0 00036      MOVL    ENTRY, START_BLOCK                              ; 0427
                                        54              53   D0 00039      MOVL    ENTRY, NEXT                                     ;
                                        52              53   D0 0003C      MOVL    ENTRY, NEXT_ENTRY                               ; 0428
                                                        35   11 0003F      BRB     6$                                              ;
                                                0000G   CF   DD 00041 2$:  PUSHL   BAD$GA_SDBSF                                    ; 0430
                                                        52   DD 00045      PUSHL   NEXT_ENTRY                                      ;
                                0000V   CF              02   FB 00047      CALLS   #2, LOOKUP_BAD                                  ;
                                        04              50   E8 0004C      BLBS    R0, 3$                                          ;
                                                        55   D6 0004F      INCL    COUNT                                           ; 0432
                                                        20   11 00051      BRB     5$                                              ;
                        15      0000G   CF              04   E1 00053 3$:  BBC     #4, BAD$GL_CONTEXT, 4$                          ; 0435
                                                0000G   CF   9F 00059      PUSHAB  BAD$GA_INPUT_DESC                               ; 0437
                                                        01   DD 0005D      PUSHL   #1                                              ;
                                                        57   DD 0005F      PUSHL   R7                                              ;
                                                0000'   CF   9F 00061      PUSHAB  SDBSF                                           ;
                                                        52   DD 00065      PUSHL   NEXT_ENTRY                                      ; 0438
                                                        02   DD 00067      PUSHL   #2                                              ; 0437
                                                        58   DD 00069      PUSHL   R8                                              ;
                                        69              07   FB 0006B      CALLS   #7, LIB$SIGNAL                                  ;
                                        54              52   D0 0006E 4$:  MOVL    NEXT_ENTRY, NEXT                                ; 0441
                                                        08   11 00071      BRB     7$                                              ; 0434
                                        54              52   D0 00073 5$:  MOVL    NEXT_ENTRY, NEXT                                ; 0444
                        C6              52      08      AC   F3 00076 6$:  AOBLEQ  HIBOUND, NEXT_ENTRY, 2$                        ; 0428
                                        53              54   D0 0007B 7$:  MOVL    NEXT, ENTRY                                     ; 0446
                                                        55   DD 0007E      PUSHL   COUNT                                           ; 0447
                                                        56   DD 00080      PUSHL   START_BLOCK                                     ;
```

```
                     0000V  CF                 02  FB 00082          CALLS   #2, INSERT_BLOCKS                                ;
                                                1B  11 00087          BRB     9$                                          ; 0423
                 15  0000G  CF                 04  E1 00089 8$:      BBC     #4, BAD$GL_CONTEXT, 9$                      ; 0450
                                      0000G  CF  9F 0008F          PUSHAB  BAD$GA_INPUT_DESC                           ; 0452
                                             01  DD 00093          PUSHL   #1                                          ;
                                             57  DD 00095          PUSHL   R7                                          ;
                                      0000'  CF  9F 00097          PUSHAB  SDBSF                                       ;
                                             53  DD 0009B          PUSHL   ENTRY                                       ; 0453
                                             02  DD 0009D          PUSHL   #2                                          ; 0452
                                             58  DD 0009F          PUSHL   R8                                          ;
                          69                 07  FB 000A1          CALLS   #7, LIB$SIGNAL                              ;
        FF7B             53  01        08   AC  F1 000A4 9$:      ACBL    HIBOUND, #1, ENTRY, 1$                      ; 0421
                                                04 000AB          RET                                                 ; 0457

; Routine Size:  172 bytes,    Routine Base:  $CODE$ + 007C

;  344          0458  1
```

```
;  346          0459  1 %SBTTL  'Conversion and Formatting routines'
;  347          0460  1 GLOBAL ROUTINE bad$cvt_ltk_long (entry, sec, trk, cyl) : NOVALUE =
;  348          0461  1 !++
;  349          0462  1 !
;  350          0463  1 ! Functional Description:
;  351          0464  1 !
;  352          0465  1 !    Convert last track entry to long words.
;  353          0466  1 !
;  354          0467  1 ! Inputs:
;  355          0468  1 !
;  356          0469  1 !       entry   val     the last track entry to covert.
;  357          0470  1 !
;  358          0471  1 ! Outputs:
;  359          0472  1 !
;  360          0473  1 !       sec     adr     the address of the longword to receive the sector number.
;  361          0474  1 !       trk     adr     the address of the longword to receive the track number.
;  362          0475  1 !       cyl     adr     the address of the longword to receive the cylinder number.
;  363          0476  1 !
;  364          0477  1 !--
;  365          0478  2 BEGIN
;  366          0479  2 MAP
;  367          0480  2     entry  : $BBLOCK,
;  368          0481  2     cyl    : REF $BBLOCK,
;  369          0482  2     trk    : REF $BBLOCK,
;  370          0483  2     sec    : REF $BBLOCK;
;  371          0484  2
;  372          0485  2 .cyl = .entry [ltk_w_cylinder];                  !+---+---+-------+
;  373          0486  2 .trk = .entry [ltk_b_track];                     !!trk!sec!  cyl  !
;  374          0487  2 .sec = .entry [ltk_b_sector];                    !+---+---+-------+
;  375          0488  2
;  376          0489  1 END;    ! of GLOBAL ROUTINE bad$cvt_ltk_long



