```
BBBBBBBBBBBB          AAAAAAAAA      DDDDDDDDDDD
BBBBBBBBBBBBB         AAAAAAAAA      DDDDDDDDDDDD
BBBBBBBBBBBBB         AAAAAAAAA      DDDDDDDDDDDD
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
BBBBBBBBBBBBB       AAA           AAA   DDD           DDD
BBBBBBBBBBBBB       AAA           AAA   DDD           DDD
BBBBBBBBBBBBB       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
BBBBBBBBBBBBB       AAA           AAA   DDDDDDDDDDDD
BBBBBBBBBBBBB       AAA           AAA   DDDDDDDDDDDD
```

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

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

```
;     1        0001  0 MODULE badinit( %TITLE 'Analyze/Media Initialization Module'
;     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 used to parse the command string
;    38        0038  1 !       and allocate/initialize the control and storage structures which define
;    39        0039  1 !       the device's analysis context.
;    40        0040  1 !
;    41        0041  1 ! Environment:
;    42        0042  1 !
;    43        0043  1 !       VAX/VMS User Mode, Non-Privileged
;    44        0044  1 !
;    45        0045  1 ! Author:
;    46        0046  1 !
;    47        0047  1 !       Michael T. Rhodes,              Creation Date: July, 1982
;    48        0048  1 !
;    49        0049  1 ! Modified By:
;    50        0050  1 !
;    51        0051  1 !       V03-004 MTR0009         Michael T. Rhodes       3-Jul-1984
;    52        0052  1 !               Fix processing of /SHOW=BEFORE qualifier to obtain
;    53        0053  1 !               the correct SDBSF and generation of listings.
;    54        0054  1 !
;    55        0055  1 !       V03-003 MTR0003         Michael T. Rhodes      11-Feb-1983
;    56        0056  1 !               Add support for Stand Alone BAD.  Routine 'alloc_mem'
;    57        0057  1 !               has been changed to be a GLOBAL routine, renamed to
```

```
;    58          0058  1 !                 'bad$alloc_mem'.
;    59          0059  1 !
;    60          0060  1 !       V03-002 MTR0002         Michael T. Rhodes       20-Jan-1983
;    61          0061  1 !               Use shared message BADVALUE instead of INVALNUM.
;    62          0062  1 !
;    63          0063  1 !       V03-001 MTR0001         Michael T. Rhodes       15-Dec-1982
;    64          0064  1 !               Add the /LOG qualifier and the keyword PATTERN to the
;    65          0065  1 !               /EXERCISE qualifier.  Several ROUTINEs have been added
;    66          0066  1 !               to support these features:
;    67          0067  1 !                   'bad_value', 'get_pattern', and 'parse_value'
;    68          0068  1 !
;    69          0069  1 !--
;    70          0070  1
```

```
;    72          0071  1 %SBTTL 'Declarations'
;    73          0072  1 !
;    74          0073  1 ! Include Files:
;    75          0074  1 !
;    76          0075  1 REQUIRE 'lib$:baddef';                          ! Define BAD's structures.
;    77          0195  1 LIBRARY 'SYS$LIBRARY:LIB';                      ! Define VMS structures.
;    78          0196  1 LIBRARY 'SYS$LIBRARY:TPAMAC';                   ! TPARSE macro definitions.
;    79          0197  1
;    80          0198  1 !
;    81          0199  1 ! Table of Contents:
;    82          0200  1 !
;    83          0201  1 FORWARD ROUTINE
;    84          0202  1     bad$alloc_mem       : NOVALUE,              ! Standard memory getter!
;    85          0203  1     bad$init            : NOVALUE,              ! Process the command line qualifiers and init struc
;    86          0204  1     addr_error,                                 ! Handle address range errors.
;    87          0205  1     bad_value,                                  ! Bad numeric value error routine.
;    88          0206  1     count_term,                                 ! Number of terms in expression.
;    89          0207  1     get_mdbsf           : NOVALUE,              ! Get a valid copy of the MDBSF.
;    90          0208  1     get_nlt_sdbsf,                              ! Get a valid copy of the SDBSF from a non last trac
;    91          0209  1     get_pattern         : NOVALUE,              ! Process user supplied test pattern(s).
;    92          0210  1     get_sdbsf           : NOVALUE,              ! Get a valid copy of the SDBSF.
;    93          0211  1     init_sdbsf          : NOVALUE,              ! Initialize the header of the SDBSF.
;    94          0212  1     parse_value         : NOVALUE,              ! Parse and process bad block info.
;    95          0213  1     process_badblocks,                          ! Process user supplied bad block information.
;    96          0214  1     rebuild_dbsfs       : NOVALUE,              ! Rebuild the MDBSF and SDBSF.
;    97          0215  1     recover_dbsfs,                              ! Attempt to rebuild/recover an SDBSF.
;    98          0216  1     syntax_err,                                 ! Error action routine for TPARSE.
;    99          0217  1     val_log_adr,                                ! Validate the user supllied logical address.
;   100          0218  1     val_phy_adr;                                ! Validate the user supplied physical address.
;   101          0219  1
;   102          0220  1 !
;   103          0221  1 ! Private Storage
;   104          0222  1 !
;   105          0223  1 BIND                                            ! The list below contains the
;   106          0224  1     after =     $DESCRIPTOR ('AFTER'),          ! strings used to parse the
;   107          0225  1     bad_blocks =$DESCRIPTOR ('BAD_BLOCKS'),     ! command qualifiers and keywords.
;   108          0226  1     before =    $DESCRIPTOR ('BEFORE'),         ! Also, some of the strings are
;   109          0227  1     command_lin=$DESCRIPTOR ('$LINE'),          ! used in the various messages
;   110          0228  1     count =     $DESCRIPTOR ('COUNT'),          ! issued from the utility.
;   111          0229  1     cylinder =  $DESCRIPTOR ('CYLINDER'),
;   112          0230  1     device =    $DESCRIPTOR ('DEVICE'),
;   113          0231  1     exercise =  $DESCRIPTOR ('EXERCISE'),
;   114          0232  1     full =      $DESCRIPTOR ('FULL'),
;   115          0233  1     keep =      $DESCRIPTOR ('KEEP'),
;   116          0234  1     log =       $DESCRIPTOR ('LOG'),
;   117          0235  1     logical_blk=$DESCRIPTOR ('LOGICAL_BLOCK'),
;   118          0236  1     mdbsf =     $DESCRIPTOR ('MDBSF'),
;   119          0237  1     output =    $DESCRIPTOR ('OUTPUT'),
;   120          0238  1     pattern =   $DESCRIPTOR ('PATTERN'),
;   121          0239  1     prompt =    $DESCRIPTOR ('BAD_BLOCKS = '),
;   122          0240  1     retry =     $DESCRIPTOR ('RETRY'),
;   123          0241  1     sdbsf =     $DESCRIPTOR ('SDBSF'),
;   124          0242  1     sector =    $DESCRIPTOR ('SECTOR'),
;   125          0243  1     show_qual = $DESCRIPTOR ('SHOW'),
;   126          0244  1     track =     $DESCRIPTOR ('TRACK');
;   127          0245  1
;   128          0246  1 !
```

```
;  129     0247  1 ! External references:
;  130     0248  1 !
;  131     0249  1 EXTERNAL ROUTINE
;  132     0250  1     bad$check_ltk       : ADDRESSING_MODE (GENERAL),    ! Check last track device for a user supplied entry.
;  133     0251  1     bad$check_nlt       : ADDRESSING_MODE (GENERAL),    ! Check non last track device for a user supplied en
;  134     0252  1     bad$cvt_ltk_long    : ADDRESSING_MODE (GENERAL),    ! Convert last track format to longwords.
;  135     0253  1     bad$cvt_nlt_long    : ADDRESSING_MODE (GENERAL),    ! Convert non last track format to longwords.
;  136     0254  1     bad$cvt_phy_log     : ADDRESSING_MODE (GENERAL),    ! Convert physical address to logical.
;  137     0255  1     bad$prepare_report  : ADDRESSING_MODE (GENERAL),    ! Open the output file and generate report heading.
;  138     0256  1     bad$produce_report  : ADDRESSING_MODE (GENERAL),    ! Produce a listing of the bad block file(s).
;  139     0257  1     bad$sta_init        : WEAK,                         ! Perform initialization for execution in the Stand
;  140     0258  1     bad$str_trim        : WEAK,                         ! Trim trailing blanks, tabs, and carriage return fr
;  141     0259  1     bad$sync_io         : ADDRESSING_MODE (GENERAL),    ! Perform synchronous IO.
;  142     0260  1     bad$validate_pack   : WEAK,                         ! Set Volume Valid bit in the UCB when executing Sta
;  143     0261  1     CHECKSUM2           : ADDRESSING_MODE (GENERAL),    ! Compute block checksum.
;  144     0262  1     CLI$GET_VALUE       : ADDRESSING_MODE (GENERAL),    ! CLI call back routine.
;  145     0263  1     CLI$PRESENT         : ADDRESSING_MODE (GENERAL),    ! CLI call back routine.
;  146     0264  1     LIB$GET_INPUT       : ADDRESSING_MODE (GENERAL),    ! Library input routine.
;  147     0265  1     LIB$TPARSE          : ADDRESSING_MODE (GENERAL);    ! Library State Table Parser.
;  148     0266  1
;  149     0267  1 EXTERNAL
;  150     0268  1     bad$gl_bad_term     : VECTOR [4, LONG],             ! Bad block information vector.
;  151     0269  1     bad$gb_block_fact   : BYTE,                         ! Number of sectors per block.
;  152     0270  1     bad$gl_bytes_cyl,                                   ! Total number of bytes per cylinder.
;  153     0271  1     bad$gl_bytes_trk,                                   ! Total number of bytes per track.
;  154     0272  1     bad$gl_chan,                                        ! Channel number for device.
;  155     0273  1     bad$ga_comnd_line   : $BBLOCK [dsc$c_s_bln],        ! Address of command line buffer descriptor.
;  156     0274  1     bad$gl_context      : BITVECTOR [32],               ! Global control context flags.
;  157     0275  1     bad$gl_cylinders,                                   ! Number of cylinders on medium.
;  158     0276  1     bad$ga_bufadr       : VECTOR [2, LONG],             ! Address vector of the data transfer buffers.
;  159     0277  1     bad$gl_devchar      : $BBLOCK,                      ! Device characteristics.
;  160     0278  1     bad$gl_devclass,                                    ! Class of the device.
;  161     0279  1     bad$ga_device       : $BBLOCK [dsc$c_s_bln],        ! Address of device name descriptor.
;  162     0280  1     bad$ga_devnam       : $BBLOCK [64],                 ! Device name buffer.
;  163     0281  1     bad$gl_devnam,                                      ! Size of device name.
;  164     0282  1     bad$gq_devnam       : $BBLOCK [dsc$c_s_bln],        ! Device name descriptor.
;  165     0283  1     bad$ga_filespec     : $BBLOCK [dsc$c_s_bln],        ! Address of filespec buffer descriptor.
;  166     0284  1     bad$gl_func,                                        ! IO function code for $QIO.
;  167     0285  1     bad$ga_getdvi       : $BBLOCK,                      ! Address of the $GETDVI item list.
;  168     0286  1     bad$ga_input_desc   : $BBLOCK [dsc$c_s_bln],        ! Generic input descriptor.
;  169     0287  1     bad$gq_iosb         : VECTOR [4, LONG],             ! IO status blocks.
;  170     0288  1     bad$gl_maxblock,                                    ! Highest user addressable LBN on device.
;  171     0289  1     bad$ga_mdbsf        : REF $BBLOCK,                  ! Address of the Manufacturers Detected Bad Sector F
;  172     0290  1     bad$gl_mdbsf_ptr,                                   ! Disk address of the MDBSF.
;  173     0291  1     bad$gl_pagcnt,                                      ! Number of 512 byte pages for 1 transfer.
;  174     0292  1     bad$ga_sdbsf        : REF $BBLOCK,                  ! Address of the Software Detected Bad Sector File.
;  175     0293  1     bad$gl_sdbsf_ptr,                                   ! Pointer to the first available entry in the SDBSF.
;  176     0294  1     bad$gl_sectors,                                     ! Number of sectors per track.
;  177     0295  1     bad$gl_sector_siz,                                  ! Size of the physical sector on a drive.
;  178     0296  1     bad$gl_serialnum,                                   ! Pack serial number.
;  179     0297  1     bad$gl_status,                                      ! Global status.
;  180     0298  1     bad$gb_term_count   : BYTE,                         ! Number of terms used to describe bad block.
;  181     0299  1     bad$ga_tpb,                                         ! Address of the Test Pattern Buffer.
;  182     0300  1     bad$gl_tracks,                                      ! Number of tracks per cylinder.
;  183     0301  1     bad$gl_trnsfr_cnt;                                  ! Number of bytes per transfer.
;  184     0302  1
;  185     0303  1 !
```

```
;  186          0304  1 ! Define message codes...
;  187          0305  1 !