                                        0000 00000          .ENTRY  BAD$CVT_LTK_LONG, Save nothing                  ; 0460
                              10  BC    04  AC  3C 00002    MOVZWL  ENTRY, @CYL                                     ; 0485
                              0C  BC    07  AC  9A 00007    MOVZBL  ENTRY+3, @TRK                                   ; 0486
                              08  BC    06  AC  9A 0000C    MOVZBL  ENTRY+2, @SEC                                   ; 0487
                                            04  00011       RET                                                     ; 0489

; Routine Size:  18 bytes,    Routine Base:  $CODE$ + 0128

;  377          0490  1
```

```
;  379          0491  1 GLOBAL ROUTINE  bad$cvt_nlt_long (entry, cnt, lbn) : NOVALUE =
;  380          0492  1 !++
;  381          0493  1 !
;  382          0494  1 ! Functional Description:
;  383          0495  1 !
;  384          0496  1 !    Convert a non last track bad block entry into a pair of long words
;  385          0497  1 !    containing the count value and Logical Block Number.
;  386          0498  1 !
;  387          0499  1 ! Inputs:
;  388          0500  1 !
;  389          0501  1 !       entry   val     the non last track entry to convert.
;  390          0502  1 !
;  391          0503  1 ! Outputs:
;  392          0504  1 !
;  393          0505  1 !       cnt     adr     the address of a longword to receive the count of
;  394          0506  1 !                       contigous bad blocks.
;  395          0507  1 !       lbn     adr     the address of a longword to receive the logical block number.
;  396          0508  1 !
;  397          0509  1 !--
;  398          0510  2 BEGIN
;  399          0511  2 MAP
;  400          0512  2     entry : $BBLOCK,
;  401          0513  2     cnt : REF $BBLOCK,
;  402          0514  2     lbn : REF $BBLOCK;
;  403          0515  2
;  404          0516  2 .lbn = .entry [nlt_b_highlbn] ^16 OR                    ! Extract the high order and
;  405          0517  2         .entry [nlt_w_lowlbn];                          ! low order portions of the lbn.
;  406          0518  2 .cnt = .entry [nlt_b_sectorcnt];                        ! Extact the count.
;  407          0519  2
;  408          0520  1 END;    ! of GLOBAL ROUTINE bad$cvt_nlt_long



                                                0000 00000          .ENTRY  BAD$CVT_NLT_LONG, Save nothing                  ; 0491
                                  50      04    AC  9A 00002        MOVZBL  ENTRY, R0                                       ; 0516
                              50  50          10  78 00006        ASHL    #16, R0, R0                                     ;
                                  51      06    AC  3C 0000A        MOVZWL  ENTRY+2, R1                                     ; 0517
                          0C  BC  50          51  C9 0000E        BISL3   R1, R0, @LBN                                    ;
                              08  BC      05    AC  9A 00013        MOVZBL  ENTRY+1, @CNT                                   ; 0518
                                                04 00018          RET                                                     ; 0520

; Routine Size:  25 bytes,    Routine Base:  $CODE$ + 013A

;  409          0521  1
```

```
;   411          0522  1 GLOBAL ROUTINE  bad$cvt_phy_log (sec, trk, cyl) =
;   412          0523  1 !++
;   413          0524  1 !
;   414          0525  1 ! Functional Description:
;   415          0526  1 !
;   416          0527  1 !    Convert Physical Address specification to Logical Block Number.
;   417          0528  1 !
;   418          0529  1 ! Outputs:
;   419          0530  1 !
;   420          0531  1 !       sec     val     the sector number
;   421          0532  1 !       trk     val     the track number
;   422          0533  1 !       cyl     val     the cylinder number
;   423          0534  1 !
;   424          0535  1 ! Routine Value:
;   425          0536  1 !
;   426          0537  1 !    The logical block number is returned as the value of the call.
;   427          0538  1 !
;   428          0539  1 !--
;   429          0540  2 BEGIN
;   430          0541  2
;   431          0542  2 RETURN ((.cyl * .bad$gl_tracks + .trk ) * .bad$gl_sectors + .sec) / .bad$gb_block_fact;
;   432          0543  2
;   433          0544  1 END;    ! of GLOBAL ROUTINE bad$cvt_phy_log



                                    0000 00000          .ENTRY  BAD$CVT_PHY_LOG, Save nothing                   ; 0522
                50      0C  AC      0000G  CF   C5 00002    MULL3   BAD$GL_TRACKS, CYL, R0                      ; 0542
                            50        08   AC   C0 00009    ADDL2   TRK, R0                                     ;
                            50      0000G  CF   C4 0000D    MULL2   BAD$GL_SECTORS, R0                          ;
                            50        04   AC   C0 00012    ADDL2   SEC, R0                                     ;
                            51      0000G  CF   9A 00016    MOVZBL  BAD$GB_BLOCK_FACT, R1                       ;
                            50             51   C6 0001B    DIVL2   R1, R0                                      ;
                                           04   04 0001E    RET                                                 ; 0544

; Routine Size:  31 bytes,    Routine Base:  $CODE$ + 0153

;   434          0545  1
```

```
;  436        0546  1 ROUTINE cvt_log_phy (lbn, addr) =
;  437        0547  1 !++
;  438        0548  1 !
;  439        0549  1 ! Functional Description:
;  440        0550  1 !
;  441        0551  1 !    Convert Logical Block Number to a Physical Address
;  442        0552  1 !
;  443        0553  1 ! Inputs:
;  444        0554  1 !
;  445        0555  1 !       lbn      val      the logical block number to convert
;  446        0556  1 !
;  447        0557  1 ! Outputs:
;  448        0558  1 !
;  449        0559  1 !       addr     adr      the address of the longword to receive the
;  450        0560  1 !                         physical address.
;  451        0561  1 !
;  452        0562  1 !--
;  453        0563  2 BEGIN
;  454        0564  2 MAP
;  455        0565  2     addr : REF $BBLOCK;
;  456        0566  2
;  457        0567  2 LOCAL
;  458        0568  2     cyl,
;  459        0569  2     sec,
;  460        0570  2     trk,
;  461        0571  2     blk : VECTOR [2, LONG];
;  462        0572  2
;  463        0573  2 BUILTIN
;  464        0574  2     EDIV;
;  465        0575  2
;  466        0576  2 blk [0] = .lbn * .bad$gb_block_fact;                        ! Scale the logical block
;  467        0577  2 blk [1] = 0;                                               ! number by the blocking factor.
;  468        0578  2 EDIV (bad$gl_sectors, blk, blk, sec);                      ! Extract the sector number.
;  469        0579  2 EDIV (bad$gl_tracks,  blk, cyl, trk);                      ! Extract the cylinder and track numbers.
;  470        0580  2 .addr = .trk ^24 OR                                        ! Format into last track entry.
;  471        0581  2         .sec ^16 OR                                        ! 31    16!15      0
;  472        0582  2         .cyl;                                              ! +---+---+--------+
;  473        0583  2 RETURN TRUE;                                               ! !trk!sec!  cyl   !
;  474        0584  2                                                            ! +---+---+--------+
;  475        0585  1 END;      ! of ROUTINE cvt_log_phy


                                          0004 00000 CVT_LOG_PHY:
                                                     .WORD    Save R2                                          ; 0546
                   5E              04   C2 00002     SUBL2    #4, SP                                           ;
                   50       0000G  CF   9A 00005     MOVZBL   BAD$GB_BLOCK_FACT, R0                            ; 0576
          7E   04  AC              50   C5 0000A     MULL3    R0, LBN, BLK                                     ;
                            04     AE   D4 0000F     CLRL     BLK+4                                            ; 0577
    51     6E      6E       0000G  CF   7B 00012     EDIV     BAD$GL_SECTORS, BLK, BLK, SEC                    ; 0578
    50     52      6E       0000G  CF   7B 00019     EDIV     BAD$GL_TRACKS, BLK, CYL, TRK                     ; 0579
           50      50              18   78 00020     ASHL     #24, R0, R0                                      ; 0580
           51      51              10   78 00024     ASHL     #16, R1, R1                                      ; 0581
                   51              50   C8 00028     BISL2    R0, R1                                           ;
       08  BC      51              52   C9 0002B     BISL3    CYL, R1, @ADDR                                   ; 0582
```

```
                                50          01  D0 00030        MOVL    #1, R0                          ; 0583
                                                04 00033        RET                                     ; 0585

; Routine Size:  52 bytes,    Routine Base:  $CODE$ + 0172

;  476          0586  1
```

```
;  478          0587  1 ROUTINE cvt_long_nlt (lbn, cnt, entry) : NOVALUE =
;  479          0588  1 !++
;  480          0589  1 !
;  481          0590  1 ! Functional Description:
;  482          0591  1 !
;  483          0592  1 !    Convert a pair of long words containing a Logical Block Number and a
;  484          0593  1 !    count value into a non last track bad block entry.
;  485          0594  1 !
;  486          0595  1 ! Inputs:
;  487          0596  1 !
;  488          0597  1 !       lbn     val     the logical block number
;  489          0598  1 !
;  490          0599  1 !       cnt     val     the count of contigous bad blocks
;  491          0600  1 !
;  492          0601  1 ! Outputs:
;  493          0602  1 !
;  494          0603  1 !       entry   adr     the address of the longword to receive the value.
;  495          0604  1 !
;  496          0605  1 !--
;  497          0606  2 BEGIN
;  498          0607  2 MAP
;  499          0608  2     entry : REF $BBLOCK;
;  500          0609  2
;  501          0610  2 entry [nlt_w_lowlbn] = .lbn <0,16>;                  ! Format into a non last track entry.
;  502          0611  2 entry [nlt_b_highlbn] = .lbn <16, 8>;                ! 31      16!15 8!7   0
;  503          0612  2 entry [nlt_b_sectorcnt] = .cnt <0,8>;                ! +---------+---------+
;  504          0613  2                                                     ! ! low lbn !cnt !hlbn!
;  505          0614  2                                                     ! +---------+----+----+
;  506          0615  1 END;    ! of ROUTINE cvt_long_nlt