;  188          0306  1 EXTERNAL LITERAL
;  189          0307  1     bad$_aligndisk,                 ! The cartridge is an alignment disk.
;  190          0308  1     bad$_assign,                    ! Failed to assign a channel.
;  191          0309  1     bad$_badvalue,                  ! Invalid keyword value.
;  192          0310  1     bad$_devnotblk,                 ! The device is not block structured.
;  193          0311  1     bad$_devnotfor,                 ! The device is not mounted foreign.
;  194          0312  1     bad$_devrct,                    ! Device has revector caching enabled.
;  195          0313  1     bad$_getdvi,                    ! $GETDVI failed.
;  196          0314  1     bad$_insvirmem,                 ! Insufficient virtual memory.
;  197          0315  1     bad$_invalqual,                 ! Invalid qualifier.
;  198          0316  1     bad$_ivblknum,                  ! Invalid block number.
;  199          0317  1     bad$_ivblkent,                  ! Misiing or invalid block entity.
;  200          0318  1     bad$_lststring,                 ! Used to list an ascid string.
;  201          0319  1     bad$_mdbsfcrupt,                ! MDBSF is corrupt.
;  202          0320  1     bad$_mdbsfrfail,                ! Failed to read MDBSF.
;  203          0321  1     bad$_nobadinfo,                 ! No bad block info.
;  204          0322  1     bad$_nooutqual,                 ! No /OUTPUT qualifier.
;  205          0323  1     bad$_rebldwarn,                 ! Failed to write enough new MDBSF and SDBSF files.
;  206          0324  1     bad$_sdbsffull,                 ! SDBSF is full.
;  207          0325  1     bad$_sdbsfrfail;                ! Failed to read the SDBSF.
;  208          0326  1
;  209          0327  1 !
;  210          0328  1 ! Macros:
;  211          0329  1 !
;  212          0330  1 MACRO
;  213        M 0331  1     def_tparse_args =
;  214        M 0332  1         BUILTIN AP;
;  215        M 0333  1         BIND tparse_args = AP : REF $BBLOCK;
;  216          0334  1         %;
;  217          0335  1
```

```
;  219     0336  1 %SBTTL  'State table definitions'
;  220     0337  1 !++
;  221     0338  1 !
;  222     0339  1 ! Functional Description:
;  223     0340  1 !
;  224     0341  1 !       The state table provides the definition of legal grammar for the
;  225     0342  1 !       acceptance of bad block information.  The legal forms of bad block
;  226     0343  1 !       information are:
;  227     0344  1 !
;  228     0345  1 !               1.  lbn
;  229     0346  1 !
;  230     0347  1 !               2.  lbn:count
;  231     0348  1 !
;  232     0349  1 !               3.  sector.track.cylinder
;  233     0350  1 !
;  234     0351  1 !               4.  sector.track.cylinder:count
;  235     0352  1 !
;  236     0353  1 !       Each of the elements may be specified in any of the 3 valid radices,
;  237     0354  1 !       OCTAL, DECIMAL, or HEXADECIMAL.  Note that ANY of the elements may be
;  238     0355  1 !       of ANY radix, no restriction is imposed on the user requiring that the
;  239     0356  1 !       same radix be used for all terms.  The default radix is DECIMAL.
;  240     0357  1 !
;  241     0358  1 !       A simple action routine "count_term" is called by TPARSE on our
;  242     0359  1 !       behalf.  It increments a count of the number of terms processed and
;  243     0360  1 !       stores the values taken from the TPARSE control block at the location
;  244     0361  1 !       indicated by "bad$gl_bad_term [.bad$gb_term_count]".  The count will
;  245     0362  1 !       later be used to convert the bad block information into either logical
;  246     0363  1 !       block number or in physical address form (plus a count value if included).
;  247     0364  1 !
;  248     0365  1 !       An additional state table is also defined, which provides a simple
;  249     0366  1 !       parsing mechanism for numeric values.  This is used by for parsing the
;  250     0367  1 !       keyword PATTERN's value(s).
;  251     0368  1 !
;  252     0369  1 !--
;  253     0370  1
;  254     0371  1
;  255     0372  1 $INIT_STATE (state_tbl, key_tbl);
;  256     0373  1
;  257   P 0374  1 $STATE  (,
;  258   P 0375  1          ((number),, count_term),
;  259     0376  1          ((syntax)));
;  260     0377  1
;  261   P 0378  1 $STATE  (,
;  262   P 0379  1          ('.', track_number),
;  263   P 0380  1          (TPA$_LAMBDA)
;  264     0381  1          );
;  265     0382  1
;  266   P 0383  1 $STATE  (colon,
;  267   P 0384  1          (':', blkcnt),
;  268   P 0385  1          (TPA$_LAMBDA)
;  269     0386  1          );
;  270     0387  1
;  271   P 0388  1 $STATE  (blkend,
;  272   P 0389  1          (TPA$_EOS, TPA$_EXIT),
;  273     0390  1          ((syntax)));
;  274     0391  1
;  275   P 0392  1 $STATE  (blkcnt,
```

```
;  276        P 0393  1          ((number), blkend, count_term),
;  277          0394  1          ((syntax)));
;  278          0395  1
;  279        P 0396  1 $STATE   (track_number,
;  280        P 0397  1          ((number),, count_term),
;  281          0398  1          ((syntax)));
;  282          0399  1
;  283        P 0400  1 $STATE   (,
;  284        P 0401  1          ('.', cylinder_number),
;  285          0402  1          ((syntax)));
;  286          0403  1
;  287        P 0404  1 $STATE   (cylinder_number,
;  288        P 0405  1          ((number), colon, count_term),
;  289          0406  1          ((syntax)));
;  290          0407  1 !
;  291          0408  1 ! The following portion of the state table is shared to allow the parsing
;  292          0409  1 ! of integer numbers of various radices.
;  293          0410  1 !
;  294          0411  1 $INIT_STATE (number_tbl, numkey_tbl);
;  295          0412  1
;  296        P 0413  1 $STATE   (,
;  297        P 0414  1          ((number),, count_term),
;  298          0415  1          ((bad_val)));
;  299          0416  1
;  300        P 0417  1 $STATE   (number,
;  301        P 0418  1          (TPA$_DECIMAL, TPA$_EXIT),
;  302          0419  1          ('%'));
;  303          0420  1
;  304        P 0421  1 $STATE   (,
;  305        P 0422  1          ('d', decnum),
;  306        P 0423  1          ('D', decnum),
;  307        P 0424  1          ('x', hexnum),
;  308        P 0425  1          ('X', hexnum),
;  309        P 0426  1          ('o', octnum),
;  310          0427  1          ('O', octnum));
;  311          0428  1
;  312        P 0429  1 $STATE   (octnum,
;  313          0430  1          (TPA$_OCTAL, TPA$_EXIT));
;  314          0431  1
;  315        P 0432  1 $STATE   (decnum,
;  316          0433  1          (TPA$_DECIMAL, TPA$_EXIT));
;  317          0434  1
;  318        P 0435  1 $STATE   (hexnum,
;  319          0436  1          (TPA$_HEX, TPA$_EXIT));
;  320          0437  1
;  321        P 0438  1 $STATE   (bad_val,
;  322        P 0439  1          (TPA$_SYMBOL,,  bad_value),
;  323        P 0440  1          (TPA$_STRING,,  bad_value),
;  324        P 0441  1          (TPA$_LAMBDA,,  bad_value),
;  325        P 0442  1          (TPA$_ANY,,     bad_value),
;  326        P 0443  1          (TPA$_EOS,,     bad_value)
;  327          0444  1          );
;  328          0445  1
;  329        P 0446  1 $STATE   (syntax,
;  330        P 0447  1          (TPA$_SYMBOL,,  syntax_err),
;  331        P 0448  1          (TPA$_STRING,,  syntax_err),
;  332        P 0449  1          (TPA$_LAMBDA,,  syntax_err),
```

```
;  333          P 0450  1          (TPA$_ANY,,     syntax_err),
;  334          P 0451  1          (TPA$_EOS,,     syntax_err)
;  335            0452  1          );
;  336            0453  1
```

```
;   338         0454  1 %SBTTL 'bad$init -- Initialization Procedure'
;   339         0455  1
;   340         0456  1 GLOBAL ROUTINE bad$init : NOVALUE =
;   341         0457  1 !++
;   342         0458  1 !
;   343         0459  1 ! Functional Description:
;   344         0460  1 !
;   345         0461  1 !    This procedure processes the qualifiers specified/defaulted in the command
;   346         0462  1 !    line.  The device is validated and an IO channel established.  If an output
;   347         0463  1 !    file is requested, it is opened at this time.
;   348         0464  1 !
;   349         0465  1 ! Calling Sequence:
;   350         0466  1 !
;   351         0467  1 !    bad$init ();                                   ! Parse the command line and
;   352         0468  1 !                                                   ! process/initialize structures.
;   353         0469  1 !
;   354         0470  1 ! Implicit Inputs:
;   355         0471  1 !
;   356         0472  1 !    Pointers to control and data structures.
;   357         0473  1 !
;   358         0474  1 ! Implicit Outputs:
;   359         0475  1 !
;   360         0476  1 !    All of the global control and data structures have been allocated and
;   361         0477  1 !    initialized.
;   362         0478  1 !
;   363         0479  1 !--
;   364         0480  2 BEGIN
;   365         0481  2
;   366         0482  2 bad$gl_context [ctx_v_init] = TRUE;                 ! Flag initialization phase.
;   367         0483  2
;   368         0484  2 !
;   369         0485  2 ! Initialize descriptors.
;   370         0486  2 !
;   371         0487  2 CH$FILL (0, dsc$c_s_bln, bad$ga_comnd_line);
;   372         0488  2 CH$FILL (0, dsc$c_s_bln, bad$ga_device);
;   373         0489  2 CH$FILL (0, dsc$c_s_bln, bad$ga_input_desc);
;   374         0490  2 CH$FILL (0, dsc$c_s_bln, bad$ga_filespec);
;   375         0491  2 bad$ga_comnd_line [dsc$b_class] = dsc$k_class_d;    ! Command line descriptor.
;   376         0492  2 bad$ga_device [dsc$b_class] = dsc$k_class_d;        ! Device name descriptor.
;   377         0493  2 bad$ga_input_desc [dsc$b_class] = dsc$k_class_d;    ! Input line descriptor.
;   378         0494  2 bad$ga_filespec [dsc$b_class] = dsc$k_class_d;      ! Output file spec descriptor.
;   379         0495  2
;   380         0496  2 IF bad$sta_init NEQ 0                              ! Are we executing Stand Alone or under VMS?
;   381         0497  2 THEN    bad$sta_init ();                           ! Stand Alone -- perform required initialization.
;   382         0498  2
;   383         0499  2 !
;   384         0500  2 ! Save the command line and get the device spec.
;   385         0501  2 !
;   386         0502  2 CLI$GET_VALUE (command_lin, bad$ga_comnd_line);    ! Get a local copy of the command line.
;   387         0503  2 IF bad$str_trim NEQ 0                              ! Trim the command line if we are stand alone.
;   388         0504  2 THEN    bad$str_trim (bad$ga_comnd_line);
;   389         0505  2
;   390         0506  2 CLI$GET_VALUE (device, bad$ga_device);             ! Copy the device name.
;   391         0507  2 IF bad$str_trim NEQ 0                              ! Trim the device specification if we are stand alon
;   392         0508  2 THEN    bad$str_trim (bad$ga_device);
;   393         0509  2
;   394         0510  3 IF NOT (bad$gl_status =
```

```
;  395        P 0511  3          $GETDVIW (DEVNAM = bad$ga_device,                    ! Get the device specifics.
;  396          0512  3                    ITMLST = bad$ga_getdvi))
;  397          0513  2 THEN                                                          ! If not obtainable -- STOP!
;  398          0514  2     SIGNAL_STOP (bad$_getdvi, 1, bad$ga_device, .bad$gl_status);
;  399          0515  2
;  400          0516  2 IF .bad$gl_devclass NEQ dc$_disk                            ! Is this a valid device type?
;  401          0517  2 THEN                                                        ! If not -- STOP!
;  402          0518  2     SIGNAL_STOP (bad$_devnotblk, 1, bad$ga_device);
;  403          0519  2
;  404          0520  2 IF .bad$gl_devchar [dev$v_rct]                              ! Revector cached device?
;  405          0521  2 THEN                                                        ! If so, STOP, don't waste time
;  406          0522  2     SIGNAL_STOP (bad$_devrct, 1, bad$ga_device);            ! essentially doing a NOP!
;  407          0523  2
;  408          0524  2 IF NOT .bad$gl_devchar [dev$v_for]                          ! Device must be mounted /FOREIGN
;  409          0525  2 AND bad$sta_init EQL 0                                      ! (while executing under VMS),
;  410          0526  2 THEN                                                        ! because we do not have enough
;  411          0527  2     SIGNAL_STOP (bad$_devnotfor, 1, bad$ga_device);         ! access privilege otherwise.