                                        0000 00000 CVT_LONG_NLT:
                                                   .WORD   Save nothing                          ; 0587
                        50      0C   AC  D0 00002          MOVL    ENTRY, R0                    ; 0610
                    02  A0      04   AC  B0 00006          MOVW    LBN, 2(R0)                   ;
                        60      06   AC  90 0000B          MOVB    LBN+2, (R0)                  ; 0611
                    01  A0      08   AC  90 0000F          MOVB    CNT, 1(R0)                   ; 0612
                                        04 00014          RET                                  ; 0615

; Routine Size:  21 bytes,    Routine Base:  $CODE$ + 01A6

;  507          0616  1
```

```
;  509          0617  1 %SBTTL  'insert_blocks -- Insert the specified bad blocks into the SDBSF'
;  510          0618  1 ROUTINE insert_blocks (entry, cnt) : NOVALUE =
;  511          0619  1 !++
;  512          0620  1 !
;  513          0621  1 ! Functional Description:
;  514          0622  1 !
;  515          0623  1 !    Insert the specified bad blocks into the SDBSF.  Call the appropriate
;  516          0624  1 !    conversion routines to convert the lbn into the required format.
;  517          0625  1 !
;  518          0626  1 ! Inputs:
;  519          0627  1 !
;  520          0628  1 !    entry       val     the block number to insert into the SDBSF.
;  521          0629  1 !
;  522          0630  1 !    cnt         val     the number of contiguous blocks to mark bad.
;  523          0631  1 !
;  524          0632  1 ! Side Effects:
;  525          0633  1 !
;  526          0634  1 !    The entry (block number) and count will be converted to the appropriate
;  527          0635  1 !    format before insertion into the SDBSF.  If the SDBSF becomes full during
;  528          0636  1 !    any stage of the analysis, we will STOP (via a SIGNAL_STOP) and inform
;  529          0637  1 !    the user of the problem.
;  530          0638  1 !
;  531          0639  1 !--
;  532          0640  2 BEGIN
;  533          0641  2 LOCAL
;  534          0642  2     new_entry;
;  535          0643  2
;  536          0644  2 IF .bad$gl_sdbsf_ptr GEQU .bad$ga_sdbsf + bad$k_page_size ! Have we reached the end of the bad block file?
;  537          0645  2 THEN                                                      ! If we did, inform the user that the medium or
;  538          0646  2     SIGNAL_STOP (bad$_bbfovf, 1, bad$ga_device);          ! device is potentially unreliable.
;  539          0647  2
;  540          0648  2 IF .bad$gl_context [ctx_v_ltdevice]                      ! Last track device?
;  541          0649  2 THEN
;  542          0650  2     cvt_log_phy (.entry, new_entry)                      ! Convert this block to a physical address.
;  543          0651  2 ELSE
;  544          0652  3     BEGIN
;  545          0653  3     cvt_long_nlt(.entry, .cnt, new_entry);               ! Convert the range to non last track entry.
;  546          0654  3     bad$ga_sdbsf [nlt_b_usedbbdsc] =
;  547          0655  3         .bad$ga_sdbsf [nlt_b_usedbbdsc] + 2;             ! Update used descriptor count.
;  548          0656  2     END;
;  549          0657  2
;  550          0658  2 .bad$gl_sdbsf_ptr = .new_entry;                         ! Append the new entry to the file.
;  551          0659  2 bad$gl_sdbsf_ptr = .bad$gl_sdbsf_ptr + 4;               ! Update the next free entry pointer.
;  552          0660  2
;  553          0661  1 END;    ! of ROUTINE insert_blocks


                                        0000 00000 INSERT_BLOCKS:
                                                   .WORD   Save nothing                                  ; 0618
                                5E              04  C2 00002          SUBL2   #4, SP                    ;
 50     0000G   CF 00000200            8F  C1 00005          ADDL3   #512, BAD$GA_SDBSF, R0                ; 0644
                50      0000G           CF  D1 0000F          CMPL    BAD$GL_SDBSF_PTR, R0                  ;
                                        13  1F 00014          BLSSU   1$                                    ;
                        0000G           CF  9F 00016          PUSHAB  BAD$GA_DEVICE                         ; 0646
```