;  412          0528  2
;  413          0529  3 IF NOT (bad$gl_status =
;  414        P 0530  3          $ASSIGN (DEVNAM = bad$ga_device,                    ! If no IO channel, we must STOP!
;  415          0531  3                   CHAN = bad$gl_chan))
;  416          0532  2 THEN
;  417          0533  2     SIGNAL_STOP (bad$_assign, 1, bad$ga_device, .bad$gl_status);
;  418          0534  2
;  419          0535  2 IF bad$validate_pack NEQ 0                                  ! Are we executing Stand Alone?
;  420          0536  2 THEN    bad$validate_pack ();                               ! Yes, set the Volume Valid bit in the UCB.
;  421          0537  2
;  422          0538  2 bad$gq_devnam [dsc$w_length] = .bad$gl_devnam;              ! Build the device name descriptor.
;  423          0539  2 bad$gq_devnam [dsc$b_dtype] = dsc$k_dtype_t;                ! Data type is text.
;  424          0540  2 bad$gq_devnam [dsc$b_class] = dsc$k_class_d;                ! Dynamic descriptor.
;  425          0541  2 bad$gq_devnam [dsc$a_pointer] = bad$ga_devnam;              ! Buffer address.
;  426          0542  2
;  427          0543  2 IF .bad$gl_maxblock GTR bad$k_nltmaxblk                    ! Is this a last track device?
;  428          0544  2 THEN
;  429          0545  3     BEGIN
;  430          0546  3     bad$gl_context [ctx_v_ltdevice] = TRUE;                 ! Default the device type to "last track"
;  431          0547  4     bad$gb_block_fact = (.bad$gl_tracks *                   ! Compute blocking factor.
;  432          0548  4                          .bad$gl_cylinders *
;  433          0549  3                          .bad$gl_sectors) /
;  434          0550  3                          .bad$gl_maxblock;
;  435          0551  3     bad$gl_mdbsf_ptr = .bad$gl_maxblock -                   ! Set beginning address of last track.
;  436          0552  3                        .bad$gl_sectors /
;  437          0553  3                        .bad$gb_block_fact;
;  438          0554  3     bad$gl_sector_siz = bad$k_page_size / .bad$gb_block_fact;   ! Compute the sector size (in bytes).
;  439          0555  3     bad$gl_bytes_trk = .bad$gl_sectors * .bad$gl_sector_siz;    ! Compute number of bytes per track.
;  440          0556  3     bad$gl_bytes_cyl = .bad$gl_bytes_trk * .bad$gl_tracks;      ! Compute number of bytes per cylinder.
;  441          0557  3     bad$gl_trnsfr_cnt = MINU (.bad$gl_bytes_cyl,                ! Select largest transfer value.
;  442          0558  3                               bad$k_max_xfer_cnt);
;  443          0559  3     END
;  444          0560  2 ELSE
;  445          0561  2     bad$gl_trnsfr_cnt = bad$k_max_xfer_cnt;                 ! Non last track device, make large transfers.
;  446          0562  2
;  447          0563  2 bad$gl_pagcnt = .bad$gl_trnsfr_cnt / bad$k_page_size;     ! Compute memory allocation size in pages.
```

```
;  449          0564  2 %SBTTL    'Process qualifiers'
;  450          0565  2 !++
;  451          0566  2 !
;  452          0567  2 ! Process qualifiers.
;  453          0568  2 !
;  454          0569  2 !    This portion of the procedure sets/clears the various bits in the
;  455          0570  2 !    context bits according to the presence of the various qualifiers
;  456          0571  2 !    and keywords.
;  457          0572  2 !
;  458          0573  2 !    The actual processing of the qualifiers/keywords will be performed
;  459          0574  2 !    selectively below.
;  460          0575  2 !
;  461          0576  2 !--
;  462          0577  2 bad$gl_context [ctx_v_badblocks] =      CLI$PRESENT (bad_blocks);       ! /BAD_BLOCKS
;  463          0578  2 bad$gl_context [ctx_v_exercise] =       CLI$PRESENT (exercise);         ! /EXERCISE
;  464          0579  2 bad$gl_context [ctx_v_keep] =           CLI$PRESENT (keep);             ! /[NO]EXERCISE=KEEP
;  465          0580  2 bad$gl_context [ctx_v_log] =            CLI$PRESENT (log);              ! /LOG
;  466          0581  2 bad$gl_context [ctx_v_output] =         CLI$PRESENT (output);           ! /OUTPUT
;  467          0582  2 bad$gl_context [ctx_v_retry] =          CLI$PRESENT (retry);            ! /RETRY
;  468          0583  2 bad$gl_context [ctx_v_show] =           CLI$PRESENT (show_qual);        ! /SHOW
;  469          0584  2
;  470          0585  2 !
;  471          0586  2 ! END of generic qualifier/keyword processing...
;  472          0587  2 !
;  473          0588  2
;  474          0589  2 !
;  475          0590  2 !    If the device is a last track device, then allocate the MDBSF and SDBSF
;  476          0591  2 !    buffers and attempt to obtain the MDBSF.  If the device is a non last
;  477          0592  2 !    track device, simply allocate the SDBSF buffer.
;  478          0593  2 !
;  479          0594  2 !    (The SDBSF is only preserved if so directed by the KEEP keyword)
;  480          0595  2 !
;  481          0596  2 bad$gl_func = IO$_READLBLK;                                  ! Set default IO function code.
;  482          0597  2 IF .bad$gl_context [ctx_v_ltdevice]
;  483          0598  2 THEN
;  484          0599  3     BEGIN
;  485          0600  3     bad$alloc_mem (bad$k_page_size * 2, bad$ga_mdbsf);  ! Allocate the detected bad sector file buffers.
;  486          0601  3     bad$ga_sdbsf = .bad$ga_mdbsf + bad$k_page_size;     ! Seperate the buffers.
;  487          0602  3     get_mdbsf ();                                       ! Obtain a valid copy of the MDBSF.
;  488          0603  3     END
;  489          0604  2 ELSE
;  490          0605  2     bad$alloc_mem (bad$k_page_size, bad$ga_sdbsf);      ! Allocate a single buffer.
;  491          0606  2
;  492          0607  2 !
;  493          0608  2 !  *                                                    *
;  494          0609  2 !*** Process individual qualifiers for specific actions. ***
;  495          0610  2 !  *                                                    *
;  496          0611  2 !
;  497          0612  2 SELECT TRUE OF SET
;  498          0613  2
```

```
;  500          0614  2 %SBTTL    'Process /OUTPUT qualifier'
;  501          0615  2 !++
;  502          0616  2 !
;  503          0617  2 ! Process /OUTPUT qualifier.
;  504          0618  2 !
;  505          0619  2 !    If requested get the file specified (or defaulted) by the user and
;  506          0620  2 !    the output file.  The default specification of /SHOW=AFTER is also
;  507          0621  2 !    set (just in case it wasn't specified in the command line).
;  508          0622  2 !
;  509          0623  2 !    The default is /NOOUTPUT.
;  510          0624  2 !
;  511          0625  2 !--
;  512          0626  2 [.bad$gl_context [ctx_v_output]]:
;  513          0627  3     BEGIN
;  514          0628  3     CLI$GET_VALUE (output, bad$ga_filespec);            ! Get the filespec if supplied.
;  515          0629  3     bad$prepare_report ();                              ! Open the file.
;  516          0630  3     bad$gl_context [ctx_v_show_after] = TRUE;           ! Default to /SHOW=AFTER if it is not supplied.
;  517          0631  3
;  518          0632  2     END;    ! of /OUTPUT processing
;  519          0633  2
```

```
;  521          0634  2 %SBTTL  'Process /SHOW qualifier'
;  522          0635  2 !++
;  523          0636  2 !
;  524          0637  2 ! Process /SHOW qualifier.
;  525          0638  2 !
;  526          0639  2 !    This procedure processes the /SHOW qualifier which will produce the
;  527          0640  2 !    following actions:
;  528          0641  2 !
;  529          0642  2 !       1.  If /SHOW=BEFORE is specified the listing is produced from the
;  530          0643  2 !           the current bad block file(s) as they exist in our buffers.
;  531          0644  2 !           The AFTER qualifier value context switch will be toggled by a
;  532          0645  2 !           call to CLI$PRESENT.
;  533          0646  2 !
;  534          0647  2 !       2.  If /SHOW or /SHOW=AFTER is specified the the context bit
;  535          0648  2 !           "show_after" will be set, resulting in the contents of the bad
;  536          0649  2 !           block file(s) being listed at the conclusion of processing.
;  537          0650  2 !
;  538          0651  2 !--
;  539          0652  2 [.bad$gl_context [ctx_v_show]]:
;  540          0653  3     BEGIN                                                  ! If /SHOW is specified without /OUTPUT,
;  541          0654  3     IF NOT .bad$gl_context [ctx_v_output]                  ! then -- STOP!  We do not have a valid
;  542          0655  3     THEN                                                   ! output file to write the report to!
;  543          0656  3         SIGNAL_STOP (bad$_nooutqual, 0, bad$_lststring, 1, bad$ga_comnd_line);
;  544          0657  3
;  545          0658  4     IF (bad$gl_context [ctx_v_show_before] = CLI$PRESENT (before))
;  546          0659  3     THEN                                                   ! Should we produce a report before
;  547          0660  4         BEGIN                                              ! testing the media?  If so, then
;  548          0661  4         IF .bad$gl_context [ctx_v_ltdevice]                ! obtain a copy of the old SDBSF,
;  549          0662  4         THEN    get_sdbsf ()                               ! produce the report, and toggle the
;  550          0663  4         ELSE    bad$gl_context [ctx_v_nlt_sdbsf] = get_nlt_sdbsf (); ! show_after context flag.
;  551          0664  4         IF .bad$gl_context [ctx_v_ltdevice] OR .bad$gl_context [ctx_v_nlt_sdbsf]
;  552          0665  4         THEN    bad$produce_report ();
;  553          0666  4         bad$gl_context [ctx_v_show_after] = CLI$PRESENT (after);
;  554          0667  4         END
;  555          0668  3     ELSE
;  556          0669  3         bad$gl_context [ctx_v_show_after] = TRUE;          ! Otherwise the default is show_after.
;  557          0670  3
;  558          0671  2     END;     ! of /SHOW processing
;  559          0672  2
```

```
;  561          0673  2 %SBTTL  'Process the KEEP keyword'
;  562          0674  2 !++
;  563          0675  2 !
;  564          0676  2 ! Process the KEEP keyword ** ALWAYS PERFORMED **.
;  565          0677  2 !
;  566          0678  2 !    This portion of the procedure will determine whether the contents of the
;  567          0679  2 !    old SDBSF should be preserved.
;  568          0680  2 !
;  569          0681  2 !    The default is dependent on the EXERCISE context.  If the context is
;  570          0682  2 !    to EXERCISE the device, the default is NOT to preserve the old SDBSF
;  571          0683  2 !    contents.  This allows the user to purge the allocation of user supplied
;  572          0684  2 !    bad blocks from the SDBSF.  IF the context is NOEXERCISE, then KEEP is
;  573          0685  2 !    implicit (we must protect and preserve the contents of the SDBSF, since
;  574          0686  2 !    it would be too easy to blow away the SDBSF in this mode), resulting in a
;  575          0687  2 !    potentially useless piece of media (at least until it was analyzed again).
;  576          0688  2 !
;  577          0689  2 !--
;  578          0690  2 [ALWAYS]:
;  579          0691  2
;  580          0692  3     BEGIN
;  581          0693  3     IF .bad$gl_context [ctx_v_keep] OR                  ! Should we preserve the old SDBSF?
;  582          0694  4     (NOT .bad$gl_context [ctx_v_exercise] AND NOT .bad$gl_context [ctx_v_show_before])
;  583          0695  3     THEN
;  584          0696  3         IF .bad$gl_context [ctx_v_ltdevice]
;  585          0697  3         THEN    get_sdbsf ()                            ! Yes, obtain a valid copy of the SDBSF.
;  586          0698  3         ELSE    get_nlt_sdbsf ()                        ! Yes, obtain a valid copy of the non last track SDB
;  587          0699  3     ELSE
;  588          0700  3         init_sdbsf ();                                  ! No, create a new copy of the SDBSF.
;  589          0701  3
;  590          0702  2     END;    ! of KEEP keyword processing
;  591          0703  2
```

```
;  593          0704  2 %SBTTL  'Process the /BAD_BLOCKS qualifier'
;  594          0705  2 !++
;  595          0706  2 !
;  596          0707  2 ! Process the /BAD_BLOCKS qualifier.
;  597          0708  2 !
;  598          0709  2 !    This qualifier may have an associated set of bad block information.
;  599          0710  2 !    Any combination of the valid bad block information may be specified
;  600          0711  2 !    (in a comma seperated list, utilizing any combination of radix directives).
;  601          0712  2 !    The default radix is DECIMAL.
;  602          0713  2 !
;  603          0714  2 !    Bad block information formats:
;  604          0715  2 !