```
                                       01  DD 0001A           PUSHL   #1                                      ;
                           00000000G  8F   DD 0001C           PUSHL   #BAD$_BBFOVF                            ;
              00000000G  00            03  FB 00022           CALLS   #3, LIB$STOP                            ;
                         0B     0000G  CF  E9 00029 1$:       BLBC    BAD$GL_CONTEXT+1, 2$                    ; 0648
                                       5E  DD 0002E           PUSHL   SP                                      ; 0650
                                   04  AC  DD 00030           PUSHL   ENTRY                                   ;
                     80  AF            02  FB 00033           CALLS   #2, CVT_LOG_PHY                         ;
                                       13  11 00037           BRB     3$                                      ;
                                       5E  DD 00039 2$:       PUSHL   SP                                      ; 0653
                     7E            04  AC  7D 0003B           MOVQ    ENTRY, -(SP)                            ;
                     A8  AF            03  FB 0003F           CALLS   #3, CVT_LONG_NLT                        ;
                         50     0000G  CF  D0 00043           MOVL    BAD$GA_SDBSF, R0                        ; 0654
                     02  A0            02  80 00048           ADDB2   #2, 2(R0)                               ; 0655
                  0000G  DF            6E  D0 0004C 3$:       MOVL    NEW_ENTRY, @BAD$GL_SDBSF_PTR            ; 0658
                  0000G  CF            04  C0 00051           ADDL2   #4, BAD$GL_SDBSF_PTR                    ; 0659
                                           04 00056           RET                                             ; 0661

; Routine Size:  87 bytes,    Routine Base:  $CODE$ + 01BB


;   554          0662  1
```

```
;  556          0663  1 %SBTTL  'lookup_bad -- Look up entry in bad block file(s)'
;  557          0664  1 ROUTINE lookup_bad (entry, buffer) =
;  558          0665  1 !++
;  559          0666  1 !
;  560          0667  1 ! Functional Description:
;  561          0668  1 !
;  562          0669  1 !    Look up the entry(ies) in the bad block buffer.
;  563          0670  1 !    If the entry exists, RETURN TRUE; ELSE FALSE.
;  564          0671  1 !
;  565          0672  1 !    The above is true also in the case of a non last track device,
;  566          0673  1 !    if the block already exists in the bad block file as implied
;  567          0674  1 !    within a bad block cluster descriptor.
;  568          0675  1 !
;  569          0676  1 ! Inputs:
;  570          0677  1 !
;  571          0678  1 !    entry      val     the block number to look up the in bad block file.
;  572          0679  1 !
;  573          0680  1 !    buffer     adr     the address of the bad block file to search for the
;  574          0681  1 !                       entry provided above
;  575          0682  1 !
;  576          0683  1 ! Routine Value:
;  577          0684  1 !
;  578          0685  1 !    TRUE       If the block is found within the given bad block file,
;  579          0686  1 !
;  580          0687  1 !    FALSE      otherwise.
;  581          0688  1 !
;  582          0689  1 !--
;  583          0690  2 BEGIN
;  584          0691  2 LOCAL
;  585          0692  2     match;
;  586          0693  2
;  587          0694  2 match = FALSE;
;  588          0695  2
;  589          0696  2 IF .bad$gl_context [ctx_v_ltdevice]                    ! Last track device?
;  590          0697  2 THEN
;  591          0698  3     BEGIN
;  592          0699  3     MAP
;  593          0700  3         buffer : REF VECTOR [, LONG];                  ! For ease of reference.
;  594          0701  3
;  595          0702  3     LOCAL
;  596          0703  3         ltentry;
;  597          0704  3
;  598          0705  3     cvt_log_phy (.entry, ltentry);                     ! Convert the LBN to lt fmt.
;  599          0706  3     INCR ptr FROM .buffer + ltk_k_headersiz             ! Examine each of the bad
;  600          0707  3              TO .buffer + bad$k_page_size / 4 - 2      ! blocks listed.
;  601          0708  3              BY 4
;  602          0709  3     DO
;  603          0710  4         BEGIN                                          ! Look for a match or the
;  604          0711  4         IF ..ptr EQL .ltentry                          ! end of the file.
;  605          0712  4         THEN
;  606          0713  5             BEGIN
;  607          0714  5             match = TRUE;                              ! We found a match.
;  608          0715  5             EXITLOOP;
;  609          0716  4             END;
;  610          0717  4
;  611          0718  4         IF ..ptr EQL -1
;  612          0719  4         THEN                                           ! No match found.
```

```
5                   BEGIN
5                   match = FALSE;                             ! End of file encountered.
5                   EXITLOOP;
4                   END;
3               END;
3           END
2   ELSE
3       BEGIN
3       MAP
3           buffer : REF $BBLOCK;
3
3       LOCAL
3           cnt,
3           lbn;
3
3       IF .buffer [nlt_b_usedbbdsc] NEQ 0                     ! Are there any bad blocks on file?
3       THEN
3           INCR ptr FROM .buffer + nlt_k_headersiz            ! Examine only the used bad block
3                    TO .buffer + nlt_k_headersiz +            ! descriptors.
4                      (.buffer [nlt_b_usedbbdsc] * 2 - 4)     ! The number of bad block descriptors
3                    BY 4                                      ! are always stored as a multiple of 2
3           DO                                                 ! (Compatibility for ODS-1 and ODS-2)
4               BEGIN
4               bad$cvt_nlt_long (..ptr, cnt, lbn);            ! Convert the descriptor into a
4               IF  .entry LEQU .lbn + .cnt                    ! starting logical block number
4               AND .entry GEQU .lbn                           ! and a count, then see if the
4               THEN                                           ! entry is within the described
5                   BEGIN                                      ! range of contiguous bad blocks.
5                   match = TRUE;                              ! Set match found and return.
5                   EXITLOOP;
4                   END;
3               END;
3       !
3       ! If there are no bad blocks on file or we did not match a block described
3       ! within the range specified in the bad block descriptor, then we will
3       ! return FALSE.
3       !
2       END;
2   RETURN .match;
1   END;    ! of ROUTINE lookup_bad


                                001C 00000 LOOKUP_BAD:
                                               .WORD   Save R2,R3,R4                  ; 0664
         5E              0C  C2 00002          SUBL2   #12, SP                        ;
                         53  D4 00005          CLRL    MATCH                          ; 0694
         52       08     AC  D0 00007          MOVL    BUFFER, R2                     ; 0706
         2E    0000G     CF  E9 0000B          BLBC    BAD$GL_CONTEXT+1, 3$           ; 0696
                         5E  DD 00010          PUSHL   SP                             ; 0705
                  04     AC  DD 00012          PUSHL   ENTRY                          ;
   FF46  CF              02  FB 00015          CALLS   #2, CVT_LOG_PHY                ;
         51       7E     A2  9E 0001A          MOVAB   126(R2), R1                    ; 0707
         50       04     A2  9E 0001E          MOVAB   4(R2), PTR                     ; 0711
                         12  11 00022          BRB     2$                             ;
```

```
                                 6E              60  D1 00024 1$:    CMPL    (PTR), LTENTRY                  ;
                                                 41  13 00027       BEQL    5$                              ;
                          FFFFFFFF   8F          60  D1 00029       CMPL    (PTR), #-1                      ; 0718
                                                 04  12 00030       BNEQ    2$                              ;
                                                 53  D4 00032       CLRL    MATCH                           ; 0721
                                                 3F  11 00034       BRB     7$                              ; 0720
            FFE8        50                   04  51  F1 00036 2$:    ACBL    R1, #4, PTR, 1$                 ; 0706
                                                 37  11 0003C       BRB     7$                              ; 0696
                                 50      02      A2  9A 0003E 3$:    MOVZBL  2(R2), R0                       ; 0735
                                                 31  13 00042       BEQL    7$                              ;
                                 54          6240  3E 00044       MOVAW   (R2)[R0], R4                    ; 0738
                                                 25  11 00048       BRB     6$                              ; 0744
                                         04      AE  9F 0004A 4$:    PUSHAB  LBN                             ; 0743
                                         0C      AE  9F 0004D       PUSHAB  CNT                             ;
                                                 62  DD 00050       PUSHL   (PTR)                           ;
                                      FED1   CF      03  FB 00052       CALLS   #3, BAD$CVT_NLT_LONG            ;
                                50        04   AE  08  AE  C1 00057       ADDL3   CNT, LBN, R0                    ; 0744
                                             50  04  AC  D1 0005D       CMPL    ENTRY, R0                       ;
                                                 0C  1A 00061       BGTRU   6$                              ;
                                          04     AE  04  AC  D1 00063       CMPL    ENTRY, LBN              ; 0745
                                                 05  1F 00068       BLSSU   6$                              ;
                                 53              01  D0 0006A 5$:    MOVL    #1, MATCH                       ; 0748
                                                 06  11 0006D       BRB     7$                              ; 0747
            FFD5        52                   04  54  F1 0006F 6$:    ACBL    R4, #4, PTR, 4$                 ; 0737
                                 50              53  D0 00075 7$:    MOVL    MATCH, R0                       ; 0758
                                                     04 00078       RET                                     ; 0759

; Routine Size:  121 bytes,    Routine Base:  $CODE$ + 0212

;  653          0760  1
```

```
;  655          0761  1 %SBTTL  'check_for_blk0 -- See if entry is block zero'
;  656          0762  1 ROUTINE check_for_blk0 (entry) : NOVALUE =
;  657          0763  1 !++
;  658          0764  1 !
;  659          0765  1 ! Functional Description:
;  660          0766  1 !
;  661          0767  1 !    See if the entry sought is block zero.  If it is, we set an
;  662          0768  1 !    internal flag in the context bits and record it in the SDBSF.
;  663          0769  1 !    The user is warned when we exit (both at SYS$ERROR and in the
;  664          0770  1 !    listing if one was requested) that block zero is bad and the
;  665          0771  1 !    medium should not be used as a system device.
;  666          0772  1 !
;  667          0773  1 ! Inputs:
;  668          0774  1 !
;  669          0775  1 !    entry      val     the block number to check.
;  670          0776  1 !
;  671          0777  1 ! Side Effects:
;  672          0778  1 !
;  673          0779  1 !    If the block number is zero (0), then the context bit (blk0bad)
;  674          0780  1 !    will be set true.
;  675          0781  1 !
;  676          0782  1 !--
;  677          0783  2 BEGIN
;  678          0784  2     MAP
;  679          0785  2         entry : REF VECTOR [, LONG];
;  680          0786  2
;  681          0787  2 IF ..entry EQL 0
;  682          0788  2 THEN
;  683          0789  3     BEGIN
;  684          0790  3     bad$gl_context [ctx_v_blk0bad] = TRUE;              ! Flag block 0 is bad!
;  685          0791  2     END;
;  686          0792  1 END;    ! of ROUTINE check_for_blk0