;  605          0716  2 !       none                        - If no input is specified with the /BAD_BLOCKS
;  606          0717  2 !                                     qualifier, the utility will prompt the user
;  607          0718  2 !                                     for the required information.
;  608          0719  2 !
;  609          0720  2 !       lbn                         - Allocate a single block to the bad block file.
;  610          0721  2 !
;  611          0722  2 !       lbn:count                   - Allocate a range of contiguous blocks starting
;  612          0723  2 !                                     at lbn to the bad block file.
;  613          0724  2 !
;  614          0725  2 !       sec.trk.cyl                 - Allocate a single physical sector to the bad
;  615          0726  2 !                                     block file.
;  616          0727  2 !
;  617          0728  2 !       sec.trk.cyl:count           - Allocate a range of contigous physical sectors
;  618          0729  2 !                                     to the bad block file.
;  619          0730  2 !
;  620          0731  2 !    Depending on whether this is a last track or non last track device the bad
;  621          0732  2 !    block information will be converted to the required format and entered into
;  622          0733  2 !    the SDBSF, if it does not already exist in either the MDBSF or the SDBSF.
;  623          0734  2 !
;  624          0735  2 !--
;  625          0736  2 [.bad$gl_context [ctx_v_badblocks]]:
;  626          0737  3     BEGIN
;  627          0738  3     LOCAL
;  628          0739  3         value_present;
;  629          0740  3
;  630          0741  3     value_present = FALSE;
;  631          0742  3     WHILE bad$gl_status = CLI$GET_VALUE (bad_blocks, bad$ga_input_desc) DO
;  632          0743  4         BEGIN                                           ! Bad blocks have been specified
;  633          0744  4         value_present = TRUE;                           ! in the command line.
;  634          0745  4         process_badblocks ();                           ! Process the current bad block set.
;  635          0746  3         END;
;  636          0747  3
;  637          0748  3     IF NOT .value_present                               ! Should we prompt for the bad blocks?
;  638          0749  3     THEN                                                ! Yes.
;  639          0750  4         BEGIN
;  640          0751  4         bad$gl_context [ctx_v_interactive] = TRUE;      ! Allow duplicate block messages to be displayed.
;  641          0752  4         WHILE bad$gl_status = LIB$GET_INPUT (bad$ga_input_desc, prompt)
;  642          0753  4         DO
;  643          0754  5             BEGIN                                       ! Read a bad block set from the user.
;  644          0755  5             IF .bad$gl_status EQL RMS$_EOF              ! Has he terminated the input?
;  645          0756  5             THEN EXITLOOP;                              ! Yes -- move on to the next phase.
;  646          0757  5             process_badblocks ();                       ! No, process the bad block set.
;  647          0758  4             END;
;  648          0759  3         END;
;  649          0760  3     bad$gl_context [ctx_v_interactive] = FALSE;         ! Disallow duplicate block messages beyond here.
```

```
;  650          0761  3
;  651          0762  2      END;  ! of /BAD_BLOCKS processing
;  652          0763  2
```

```
;  654     0764  2 %SBTTL    'Process /EXERCISE qualifier'
;  655     0765  2 !++
;  656     0766  2 !
;  657     0767  2 ! Process /EXERCISE qualifier.
;  658     0768  2 !
;  659     0769  2 !    If the device is supposed to be analyzed, then allocate the data
;  660     0770  2 !    and test pattern buffers.
;  661     0771  2 !
;  662     0772  2 !    Check to see if the user requested a completion log of the number of
;  663     0773  2 !    bad blocks detected.                                                    ;
;  664     0774  2 !
;  665     0775  2 !    If the user supplied a test pattern, then obtain a copy of it (upto
;  666     0776  2 !    an OCTAWORD (128 bits) of pattern), which will be stored in the vector  ;
;  667     0777  2 !    'bad$gl_bad_term'.
;  668     0778  2 !
;  669     0779  2 !       ***************************************************************
;  670     0780  2 !       ***                          N O T E                        ***
;  671     0781  2 !       *** This MUST be the last operation performed on the vector before ***
;  672     0782  2 !       *** returning to the driver, since the routine 'bad$init_buffers'  ***
;  673     0783  2 !       *** expects the vector to contain the user supplied test pattern!  ***
;  674     0784  2 !       ***************************************************************
;  675     0785  2 !
;  676     0786  2 !    The default is /NOEXERCISE.
;  677     0787  2 !
;  678     0788  2 !--
;  679     0789  2
;  680     0790  2 [.bad$gl_context [ctx_v_exercise]]:
;  681     0791  3     BEGIN
;  682     0792  3     bad$gl_context [ctx_v_exercise_full] = CLI$PRESENT (full);  ! Set/Clear the FULL context bit.
;  683     0793  3     bad$gl_conte't [ctx_v_nlt_sdbsf] = true;                    ! A new/updated SDBSF will be written to the
;  684     0794  3
;  685     0795  4     IF (bad$gl_context [ctx_v_pattern] = CLI$PRESENT (pattern)) ! Set/Clear the PATTERN context bit.
;  686     0796  3     THEN
;  687     0797  3         get_pattern ();                                         ! Process the user supplied test pattern(s).
;  688     0798  3
;  689     0799  3     bad$alloc_mem (.bad$gl_pagcnt * bad$k_page_size * 2,
;  690     0800  3                    bad$ga_bufadr [0]);                          ! Allocate the data buffers.
;  691     0801  3
;  692     0802  3     bad$ga_bufadr [1] = .bad$ga_bufadr [0] +                    ! Seperate them into distinct entities.
;  693     0803  3                         .bad$gl_pagcnt *
;  694     0804  3                         bad$k_page_size;
;  695     0805  3
;  696     0806  3     bad$alloc_mem (bad$k_page_size, bad$ga_tpb);                ! Allocate the Test Pattern Buffer.
;  697     0807  3
;  698     0808  2     END;  !of /EXERCISE processing
;  699     0809  2
;  700     0810  2 TES;    ! of Specific Qualifier processing.
;  701     0811  2
;  702     0812  2 bad$gl_context [ctx_v_init] = FALSE;                ! Flag end of initialization phase.
;  703     0813  2
;  704     0814  1 END;    ! Of GLOBAL ROUTINE bad$init


                                                        .TITLE  BADINIT Analyze/Media Initialization Module
                                                        .IDENT  \V04-000\

                                                        .PSECT  _LIB$STATE$,NOWRT,  SHR,  PIC,1
```

```
                        00000 STATE_TBL::
                                      .BLKB   0
                    89F8  00000 ;TPA$TYPE
                                U.2:  .WORD   -30216                                    ;
                    0000* 00002 ;TPA$SUBEXP
                                U.4:  .WORD   <<U.3-U.4>-2>                             ;
                00000000V 00004 ;TPA$ACTION
                                U.5:  .LONG   <<COUNT_TERM-U.5>-4>                      ;
                    0DF8  00008 ;TPA$TYPE
                                U.6:  .WORD   3576                                      ;
                    0000* 0000A ;TPA$SUBEXP
                                U.8:  .WORD   <<U.7-U.8>-2>                             ;
                    102E  0000C ;TPA$TYPE
                                U.9:  .WORD   4142                                      ;
                    0000* 0000E ;TPA$TARGET
                                U.11: .WORD   <<U.10-U.11>-2>                           ;
                    05F6  00010 ;TPA$TYPE
                                U.12: .WORD   1526                                      ;
                          00012 COLON:  .BLKB   0
                    103A  00012 ;TPA$TYPE
                                U.13: .WORD   4154                                      ;
                    0000* 00014 ;TPA$TARGET
                                U.15: .WORD   <<U.14-U.15>-2>                           ;
                    05F6  00016 ;TPA$TYPE
                                U.16: .WORD   1526                                      ;
                          00018 BLKEND: .BLKB   0
                    11F7  00018 ;TPA$TYPE
                                U.17: .WORD   4599                                      ;
                    FFFF  0001A ;TPA$TARGET
                                U.18: .WORD   -1                                        ;
                    0DF8  0001C ;TPA$TYPE
                                U.19: .WORD   3576                                      ;
                    0000* 0001E ;TPA$SUBEXP
                                U.20: .WORD   <<U.7-U.20>-2>                            ;
                          00020 ;BLKCNT
                                U.14: .BLKB   0
                    99F8  00020 ;TPA$TYPE
                                U.21: .WORD   -26120                                    ;
                    0000* 00022 ;TPA$SUBEXP
                                U.22: .WORD   <<U.3-U.22>-2>                            ;
                00000000V 00024 ;TPA$ACTION
                                U.23: .LONG   <<COUNT_TERM-U.23>-4>                     ;
                    0000* 00028 ;TPA$TARGET
                                U.24: .WORD   <<BLKEND-U.24>-2>                         ;
                    0DF8  0002A ;TPA$TYPE
                                U.25: .WORD   3576                                      ;
                    0000* 0002C ;TPA$SUBEXP
                                U.26: .WORD   <<U.7-U.26>-2>                            ;
                          0002E ;TRACK_NUMBER
                                U.10: .BLKB   0
                    89F8  0002E ;TPA$TYPE
                                U.27: .WORD   -30216                                    ;
                    00C0* 00030 ;TPA$SUBEXP
                                U.28: .WORD   <<U.3-U.28>-2>                            ;
                00000000V 00032 ;TPA$ACTION
                                U.29: .LONG   <<COUNT_TERM-U.29>-4>                     ;
```

```
                    0DF8  00036 ;TPA$TYPE
                                U.30:    .WORD   3576                                    ;
                    0000* 00038 ;TPA$SUBEXP
                                U.31:    .WORD   <<U.7-U.31>-2>                          ;
                    102E  0003A ;TPA$TYPE
                                U.32:    .WORD   4142                                    ;
                    0000* 0003C ;TPA$TARGET
                                U.34:    .WORD   <<U.33-U.34>-2>                         ;
                    0DF8  0003E ;TPA$TYPE
                                U.35:    .WORD   3576                                    ;
                    0000* 00040 ;TPA$SUBEXP
                                U.36:    .WORD   <<U.7-U.36>-2>                          ;
                          00042 ;CYLINDER_NUMBER
                                U.33:    .BLKB   0
                    99F8  00042 ;TPA$TYPE
                                U.37:    .WORD   -26120                                  ;
                    0000* 00044 ;TPA$SUBEXP
                                U.38:    .WORD   <<U.3-U.38>-2>                          ;
                00000000V 00046 ;TPA$ACTION
                                U.39:    .LONG   <<COUNT_TERM-U.39>-4>                   ;
                    0000* 0004A ;TPA$TARGET
                                U.40:    .WORD   <<COLON-U.40>-2>                        ;
                    0DF8  0004C ;TPA$TYPE
                                U.41:    .WORD   3576                                    ;
                    0000* 0004E ;TPA$SUBEXP
                                U.42:    .WORD   <<U.7-U.42>-2>                          ;
                          00050 NUMBER_TBL::
                                .BLKB   0
                    89F8  00050 ;TPA$TYPE
                                U.44:    .WORD   -30216                                  ;
                    0000* 00052 ;TPA$SUBEXP
                                U.45:    .WORD   <<U.3-U.45>-2>                          ;
                00000000V 00054 ;TPA$ACTION
                                U.46:    .LONG   <<COUNT_TERM-U.46>-4>                   ;
                    0DF8  00058 ;TPA$TYPE
                                U.47:    .WORD   3576                                    ;
                    0000* 0005A ;TPA$SUBEXP
                                U.49:    .WORD   <<U.48-U.49>-2>                         ;
                          0005C ;NUMBER
                                U.3:     .BLKB   0
                    11F3  0005C ;TPA$TYPE
                                U.50:    .WORD   4595                                    ;
                    FFFF  0005E ;TPA$TARGET
                                U.51:    .WORD   -1                                      ;
                    0425  00060 ;TPA$TYPE
                                U.52:    .WORD   1061                                    ;
                    1064  00062 ;TPA$TYPE
                                U.53:    .WORD   4196                                    ;
                    0000* 00064 ;TPA$TARGET
                                U.55:    .WORD   <<U.54-U.55>-2>                         ;
                    1044  00066 ;TPA$TYPE
                                U.56:    .WORD   4164                                    ;
                    0000* 00068 ;TPA$TARGET
                                U.57:    .WORD   <<U.54-U.57>-2>                         ;
                    1078  0006A ;TPA$TYPE
                                U.58:    .WORD   4216                                    ;
                    0000* 0006C ;TPA$TARGET
```

```
                        U.60:   .WORD   <<U.59-U.60>-2>                 ;
1058    0006E           ;TPA$TYPE
                        U.61:   .WORD   4184                            ;
0000*   00070           ;TPA$TARGET
                        U.62:   .WORD   <<U.59-U.62>-2>                 ;
106F    00072           ;TPA$TYPE
                        U.63:   .WORD   4207                            ;
0000*   00074           ;TPA$TARGET
                        U.65:   .WORD   <<U.64-U.65>-2>                 ;