                                        0000 00000 CHECK_FOR_BLK0:
                                                   .WORD   Save nothing                                    ; 0762
                                    04  BC  D5 00002       TSTL    @ENTRY                                  ; 0787
                                        05  12 00005       BNEQ    1$                                      ;
                                0000G   CF      02  88 00007       BISB2   #2, BAD$GL_CONTEXT              ; 0790
                                                04 0000C 1$:       RET                                     ; 0792

; Routine Size:  13 bytes,    Routine Base:  $CODE$ + 028B

;  687          0793  1
```

```
;  689          0794  1 %SBTTL  'bad$init_buffers -- Initialize the contents of the data buffers'
;  690          0795  1 GLOBAL ROUTINE  bad$init_buffers (pattern) : NOVALUE =
;  691          0796  1 !++
;  692          0797  1 !
;  693          0798  1 ! Functional Description:
;  694          0799  1 !
;  695          0800  1 !    Initialize the contents of the data buffers with the given test pattern.
;  696          0801  1 !
;  697          0802  1 ! Inputs:
;  698          0803  1 !
;  699          0804  1 !    pattern    val     the test pattern used to initialize the contents
;  700          0805  1 !                       of the buffers.
;  701          0806  1 !
;  702          0807  1 ! Side Effects:
;  703          0808  1 !
;  704          0809  1 !    The two data buffers and the test pattern buffer will have their contents
;  705          0810  1 !    initialized to the pattern specified.
;  706          0811  1 !
;  707          0812  1 !--
;  708          0813  2 BEGIN
;  709          0814  2 BUILTIN
;  710          0815  2     ACTUALCOUNT;
;  711          0816  2
;  712          0817  2 BIND
;  713          0818  2     buf0 = .bad$ga_tpb          : VECTOR [, LONG],
;  714          0819  2     buf1 = .bad$ga_bufadr [0]   : VECTOR [, LONG],
;  715          0820  2     buf2 = .bad$ga_bufadr [1]   : VECTOR [, LONG];
;  716          0821  2
;  717          0822  2 LOCAL
;  718          0823  2     tpb_boundary,
;  719          0824  2     buf_boundary;
;  720          0825  2
;  721          0826  2 tpb_boundary = bad$k_page_size / 4 - 1;                  ! Compute the test pattern buffer
;  722          0827  2 buf_boundary = .bad$gl_trnsfr_cnt / 4 - 1;                ! and the data buffer's boundaries.
;  723          0828  2
;  724          0829  2 IF ACTUALCOUNT () NEQ 0
;  725          0830  2 THEN                                                     ! Use the supplied test pattern.
;  726          0831  3     BEGIN
;  727          0832  3     INCR loc FROM 0 TO .tpb_boundary
;  728          0833  3     DO buf0 [.loc] = .pattern;                          ! Fill the test pattern buffer.
;  729          0834  3
;  730          0835  3     INCR loc FROM 0 TO .buf_boundary
;  731          0836  3     DO buf1 [.loc] = buf2 [.loc] = .pattern;            ! Fill the data buffers.
;  732          0837  3     END
;  733          0838  2 ELSE                                                     ! Use the pattern supplied in the data vector.
;  734          0839  3     BEGIN
;  735          0840  3     INCR loc FROM 0 TO .tpb_boundary                    ! Fill the test pattern buffer,
;  736          0841  3     BY .bad$gb_term_count                               ! compute the offset to the next location.
;  737          0842  3     DO  INCR offset FROM 0 TO .bad$gb_term_count - 1
;  738          0843  3         DO  IF (.loc + .offset) LEQ .tpb_boundary       ! Be sure we're still within the buffer's boundary.
;  739          0844  3             THEN buf0 [.loc + .offset] = .bad$gl_bad_term [.offset]
;  740          0845  3             ELSE EXITLOOP;                              ! We're done.
;  741          0846  3
;  742          0847  3     INCR loc FROM 0 TO .buf_boundary                    ! Fill the data buffers.
;  743          0848  3     BY .bad$gb_term_count                               ! Compute the offset.
;  744          0849  3     DO  INCR offset FROM 0 TO .bad$gb_term_count - 1    ! And move the pattern.
;  745          0850  3         DO  IF (.loc + .offset) LEQ .buf_boundary       ! Be sure we're still within the buffers' boundaries
```

```
;  746          0851  3              THEN buf1 [.loc + .offset] = buf2 [.loc + .offset] = .bad$gl_bad_term [.offset]
;  747          0852  3              ELSE EXITLOOP;
;  748          0853  2      END;
;  749          0854  2
;  750          0855  1 END;     ! of GLOBAL ROUTINE bad$init_buffers.



                                                007C 00000          .ENTRY  BAD$INIT_BUFFERS, Save R2,R3,R4,R5,R6          ; 0795
                                     54      7F 8F  9A 00002        MOVZBL  #127, TPB_BOUNDARY                             ; 0826
                              51     0000G   CF     04  C7 00006    DIVL3   #4, BAD$GL_TRNSFR_CNT, R1                      ; 0827
                                                    51  D7 0000C    DECL    BUF_BOUNDARY
                                                    6C  95 0000E    TSTB    (AP)                                           ; 0829
                                                    2A  13 00010    BEQL    5$
                                             50                     01  CE 00012    MNEGL   #1, LOC                        ; 0833
                                                    07  11 00015    BRB     2$
                                          0000GDF4   0      04      AC  D0 00017 1$:  MOVL    PATTERN, @BAD$GA_TPB[LOC]
                              F5                 50                 54  F3 0001E 2$:  AOBLEQ  TPB_BOUNDARY, LOC, 1$
                                                 50                 01  CE 00022    MNEGL   #1, LOC                        ; 0835
                                                    10  11 00025    BRB     4$
                                             52      04      AC     D0 00027 3$:  MOVL    PATTERN, R2                      ; 0836
                                          0000GDF4                  52  D0 0002B    MOVL    R2, @BAD$GA_BUFADR+4[LOC]
                                          0000GDF4                  52  D0 00031    MOVL    R2, @BAD$GA_BUFADR[LOC]
                              EC                 50                 51  F3 00037 4$:  AOBLEQ  BUF_BOUNDARY, LOC, 3$
                                                                    04  0003B    RET                                            ; 0829
                                             55      0000G   CF     9A 0003C 5$:  MOVZBL  BAD$GB_TERM_COUNT, R5            ; 0841
                                             56        FF    A5     9E 00041    MOVAB   -1(R5), R6                         ; 0842
                                                               50   D4 00045    CLRL    LOC
                                                               1E   11 00047    BRB     10$
                                                 52                01  CE 00049 6$:  MNEGL   #1, OFFSET                     ; 0843
                                                                   12  11 0004C    BRB     8$
                              53                 50                52  C1 0004E 7$:  ADDL3   OFFSET, LOC, R3
                                                 54                53  D1 00052    CMPL    R3, TPB_BOUNDARY
                                                                   0D  14 00055    BGTR    9$
                                          0000GDF4   0000GCF4      42  D0 00057    MOVL    BAD$GL_BAD_TERM[OFFSET], @BAD$GA_TPB[R3]  ; 0844
                              EA                 52                56  F3 00060 8$:  AOBLEQ  R6, OFFSET, 7$                ; 0843
                                                 50                55  C0 00064 9$:  ADDL2   R5, LOC                        ; 0842
                                                 54                50  D1 00067 10$: CMPL    LOC, TPB_BOUNDARY
                                                                   DD  15 0006A    BLEQ    6$
                                                                   50  D4 0006C    CLRL    LOC                            ; 0848
                                                                   27  11 0006E    BRB     15$
                                                 53                01  CE 00070 11$: MNEGL   #1, OFFSET                     ; 0850
                                                                   1B  11 00073    BRB     13$
                              52                 50                53  C1 00075 12$: ADDL3   OFFSET, LOC, R2
                                                 51                52  D1 00079    CMPL    R2, BUF_BOUNDARY
                                                                   16  14 0007C    BGTR    14$
                                                 54   0000GCF4     43  D0 0007E    MOVL    BAD$GL_BAD_TERM[OFFSET], R4    ; 0851
                                          0000GDF4                  54  D0 00084    MOVL    R4, @BAD$GA_BUFADR+4[R2]
                                          0000GDF4                  54  D0 0008A    MOVL    R4, @BAD$GA_BUFADR[R2]
                              E1                 53                56  F3 00090 13$: AOBLEQ  R6, OFFSET, 12$               ; 0850
                                                 50                55  C0 00094 14$: ADDL2   R5, LOC                        ; 0849
                                                 51                50  D1 00097 15$: CMPL    LOC, BUF_BOUNDARY
                                                                   D4  15 0009A    BLEQ    11$
                                                                       04 0009C    RET                                    ; 0855
```

```
; Routine Size:  157 bytes,    Routine Base:  $CODE$ + 0298

;  751          0856  1
;  752          0857  1 END     ! of MODULE badblocks
;  753          0858  0 ELUDOM



                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                                PSECT SUMMARY
;
;      Name                      Bytes                         Attributes
;
;  $PLIT$                          32  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                         821  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         3         0      1000          00:01.4



;                                    COMMAND QUALIFIERS

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

; Size:          821 code + 32 data bytes
; Run Time:          00:12.0
; Elapsed Time:      00:35.4
; Lines/CPU Min:      4286
; Lexemes/CPU-Min: 15107
; Memory Used:  111 pages
; Compilation Complete
```

BADBLOCKS
LIS

BAD

STABAD
MAP

BADDATA
LIS

ANALYZBAD
MAP

ANALYZFMD
CLD

BADDEF
SDL

WRITESAVE
LIS

BADINIT
LIS

ANALYZCMD
LIS