144F    00076           ;TPA$TYPE
                        U.66:   .WORD   5199                            ;
0000*   00078           ;TPA$TARGET
                        U.67:   .WORD   <<U.64-U.67>-2>                 ;
        0007A           ;OCTNUM
                        U.64:   .BLKB   0
15F4    0007A           ;TPA$TYPE
                        U.68:   .WORD   5620                            ;
FFFF    0007C           ;TPA$TARGET
                        U.69:   .WORD   -1                              ;
        0007E           ;DECNUM
                        U.54:   .BLKB   0
15F3    0007E           ;TPA$TYPE
                        U.70:   .WORD   5619                            ;
FFFF    00080           ;TPA$TARGET
                        U.71:   .WORD   -1                              ;
        00082           ;HEXNUM
                        U.59:   .BLKB   0
15F5    00082           ;TPA$TYPE
                        U.72:   .WORD   5621                            ;
FFFF    00084           ;TPA$TARGET
                        U.73:   .WORD   -1                              ;
        00086           ;BAD_VAL
                        U.48:   .BLKB   0
81F1    00086           ;TPA$TYPE
                        U.74:   .WORD   -32271                          ;
00000000V 00088         ;TPA$ACTION
                        U.75:   .LONG   <<BAD_VALUE-U.75>-4>            ;
81F0    0008C           ;TPA$TYPE
                        U.76:   .WORD   -32272                          ;
00000000V 0008E         ;TPA$ACTION
                        U.77:   .LONG   <<BAD_VALUE-U.77>-4>            ;
81F6    00092           ;TPA$TYPE
                        U.78:   .WORD   -32266                          ;
00000000V 00094         ;TPA$ACTION
                        U.79:   .LONG   <<BAD_VALUE-U.79>-4>            ;
81ED    00098           ;TPA$TYPE
                        U.80:   .WORD   -32275                          ;
00000000V 0009A         ;TPA$ACTION
                        U.81:   .LONG   <<BAD_VALUE-U.81>-4>            ;
85F7    0009E           ;TPA$TYPE
                        U.82:   .WORD   -31241                          ;
00000000V 000A0         ;TPA$ACTION
                        U.83:   .LONG   <<BAD_VALUE-U.83>-4>            ;
        000A4           ;SYNTAX
                        U.7:    .BLKB   0
81F1    000A4           ;TPA$TYPE
                        U.84:   .WORD   -32271                          ;
```

```
                00000000V 000A6 ;TPA$ACTION
                          U.85:    .LONG   <<SYNTAX_ERR-U.85>-4>                          ;
                    81F0  000AA ;TPA$TYPE
                          U.86:    .WORD   -32272                                         ;
                00000000V 000AC ;TPA$ACTION
                          U.87:    .LONG   <<SYNTAX_ERR-U.87>-4>                          ;
                    81F6  000B0 ;TPA$TYPE
                          U.88:    .WORD   -32266                                         ;
                00000000V 000B2 ;TPA$ACTION
                          U.89:    .LONG   <<SYNTAX_ERR-U.89>-4>                          ;
                    81ED  000B6 ;TPA$TYPE
                          U.90:    .WORD   -32275                                         ;
                00000000V 000B8 ;TPA$ACTION
                          U.91:    .LONG   <<SYNTAX_ERR-U.91>-4>                          ;
                    85F7  000BC ;TPA$TYPE
                          U.92:    .WORD   -31241                                         ;
                00000000V 000BE ;TPA$ACTION
                          U.93:    .LONG   <<SYNTAX_ERR-U.93>-4>                          ;

                                   .PSECT  _LIB$KEY0$,NOWRT,  SHR,  PIC,1

                          00000 KEY_TBL::
                                   .BLKB   0
                          00000 ;TPA$KEY0
                          U.1:     .BLKB   0
                          00000 NUMKEY_TBL::
                                   .BLKB   0
                          00000 ;TPA$KEY0
                          U.43:    .BLKB   0

                                   .PSECT  $PLIT$,NOWRT,NOEXE,2

                      52  45  54  46  41  00000 P.AAB:   .ASCII  \AFTER\                  ;
                                  00005            .BLKB   3
                            00000005  00008 P.AAA:   .LONG   5                           ;
                            00000000' 0000C          .ADDRESS P.AAB                     ;
53  4B  43  4F  4C  42  5F  44  41  42  00010 P.AAD:   .ASCII  \BAD_BLOCKS\              ;
                                  0001A            .BLKB   2
                            0000000A  0001C P.AAC:   .LONG   10                          ;
                            00000000' 00020          .ADDRESS P.AAD                     ;
                          45  52  4F  46  45  42  00024 P.AAF:   .ASCII  \BEFORE\         ;
                                  0002A            .BLKB   2
                            00000006  0002C P.AAE:   .LONG   6                           ;
                            00000000' 00030          .ADDRESS P.AAF                     ;
                              45  4E  49  4C  24  00034 P.AAH:   .ASCII  \$LINE\         ;
                                  00039            .BLKB   3
                            00000005  0003C P.AAG:   .LONG   5                           ;
                            00000000' 00040          .ADDRESS P.AAH                     ;
                              54  4E  55  4F  43  00044 P.AAJ:   .ASCII  \COUNT\         ;
                                  00049            .BLKB   3
                            00000005  0004C P.AAI:   .LONG   5                           ;
                            00000000' 00050          .ADDRESS P.AAJ                     ;
                    52  45  44  4E  49  4C  59  43  00054 P.AAL:   .ASCII  \CYLINDER\     ;
                            00000008  0005C P.AAK:   .LONG   8                           ;
                            00000000' 00060          .ADDRESS P.AAL                     ;
                          45  43  49  56  45  44  00064 P.AAN:   .ASCII  \DEVICE\         ;
                                  0006A            .BLKB   2
```

```
                                                00000006  0006C P.AAM:  .LONG   6
                                                00000000' 00070         .ADDRESS P.AAN
                                45  53  49  43  52  45  58  45  00074 P.AAP:  .ASCII  \EXERCISE\
                                                00000008  0007C P.AAO:  .LONG   8
                                                00000000' 00080         .ADDRESS P.AAP
                                            4C  4C  55  46  00084 P.AAR:  .ASCII  \FULL\
                                                00000004  00088 P.AAQ:  .LONG   4
                                                00000000' 0008C         .ADDRESS P.AAR
                                            50  45  45  4B  00090 P.AAT:  .ASCII  \KEEP\
                                                00000004  00094 P.AAS:  .LONG   4
                                                00000000' 00098         .ADDRESS P.AAT
                                                47  4F  4C  0009C P.AAV:  .ASCII  \LOG\
                                                            0009F         .BLKB   1
                                                00000003  000A0 P.AAU:  .LONG   3
                                                00000000' 000A4         .ADDRESS P.AAV
    4B  43  4F  4C  42  5F  4C  41  43  49  47  4F  4C  000A8 P.AAX:  .ASCII  \LOGICAL_BLOCK\
                                                            000B5         .BLKB   3
                                                0000000D  000B8 P.AAW:  .LONG   13
                                                00000000' 000BC         .ADDRESS P.AAX
                                        46  53  42  44  4D  000C0 P.AAZ:  .ASCII  \MDBSF\
                                                            000C5         .BLKB   3
                                                00000005  000C8 P.AAY:  .LONG   5
                                                00000000' 000CC         .ADDRESS P.AAZ
                                    54  55  50  54  55  4F  000D0 P.ABB:  .ASCII  \OUTPUT\
                                                            000D6         .BLKB   2
                                                00000006  000D8 P.ABA:  .LONG   6
                                                00000000' 000DC         .ADDRESS P.ABB
                                4E  52  45  54  54  41  50  000E0 P.ABD:  .ASCII  \PATTERN\
                                                            000E7         .BLKB   1
                                                00000007  000E8 P.ABC:  .LONG   7
                                                00000000' 000EC         .ADDRESS P.ABD
20  3D  20  53  4B  43  4F  4C  42  5F  44  41  42  000F0 P.ABF:  .ASCII  \BAD_BLOCKS = \
                                                            000FD         .BLKB   3
                                                0000000D  00100 P.ABE:  .LONG   13
                                                00000000' 00104         .ADDRESS P.ABF
                                        59  52  54  45  52  00108 P.ABH:  .ASCII  \RETRY\
                                                            0010D         .BLKB   3
                                                00000005  00110 P.ABG:  .LONG   5
                                                00000000' 00114         .ADDRESS P.ABH
                                        46  53  42  44  53  00118 P.ABJ:  .ASCII  \SDBSF\
                                                            0011D         .BLKB   3
                                                00000005  00120 P.ABI:  .LONG   5
                                                00000000' 00124         .ADDRESS P.ABJ
                                52  4F  54  43  45  53  00128 P.ABL:  .ASCII  \SECTOR\
                                                            0012E         .BLKB   2
                                                00000006  00130 P.ABK:  .LONG   6
                                                00000000' 00134         .ADDRESS P.ABL
                                            57  4F  48  53  00138 P.ABN:  .ASCII  \SHOW\
                                                00000004  0013C P.ABM:  .LONG   4
                                                00000000' 00140         .ADDRESS P.ABN
                                        4B  43  41  52  54  00144 P.ABP:  .ASCII  \TRACK\
                                                            00149         .BLKB   3
                                                00000005  0014C P.ABO:  .LONG   5
                                                00000000' 00150         .ADDRESS P.ABP

                                                AFTER=                  P.AAA
                                                BAD_BLOCKS=             P.AAC
```

```
BEFORE=                 P.AAE
COMMAND_LIN=            P.AAG
COUNT=                  P.AAI
CYLINDER=               P.AAK
DEVICE=                 P.AAM
EXERCISE=               P.AAO
FULL=                   P.AAQ
KEEP=                   P.AAS
LOG=                    P.AAU
LOGICAL_BLK=            P.AAW
MDBSF=                  P.AAY
OUTPUT=                 P.ABA
PATTERN=                P.ABC
PROMPT=                 P.ABE
RETRY=                  P.ABG
SDBSF=                  P.ABI
SECTOR=                 P.ABK
SHOW_QUAL=              P.ABM
TRACK=                  P.ABO
        .EXTRN  BAD$CHECK_LTK, BAD$CHECK_NLT
        .EXTRN  BAD$CVT_LTK_LONG
        .EXTRN  BAD$CVT_NLT_LONG
        .EXTRN  BAD$CVT_PHY_LOG
        .EXTRN  BAD$PREPARE_REPORT
        .EXTRN  BAD$PRODUCE_REPORT
        .EXTRN  BAD$SYNC_IO, CHECKSUM2
        .EXTRN  CLI$GET_VALUE, CLI$PRESENT
        .EXTRN  LIB$GET_INPUT, LIB$TPARSE
        .EXTRN  BAD$GL_BAD_TERM
        .EXTRN  BAD$GB_BLOCK_FACT
        .EXTRN  BAD$GL_BYTES_CYL
        .EXTRN  BAD$GL_BYTES_TRK
        .EXTRN  BAD$GL_CHAN, BAD$GA_COMND_LINE
        .EXTRN  BAD$GL_CONTEXT, BAD$GL_CYLINDERS
        .EXTRN  BAD$GA_BUFADR, BAD$GL_DEVCHAR
        .EXTRN  BAD$GL_DEVCLASS
        .EXTRN  BAD$GA_DEVICE, BAD$GA_DEVNAM
        .EXTRN  BAD$GL_DEVNAM, BAD$GQ_DEVNAM
        .EXTRN  BAD$GA_FILESPEC
        .EXTRN  BAD$GL_FUNC, BAD$GA_GETDVI
        .EXTRN  BAD$GA_INPUT_DESC
        .EXTRN  BAD$GQ_IOSB, BAD$GL_MAXBLOCK
        .EXTRN  BAD$GA_MDBSF, BAD$GL_MDBSF_PTR
        .EXTRN  BAD$GL_PAGCNT, BAD$GA_SDBSF
        .EXTRN  BAD$GL_SDBSF_PTR
        .EXTRN  BAD$GL_SECTORS, BAD$GL_SECTOR_SIZ
        .EXTRN  BAD$GL_SERIALNUM
        .EXTRN  BAD$GL_STATUS, BAD$GB_TERM_COUNT
        .EXTRN  BAD$GA_TPB, BAD$GL_TRACKS
        .EXTRN  BAD$GL_TRNSFR_CNT
        .EXTRN  BAD$_ALIGNDISK, BAD$_ASSIGN
        .EXTRN  BAD$_BADVALUE, BAD$_DEVNOTBLK
        .EXTRN  BAD$_DEVNOTFOR, BAD$_DEVRCT
        .EXTRN  BAD$_GETDVI, BAD$_INSVIRMEM
        .EXTRN  BAD$_INVALQUAL, BAD$_IVBLKNUM
        .EXTRN  BAD$_IVBLKENT, BAD$_LSTSTRING
        .EXTRN  BAD$_MDBSFCRUPT
```

```
                                                        .EXTRN  BAD$_MDBSFRFAIL
                                                        .EXTRN  BAD$_NOBADINFO, BAD$_NOOUTQUAL
                                                        .EXTRN  BAD$_REBLDWARN, BAD$_SDBSFFULL
                                                        .EXTRN  BAD$_SDBSFRFAIL
                                                        .EXTRN  SYS$GETDVIW, SYS$ASSIGN
                                                        .WEAK   BAD$STA_INIT, BAD$STR_TRIM
                                                        .WEAK   BAD$VALIDATE_PACK

                                                        .PSECT  $CODE$,NOWRT,2

                                        0FFC 00000      .ENTRY  BAD$INIT, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0456
                                                                R11
                                    5B 00000000G  00  9E 00002  MOVAB   CLI$GET_VALUE, R11                    ;
                                    5A      0000G CF  9E 00009  MOVAB   BAD$GA_DEVICE, R10                   ;
                                    59 00000000G  00  9E 0000E  MOVAB   LIB$STOP, R9                         ;
                                    58      0000' CF  9E 00015  MOVAB   BAD_BLOCKS, R8                       ;
                                    57 00000000G  00  9E 0001A  MOVAB   CLI$PRESENT, R7                      ;
                                    56      0000G CF  9E 00021  MOVAB   BAD$GL_CONTEXT, R6                   ;
                                    66                10  88 00026  BISB2   #16, BAD$GL_CONTEXT               ; 0482
        08              00          6E                00  2C 00029  MOVC5   #0, (SP), #0, #8, BAD$GA_COMND_LINE ; 0487
                                                0000G   CF     0002E
        08              00          6E                00  2C 00031  MOVC5   #0, (SP), #0, #8, BAD$GA_DEVICE   ; 0488
                                                          6A   00036
        08              00          6E                00  2C 00037  MOVC5   #0, (SP), #0, #8, BAD$GA_INPUT_DESC ; 0489
                                                0000G   CF     0003C
        08              00          6E                00  2C 0003F  MOVC5   #0, (SP), #0, #8, BAD$GA_FILESPEC  ; 0490
                                                0000G   CF     00044
                                0000G   CF                  02  90 00047  MOVB    #2, BAD$GA_COMND_LINE+3       ; 0491
                                  03    AA                  02  90 0004C  MOVB    #2, BAD$GA_DEVICE+3           ; 0492
                                0000G   CF                  02  90 00050  MOVB    #2, BAD$GA_INPUT_DESC+3       ; 0493
                                0000G   CF                  02  90 00055  MOVB    #2, BAD$GA_FILESPEC+3         ; 0494
                                        50          0000G   CF  9E 0005A  MOVAB   BAD$STA_INIT, R0              ; 0496
                                                            05  13 0005F  BEQL    1$
                                0000G   CF                  00  FB 00061  CALLS   #0, BAD$STA_INIT            ; 0497
                                                0000G   CF  9F 00066 1$:  PUSHAB  BAD$GA_COMND_LINE           ; 0502
                                                  20    A8  9F 0006A      PUSHAB  COMMAND_LIN
                                                6B          02  FB 0006D  CALLS   #2, CLI$GET_VALUE
                                                50      0000G   CF  9E 00070  MOVAB   BAD$STR_TRIM, R0         ; 0503
                                                            52  D4 00075  CLRL    R2
                                                            50  D5 00077  TSTL    R0
                                                            0B  13 00079  BEQL    2$
                                                            52  D6 0007B  INCL    R2
                                                0000G   CF  9F 0007D      PUSHAB  BAD$GA_COMND_LINE           ; 0504
                                0000G   CF                  01  FB 00081  CALLS   #1, BAD$STR_TRIM
                                                            5A  DD 00086 2$:  PUSHL   R10                     ; 0506
                                                  50    A8  9F 00088      PUSHAB  DEVICE
                                                6B          02  FB 0008B  CALLS   #2, CLI$GET_VALUE
                                                07          52  E9 0008E  BLBC    R2, 3$                      ; 0507
                                                            5A  DD 00091  PUSHL   R10                         ; 0508
                                0000G   CF                  01  FB 00093  CALLS   #1, BAD$STR_TRIM
                                                            7E  7C 00098 3$:  CLRQ    -(SP)                   ; 0512
                                                            7E  7C 0009A      CLRQ    -(SP)
                                                0000G   CF  9F 0009C      PUSHAB  BAD$GA_GETDVI
                                                            5A  DD 000A0      PUSHL   R10
                                                            7E  7C 000A2      CLRQ    -(SP)
                            00000000G   00                  08  FB 000A4  CALLS   #8, SYS$GETDVIW
                                0000G   CF                  50  D0 000AB  MOVL    R0, BAD$GL_STATUS
```

```
                                11              50  E8 000B0          BLBS    R0, 4$                                                          ;
                                      0000G     CF  DD 000B3          PUSHL   BAD$GL_STATUS                                                   ; 0514
                                                5A  DD 000B7          PUSHL   R10                                                             ;
                                                01  DD 000B9          PUSHL   #1                                                              ;
                                   00000000G    8F  DD 000BB          PUSHL   #BAD$_GETDVI                                                    ;
                                69              04  FB 000C1          CALLS   #4, LIB$STOP                                                    ;
                                01      0000G   CF  D1 000C4 4$:      CMPL    BAD$GL_DEVCLASS, #1                                             ; 0516
                                                0D  13 000C9          BEQL    5$                                                              ;
                                                5A  DD 000CB          PUSHL   R10                                                             ; 0518
                                                01  DD 000CD          PUSHL   #1                                                              ;
                                   00000000G    8F  DD 000CF          PUSHL   #BAD$_DEVNOTBLK                                                 ;
                                69              03  FB 000D5          CALLS   #3, LIB$STOP                                                    ;
                                0D      0000G   CF  E9 000D8 5$:      BLBC    BAD$GL_DEVCHAR+1, 6$                                            ; 0520
                                                5A  DD 000DD          PUSHL   R10                                                             ; 0522
                                                01  DD 000DF          PUSHL   #1                                                              ;
                                   00000000G    8F  DD 000E1          PUSHL   #BAD$_DEVRCT                                                    ;
                                69              03  FB 000E7          CALLS   #3, LIB$STOP                                                    ;
                                14      0000G   CF  E8 000EA 6$:      BLBS    BAD$GL_DEVCHAR+3, 7$                                            ; 0524
                                50      0000G   CF  9E 000EF          MOVAB   BAD$STA_INIT, R0                                                ; 0525
                                                0D  12 000F4          BNEQ    7$                                                              ;
                                                5A  DD 000F6          PUSHL   R10                                                             ; 0527
                                                01  DD 000F8          PUSHL   #1                                                              ;
                                   00000000G    8F  DD 000FA          PUSHL   #BAD$_DEVNOTFOR                                                 ;
                                69              03  FB 00100          CALLS   #3, LIB$STOP                                                    ;
                                                7E  7C 00103 7$:      CLRQ    -(SP)                                                           ; 0531
                                      0000G     CF  9F 00105          PUSHAB  BAD$GL_CHAN                                                     ;
                                                5A  DD 00109          PUSHL   R10                                                             ;
                   00000000G    00              04  FB 0010B          CALLS   #4, SYS$ASSIGN                                                  ;
                        0000G   CF              50  D0 00112          MOVL    R0, BAD$GL_STATUS                                               ;
                                11              50  E8 00117          BLBS    R0, 8$                                                          ;
                                      0000G     CF  DD 0011A          PUSHL   BAD$GL_STATUS                                                   ; 0533
                                                5A  DD 0011E          PUSHL   R10                                                             ;
                                                01  DD 00120          PUSHL   #1                                                              ;
                                   00000000G    8F  DD 00122          PUSHL   #BAD$_ASSIGN                                                    ;
                                69              04  FB 00128          CALLS   #4, LIB$STOP                                                    ;
                                50      0000G   CF  9E 0012B 8$:      MOVAB   BAD$VALIDATE_PACK, R0                                           ; 0535
                                                05  13 00130          BEQL    9$                                                              ;
                        0000G   CF              00  FB 00132          CALLS   #0, BAD$VALIDATE_PACK                                           ; 0536
                        0000G   CF      0000G   CF  B0 00137 9$:      MOVW    BAD$GL_DEVNAM, BAD$GQ_DEVNAM                                    ; 0538
                        0000G   CF      020E    8F  B0 0013E          MOVW    #526, BAD$GQ_DEVNAM+2                                           ; 0539
                        0000G   CF      0000G   CF  9E 00145          MOVAB   BAD$GA_DEVNAM, BAD$GQ_DEVNAM+4                                  ; 0541
                                52      0000G   CF  D0 0014C          MOVL    BAD$GL_MAXBLOCK, R2                                             ; 0543
                   00001000     8F              52  D1 00151          CMPL    R2, #4096                                                       ;
                                67              15 00158              BLEQ    11$                                                             ;
                          01    A6              01  88 0015A          BISB2   #1, BAD$GL_CONTEXT+1                                            ; 0546
                50      0000G   CF      0000G   CF  C5 0015E          MULL3   BAD$GL_CYLINDERS, BAD$GL_TRACKS, R0                             ; 0548
                                51      0000G   CF  D0 00166          MOVL    BAD$GL_SECTORS, R1                                              ; 0549
                                50              51  C4 0016B          MULL2   R1, R0                                                          ;
                53              50              52  C7 0016E          DIVL3   R2, R0, R3                                                      ; 0550
                        0000G   CF              53  90 00172          MOVB    R3, BAD$GB_BLOCK_FACT                                           ;
                                50      0000G   CF  9A 00177          MOVZBL  BAD$GB_BLOCK_FACT, R0                                           ; 0553
                50              51              50  C7 0017C          DIVL3   R0, R1, R0                                                      ;
          0000G CF              52              50  C3 00180          SUBL3   R0, R2, BAD$GL_MDBSF_PTR                                        ; 0552
                                50      0000G   CF  9A 00186          MOVZBL  BAD$GB_BLOCK_FACT, R0                                           ; 0554
          0000G CF  00000200    8F              50  C7 0018B          DIVL3   R0, #512, BAD$GL_SECTOR_SIZ                                     ;
          0000G CF              51      0000G   CF  C5 00195          MULL3   BAD$GL_SECTOR_SIZ, R1, BAD$GL_BYTES_TRK                         ; 0555
          0000G CF      0000G   CF      0000G   CF  C5 0019D          MULL3   BAD$GL_TRACKS, BAD$GL_BYTES_TRK, -                               ; 0556
```

```
                                                                BAD$GL_BYTES_CYL                              ;
                     50        0000G  CF  D0 001A7       MOVL   BAD$GL_BYTES_CYL, R0                          ; 0557
            0000FE00  8F                  50  D1 001AC       CMPL   R0, #65024                                    ;
                      05                  1B 001B3       BLEQU  10$                                           ;
                     50         FE00  8F  3C 001B5       MOVZWL #65024, R0                                    ;
                0000G CF                  50  D0 001BA 10$:  MOVL   R0, BAD$GL_TRNSFR_CNT                         ;
                      07                  11 001BF       BRB    12$                                           ; 0543
                0000G CF        FE00  8F  3C 001C1 11$:  MOVZWL #65024, BAD$GL_TRNSFR_CNT                     ; 0561
      0000G CF  0000G CF  00000200  8F  C7 001C8 12$:  DIVL3  #512, BAD$GL_TRNSFR_CNT, BAD$GL_PAGCNT       ; 0563
                      58                  DD 001D4       PUSHL  R8                                            ; 0577
                      67                  01  FB 001D6       CALLS  #1, CLI$PRESENT                               ;
   66        01       00                  50  F0 001D9       INSV   R0, #0, #1, BAD$GL_CONTEXT                    ;
                                 60   A8  9F 001DE       PUSHAB EXERCISE                                      ; 0578
                      67                  01  FB 001E1       CALLS  #1, CLI$PRESENT                               ;
   66        01       02                  50  F0 001E4       INSV   R0, #2, #1, BAD$GL_CONTEXT                    ;
                                 78   A8  9F 001E9       PUSHAB KEEP                                          ; 0579
                      67                  01  FB 001EC       CALLS  #1, CLI$PRESENT                               ;
   66        01       06                  50  F0 001EF       INSV   R0, #6, #1, BAD$GL_CONTEXT                    ;
                               0084   C8  9F 001F4       PUSHAB LOG                                           ; 0580
                      67                  01  FB 001F8       CALLS  #1, CLI$PRESENT                               ;
   66        01       07                  50  F0 001FB       INSV   R0, #7, #1, BAD$GL_CONTEXT                    ;
                               00BC   C8  9F 00200       PUSHAB OUTPUT                                        ; 0581
                      67                  01  FB 00204       CALLS  #1, CLI$PRESENT                               ;
01 A6        01       01                  50  F0 00207       INSV   R0, #1, #1, BAD$GL_CONTEXT+1                  ;
                               00F4   C8  9F 0020D       PUSHAB RETRY                                         ; 0582
                      67                  01  FB 00211       CALLS  #1, CLI$PRESENT                               ;
01 A6        01       03                  50  F0 00214       INSV   R0, #3, #1, BAD$GL_CONTEXT+1                  ;
                               0120   C8  9F 0021A       PUSHAB SHOW_QUAL                                     ; 0583
                      67                  01  FB 0021E       CALLS  #1, CLI$PRESENT                               ;
01 A6        01       04                  50  F0 00221       INSV   R0, #4, #1, BAD$GL_CONTEXT+1                  ;
                0000G CF                  21  D0 00227       MOVL   #33, BAD$GL_FUNC                              ; 0596
                      21          01      A6  E9 0022C       BLBC   BAD$GL_CONTEXT+1, 13$                         ; 0597
                               0000G  CF  9F 00230       PUSHAB BAD$GA_MDBSF                                  ; 0600
                      7E       0400   8F  3C 00234       MOVZWL #1024, -(SP)                                  ;
                0000V CF                  02  FB 00239       CALLS  #2, BAD$ALLOC_MEM                             ;
      0000G CF  0000G CF  00000200  8F  C1 0023E       ADDL3  #512, BAD$GA_MDBSF, BAD$GA_SDBSF              ; 0601
                0000V CF                  00  FB 0024A       CALLS  #0, GET_MDBSF                                 ; 0602
                                          0E  11 0024F       BRB    14$                                           ; 0597
                               0000G  CF  9F 00251 13$:  PUSHAB BAD$GA_SDBSF                                  ; 0605
                      7E       0200   8F  3C 00255       MOVZWL #512, -(SP)                                   ;
                0000V CF                  02  FB 0025A       CALLS  #2, BAD$ALLOC_MEM                             ;
             16    01 A6                  01  E1 0025F 14$:  BBC    #1, BAD$GL_CONTEXT+1, 15$                     ; 0626
                               0000G  CF  9F 00264       PUSHAB BAD$GA_FILESPEC                               ; 0628
                               00BC   C8  9F 00268       PUSHAB OUTPUT                                        ;
                      6B                  02  FB 0026C       CALLS  #2, CLI$GET_VALUE                             ;
           00000000G 00                   00  FB 0026F       CALLS  #0, BAD$PREPARE_REPORT                        ; 0629
                   01 A6                  20  88 00276       BISB2  #32, BAD$GL_CONTEXT+1                         ; 0630
             63    01 A6                  04  E1 0027A 15$:  BBC    #4, BAD$GL_CONTEXT+1, 22$                     ; 0652
             17    01 A6                  01  E0 0027F       BBS    #1, BAD$GL_CONTEXT+1, 16$                     ; 0654
                               0000G  CF  9F 00284       PUSHAB BAD$GA_COMND_LINE                             ; 0656
                                          01  DD 00288       PUSHL  #1                                            ;
                           00000000G  8F  DD 0028A       PUSHL  #BAD$_LSTSTRING                               ;
                                          7E  D4 00290       CLRL   -(SP)                                         ;
                           00000000G  8F  DD 00292       PUSHL  #BAD$_NOOUTQUAL                               ;
                      69                  05  FB 00298       CALLS  #5, LIB$STOP                                  ;
                                 10   A8  9F 0029B 16$:  PUSHAB BEFORE                                        ; 0658
                      67                  01  FB 0029E       CALLS  #1, CLI$PRESENT                               ;
```

```
 01   A6         01             06               50  F0 002A1         INSV     R0, #6, #1, BAD$GL_CONTEXT+1         ;
                                34               50  E9 002A7         BLBC     R0, 21$                              ;
                                07         01    A6  E9 002AA         BLBC     BAD$GL_CONTEXT+1, 17$                ; 0661
                         0000V  CF               00  FB 002AE         CALLS    #0, GET_SDBSF                        ; 0662
                                0B               11  002B3            BRB      18$                                  ;
                         0000V  CF               00  FB 002B5 17$:    CALLS    #0, GET_NLT_SDBSF                    ; 0663
 01   A6         01             07               50  F0 002BA         INSV     R0, #7, #1, BAD$GL_CONTEXT+1         ;
                                05         01    A6  E8 002C0 18$:    BLBS     BAD$GL_CONTEXT+1, 19$                ; 0664
                                           01    A6  95 002C4         TSTB     BAD$GL_CONTEXT+1                     ;
                                                 07  18 002C7         BGEQ     20$                                  ;
                      00000000G 00               00  FB 002C9 19$:    CALLS    #0, BAD$PRODUCE_REPORT               ; 0665
                                           EC    A8  9F 002D0 20$:    PUSHAB   AFTER                                ; 0666
                                67               01  FB 002D3         CALLS    #1, CLI$PRESENT                      ;
 01   A6         01             05               50  F0 002D6         INSV     R0, #5, #1, BAD$GL_CONTEXT+1         ;
                                                 04  11 002DC         BRB      22$                                  ; 0658
                            01  A6               20  88 002DE 21$:    BISB2    #32, BAD$GL_CONTEXT+1                ; 0669
                 09             66               06  E0 002E2 22$:    BBS      #6, BAD$GL_CONTEXT, 23$              ; 0693
                 17             66               02  E0 002E6         BBS      #2, BAD$GL_CONTEXT, 25$              ; 0694
                 12         01  A6               06  E0 002EA         BBS      #6, BAD$GL_CONTEXT+1, 25$            ;
                                07         01    A6  E9 002EF 23$:    BLBC     BAD$GL_CONTEXT+1, 24$                ; 0696
                         0000V  CF               00  FB 002F3         CALLS    #0, GET_SDBSF                        ; 0697
                                                 0C  11 002F8         BRB      26$                                  ;
                         0000V  CF               00  FB 002FA 24$:    CALLS    #0, GET_NLT_SDBSF                    ; 0698
                                                 05  11 002FF         BRB      26$                                  ; 0696
                         0000V  CF               00  FB 00301 25$:    CALLS    #0, INIT_SDBSF                       ; 0700
                                4F               66  E9 00306 26$:    BLBC     BAD$GL_CONTEXT, 31$                  ; 0736
                                                 52  D4 00309         CLRL     VALUE_PRESENT                        ; 0741
                                         0000G   CF  9F 0030B 27$:    PUSHAB   BAD$GA_INPUT_DESC                    ; 0742
                                                 58  DD 0030F         PUSHL    R8                                   ;
                                6B               02  FB 00311         CALLS    #2, CLI$GET_VALUE                    ;
                         0000G  CF               50  D0 00314         MOVL     R0, BAD$GL_STATUS                    ;
                                0A               50  E9 00319         BLBC     R0, 28$                              ;
                                52               01  D0 0031C         MOVL     #1, VALUE_PRESENT                    ; 0744
                         0000V  CF               00  FB 003..         CALLS    #0, PROCESS_BADBLOCKS                ; 0745
                                                 E5  11 00324         BRB      27$                                  ; 0742
                                2C               52  E8 00326 28$:    BLBS     VALUE_PRESENT, 30$                   ; 0748
                                66               20  88 00329         BISB2    #32, BAD$GL_CONTEXT                  ; 0751
                                         00E4    C8  9F 0032C 29$:    PUSHAB   PROMPT                               ; 0752
                                         0000G   CF  9F 00330         PUSHAB   BAD$GA_INPUT_DESC                    ;
                      00000000G 00               02  FB 00334         CALLS    #2, LIB$GET_INPUT                    ;
                          0000G CF               50  D0 0033B         MOVL     R0, BAD$GL_STATUS                    ;
                                12               50  E9 00340         BLBC     R0, 30$                              ;
                      0001827A  8F       0000G   CF  D1 00343         CMPL     BAD$GL_STATUS, #98938                ; 0755
                                                 07  13 0034C         BEQL     30$                                  ;
                         0000V  CF               00  FB 0034E         CALLS    #0, PROCESS_BADBLOCKS                ; 0757
                                                 D7  11 00353         BRB      29$                                  ; 0752
                                66               20  8A 00355 30$:    BICB2    #32, BAD$GL_CONTEXT                  ; 0760
                 50             66               02  E1 00358 31$:    BBC      #2, BAD$GL_CONTEXT, 33$              ; 0790
                                           6C    A8  9F 0035C         PUSHAB   FULL                                 ; 0792
                                67               01  FB 0035F         CALLS    #1, CLI$PRESENT                      ;
      66         01             03               50  F0 00362         INSV     R0, #3, #1, BAD$GL_CONTEXT           ;
                            01  A6         80    8F  88 00367         BISB2    #128, BAD$GL_CONTEXT+1               ; 0793
                                         00CC    C8  9F 0036C         PUSHAB   PATTERN                              ; 0795
                                67               01  FB 00370         CALLS    #1, CLI$PRESENT                      ;
 01   A6         01             02               50  F0 00373         INSV     R0, #2, #1, BAD$GL_CONTEXT+1         ;
                                05               50  E9 00379         BLBC     R0, 32$                              ;
                         0000V  CF               00  FB 0037C         CALLS    #0, GET_PATTERN                      ; 0797
```

```
                                 0000G  CF   9F 00381 32$:   PUSHAB  BAD$GA_BUFADR                                  ; 0800
                          7E     0000G  CF        0A 78 00385        ASHL    #10, BAD$GL_PAGCNT, -(SP)              ; 0799
                                 0000V  CF        02 FB 00388        CALLS   #2, BAD$ALLOC_MEM                      ;
                          50     0000G  CF        09 78 00390        ASHL    #9, BAD$GL_PAGCNT, R0                  ; 0803
                                 0000G  CF  0000GDF40 9E 00396        MOVAB   @BAD$GA_BUFADR[R0], BAD$GA_BUFADR+4    ;
                                        0000G  CF  9F 0039E          PUSHAB  BAD$GA_TPB                             ; 0806
                                           7E 0200  8F 3C 003A2      MOVZWL  #512, -(SP)                            ;
                                 0000V     CF        02 FB 003A7     CALLS   #2, BAD$ALLOC_MEM                      ;
                                           66        10 8A 003AC 33$: BICB2   #16, BAD$GL_CONTEXT                    ; 0812
                                                     04 003AF        RET                                            ; 0814

; Routine Size:  944 bytes,    Routine Base:  $CODE$ + 0000

;  705          0815  1
```

```
;  707          0816  1 %SBTTL  'bad$alloc_mem -- Allocate Memory'
;  708          0817  1 GLOBAL ROUTINE  bad$alloc_mem (byte_count, return_address) : NOVALUE =
;  709          0818  1 !++
;  710          0819  1 !
;  711          0820  1 ! Functional Description:
;  712          0821  1 !
;  713          0822  1 !    This routine will allocate page(s) of memory.  If an error occurs while
;  714          0823  1 !    attempting to allocate the memory, the error will be signalled FATAL.
;  715          0824  1 !
;  716          0825  1 ! Inputs:
;  717          0826  1 !
;  718          0827  1 !    byte_count          VALUE   Number of bytes to allocate.
;  719          0828  1 !
;  720          0829  1 ! Outputs:
;  721          0830  1 !
;  722          0831  1 !    return_address      ADDRESS Address of the location to store the
;  723          0832  1 !                                beginning address of allocated memory.
;  724          0833  1 !
;  725          0834  1 ! Side Effects:
;  726          0835  1 !
;  727          0836  1 !    If there is an error in allocating memory, this procedure will terminate
;  728          0837  1 !    the analysis via a SIGNAL_STOP.
;  729          0838  1 !
;  730          0839  1 !--
;  731          0840  2 BEGIN
;  732          0841  2 MAP
;  733          0842  2     return_address : REF $BBLOCK;                       ! Point to return address.
;  734          0843  2
;  735          0844  2 LOCAL
;  736          0845  2     addr_vec : VECTOR [2, LONG],
;  737          0846  2     page_count;
;  738          0847  2
;  739          0848  2 IF (page_count = .byte_count / bad$k_page_size) LSS 1
;  740          0849  2 THEN    page_count = 1;
;  741          0850  2
;  742        P 0851  3 IF NOT (bad$gl_status = $EXPREG (PAGCNT = .page_count,
;  743          0852  3                                  RETADR = addr_vec))       ! Allocate the requested amount
;  744          0853  2 THEN                                                      ! of storage.
;  745          0854  2     SIGNAL_STOP (bad$_insvirmem, 0, .bad$gl_status);
;  746          0855  2
;  747          0856  2 .return_address = .addr_vec [0];
;  748          0857  2
;  749          0858  1 END;    ! Of GLOBAL ROUTINE bad$alloc_mem


                                                        .EXTRN  SYS$EXPREG

                                              0000 00000        .ENTRY  BAD$ALLOC_MEM, Save nothing                     ; 0817
                                 5E           08  C2 00002        SUBL2   #8, SP                                          ;
                  50       04    AC 00000200  8F  C7 00005        DIVL3   #512, BYTE_COUNT, PAGE_COUNT                    ; 0848
                                              03  14 0000E        BGTR    1$                                          ;
                                 50           01  D0 00010        MOVL    #1, PAGE_COUNT                              ; 0849
                                              7E  7C 00013 1$:    CLRQ    -(SP)                                       ; 0852          ;
                                       08     AE  9F 00015        PUSHAB  ADDR_VEC                                    ;
                                              50  DD 00018        PUSHL   PAGE_COUNT                                  ;
                     00000000G    00          04  FB 0001A        CALLS   #4, SYS$EXPREG                              ;               ;
```

```
                          0000G  CF              50  DO 00021          MOVL    R0, BAD$GL_STATUS                       ;
                                 13              50  E8 00026          BLBS    R0, 2$                                  ;
                                           0000G  CF  DD 00029          PUSHL   BAD$GL_STATUS                   ; 0854
                                                  7E  D4 0002D          CLRL    -(SP)                                   ;
                                      00000000G   8F  DD 0002F          PUSHL   #BAD$_INSVIRMEM                         ;
                          00000000G   00          03  FB 00035          CALLS   #3, LIB$STOP                            ;
                                 08   BC          6E  DO 0003C 2$:      MOVL    ADDR_VEC, @RETURN_ADDRESS               ; 0856
                                                     04 00040          RET                                             ; 0858

; Routine Size:  65 bytes,    Routine Base:  $CODE$ + 03B0

;  750          0859  1
```

```
;  752     0860  1 %SBTTL  'get_mdbsf -- Get a valid copy of the MDBSF'
;  753     0861  1 ROUTINE get_mdbsf : NOVALUE =
;  754     0862  1 !++
;  755     0863  1 !
;  756     0864  1 ! Functional Description:
;  757     0865  1 !
;  758     0866  1 !    Obtain a valid copy of the Manufacturer's Detected Bad Sector File.
;  759     0867  1 !
;  760     0868  1 ! Implicit Inputs:
;  761     0869  1 !
;  762     0870  1 !    bad$ga_mdbsf       the address of the MDBSF buffer.
;  763     0871  1 !    bad$ga_sdbsf       the address of the SDBSF buffer.
;  764     0872  1 !    bad$gl_mdbsf_ptr   the block number of the first block of the last track.
;  765     0873  1 !
;  766     0874  1 ! Side Effects:
;  767     0875  1 !
;  768     0876  1 !    If unable to read the MDBSF, we will recreate BOTH the MDBSF and SDBSF
;  769     0877  1 !    ONLY if we are going to exercise the medium and the user specified NOKEEP.
;  770     0878  1 !    If he specified /EXERCISE=KEEP, we will abort the analysis.
;  771     0879  1 !
;  772     0880  1 !    If the request was something like /NOEXERCISE /OUTPUT we will attempt to
;  773     0881  1 !    read a valid non last track format SDBSF.  Failing this we will
;  774     0882  1 !    SIGNAL_STOP to inform the user of the situation.  If we were successful
;  775     0883  1 !    in reading the non last track format SDBSF, we will convert the analysis
;  776     0884  1 !    context to non last track by clearing the [ctx_v_ltdevice] context bit.
;  777     0885  1 !    Thus allowing a listing to be produced.
;  778     0886  1 !
;  779     0887  1 !    In order to be able to exercise the media or update the bad block file,
;  780     0888  1 !    the user must execute ANALYZE/MEDIA explicitly specifing /EXERCISE=NOKEEP.
;  781     0889  1 !    This will convert the device to last track format and implies that the
;  782     0890  1 !    user knows that data will be lost and the bad block files will be rebuilt.
;  783     0891  1 !
;  784     0892  1 !--
;  785     0893  2 BEGIN
;  786     0894  2 LOCAL
;  787     0895  2     first_found,
;  788     0896  2     value_present;
;  789     0897  2
;  790     0898  2 first_found = FALSE;                                    ! Indicator for buffer usage (dup copies of MDBSF).
;  791     0899  2 value_present = FALSE;                                  ! Indicate whether we found a valid MDBSF.
;  792     0900  2 INCR block FROM 0 TO 9 DO                               ! Search the first 10 blocks of the
;  793     0901  3     BEGIN                                               ! last track for the MDBSF.  Once found
;  794     0902  3     IF bad$gl_status = bad$sync_io (IF NOT .first_found ! a second copy is obtained to validate
;  795     0903  3                                     THEN    .bad$ga_mdbsf   ! the contents (since no checksum is
;  796     0904  3                                     ELSE    .bad$ga_sdbsf,  ! available).
;  797     0905  3                                     bad$k_page_size,
;  798     0906  3                                     .bad$gl_mdbsf_ptr + .block)
;  799     0907  3     THEN
;  800     0908  4         BEGIN
;  801     0909  4         IF NOT .first_found                             ! If this is the first time an MDBSF
;  802     0910  4         THEN                                            ! was found, check that it is not full,
;  803     0911  5             BEGIN                                       ! switch to second buffer for verfication.
;  804     0912  5             IF .(.bad$ga_mdbsf + (bad$k_page_size - 4)) EQL -1
;  805     0913  5             THEN    first_found = TRUE;                 ! Indicate that we found one.
;  806     0914  5             END
;  807     0915  4         ELSE
;  808     0916  5             BEGIN                                       ! Validate the MDBSF copies,
```

```
;  809          0917  5              IF CH$EQL (bad$k_page_size, .bad$ga_mdbsf,  ! they should be identical.
;  810          0918  5              bad$k_page_size, .bad$ga_sdbsf,0)
;  811          0919  5              THEN
;  812          0920  6                  BEGIN
;  813          0921  6                  value_present = TRUE;                   ! We have a valid copy of the MDBSF,
;  814          0922  6                  EXITLOOP;                               ! go finish the validation of the
;  815          0923  5                  END;                                    ! pack.
;  816          0924  4              END;
;  817          0925  3          END;
;  818          0926  2      END;
;  819          0927  2
;  820          0928  2 IF NOT .value_present                                   ! If we did not find a valid MDBSF,
;  821          0929  2 THEN                                                    ! see if we can either rebuild a new
;  822          0930  2     IF NOT recover_dbsfs ()                            ! set of bad block flies or recover
;  823          0931  2     THEN                                               ! a non lasttrack SDBSF.  If we can't,
;  824          0932  2         SIGNAL_STOP (bad$_mdbsfrfail, 1, bad$ga_device);! then abort the analysis.
;  825          0933  2
;  826          0934  2 IF NOT .bad$gl_context [ctx_v_ltdevice]                ! Did we recover a non last track SDBSF?
;  827          0935  2 THEN
;  828          0936  2     RETURN;                                            ! Yes, bypass the pack validation.
;  829          0937  2
;  830          0938  2 !
;  831          0939  2 ! Validate the pack...
;  832          0940  2 !
;  833          0941  2 IF .bad$ga_mdbsf [ltk_w_cartridge] EQLU %X'FFFF'        ! Assure that it's not an
;  834          0942  2 THEN                                                    ! alignment cartridge.
;  835          0943  2     SIGNAL_STOP (bad$_aligndisk, 1, bad$ga_device)
;  836          0944  2 ELSE
;  837          0945  3     BEGIN                                              ! Has the header been corrrupted?
;  838          0946  3     IF (.bad$ga_mdbsf [ltk_l_serial_number] EQL 0) AND ! It should have a serial number
;  839          0947  4        (.bad$ga_mdbsf [ltk_w_unused] NEQ 0)            ! followed by a zero word, and the
;  840          0948  3     THEN                                               ! cartridge id (0's-data packs, 1's-diag pack)
;  841          0949  3         IF .bad$gl_context [ctx_v_exercise]            ! in the header.
;  842          0950  3         THEN
;  843          0951  3             rebuild_dbsfs ()                           ! Rebuild the MDBSF and SDBSF if we can.
;  844          0952  3         ELSE
;  845          0953  3             SIGNAL_STOP (bad$_mdbsfcrupt, 1, bad$ga_device);
;  846          0954  2     END;
;  847          0955  2
;  848          0956  1 END;     ! of ROUTINE get_mdbsf
```

```
                        03FC 00000 GET_MDBSF:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                    ; 0861
59      0000G   CF   9E 00002              MOVAB   BAD$GA_DEVICE, R9                               ;
58 00000000G    00   9E 00007              MOVAB   LIB$STOP, R8                                    ;
57      0000G   CF   9E 0000E              MOVAB   BAD$GA_MDBSF, R7                                ;
                55   7C 00013              CLRQ    VALUE_PRESENT                                   ; 0899
                54   D4 00015              CLRL    BLOCK                                           ; 0900
        0000GDF 44   9F 00017 1$:          PUSHAB  @BAD$GL_MDBSF_PTR[BLOCK]                        ; 0906
7E      0200    8F   3C 0001C              MOVZWL  #512, -(SP)                                     ; 0902
04              56   E8 00021              BLBS    FIRST_FOUND, 2$                                 ;
                67   DD 00024              PUSHL   BAD$GA_MDBSF                                    ; 0903
                04   11 00026              BRB     3$                                              ;
```

```
                                                0000G  CF   DD 00028 2$:     PUSHL   BAD$GA_SDBSF                                    ; 0904
                                    00000000G   00       03   FB 0002C 3$:     CALLS   #3, BAD$SYNC_IO                                 ; 0902
                                        0000G   CF       50   D0 00033         MOVL    R0, BAD$GL_STATUS
                                                26       50   E9 00038         BLBC    R0, 5$
                                                13       56   E8 0003B         BLBS    FIRST_FOUND, 4$                                 ; 0912
                                                50       67   D0 0003E         MOVL    BAD$GA_MDBSF, R0
                                    FFFFFFFF    8F   01FC   C0   D1 00041         CMPL    508(R0), #-1
                                                         15   12 0004A         BNEQ    5$
                                                56       01   D0 0004C         MOVL    #1, FIRST_FOUND                                 ; 0913
                                                         10   11 0004F         BRB     5$                                              ; 0909
                    0000G   DF          00   B7   0200   8F   29 00051 4$:     CMPC3   #512, @BAD$GA_MDBSF, @BAD$GA_SDBSF              ; 0917
                                                         05   12 0005A         BNEQ    5$
                                                55       01   D0 0005C         MOVL    #1, VALUE_PRESENT                               ; 0921
                                                         04   11 0005F         BRB     6$                                              ; 0920
                            B2                   54       09   F3 00061 5$:     AOBLEQ  #9, BLOCK, 1$                                   ; 0900
                                                15       55   E8 00065 6$:     BLBS    VALUE_PRESENT, 7$                               ; 0928
                                        0000V   CF       00   FB 00068         CALLS   #0, RECOVER_DBSFS                               ; 0930
                                                0D       50   E8 0006D         BLBS    R0, 7$
                                                         59   DD 00070         PUSHL   R9                                              ; 0932
                                                         01   DD 00072         PUSHL   #1
                                            00000000G    8F   DD 00074         PUSHL   #BAD$_MDBSFRFAIL
                                                68       03   FB 0007A         CALLS   #3, LIB$STOP
                                                39   0000G   CF   E9 0007D 7$:     BLBC    BAD$GL_CONTEXT+1, 11$                   ; 0934
                                                52       67   D0 00082         MOVL    BAD$GA_MDBSF, R2                                ; 0941
                                        FFFF    8F       06   A2   B1 00085         CMPW    6(R2), #65535
                                                         0C   12 0008B         BNEQ    8$
                                                59       DD   0008D           PUSHL   R9                                              ; 0943
                                                01       DD   0008F           PUSHL   #1
                                            00000000G    8F   DD 00091         PUSHL   #BAD$_ALIGNDISK
                                                         1F   11 00097         BRB     10$
                                                         62   D5 00099 8$:     TSTL    (R2)                                            ; 0946
                                                         1E   12 0009B         BNEQ    11$
                                                    04   A2   B5 0009D         TSTW    4(R2)                                           ; 0947
                                                         19   13 000A0         BEQL    11$
                            06          0000G   CF       02   E1 000A2         BBC     #2, BAD$GL_CONTEXT, 9$                          ; 0949
                                        0000V   CF       00   FB 000A8         CALLS   #0, REBUILD_DBSFS                               ; 0951
                                                         04   000AD           RET
                                                59       DD   000AE 9$:     PUSHL   R9                                              ; 0953
                                                01       DD   000B0         PUSHL   #1
                                            00000000G    8F   DD 000B2         PUSHL   #BAD$_MDBSFCRUPT
                                                68       03   FB 000B8 10$:     CALLS   #3, LIB$STOP
                                                         04   000BB 11$:     RET                                                     ; 0956

; Routine Size:  188 bytes,    Routine Base:  $CODE$ + 03F1

;  849          0957  1
```

```
;  851          0958  1 %SBTTL 'recover_dbsfs -- Attempt to rebuild/recover an SDBSF'
;  852          0959  1 ROUTINE recover_dbsfs =
;  853          0960  1 !++
;  854          0961  1 !
;  855          0962  1 ! Functional Description:
;  856          0963  1 !
;  857          0964  1 !    A valid MDBSF was not found.  Check to see if the analysis context
;  858          0965  1 !    will allow either of the following actions to be performed.
;  859          0966  1 !
;  860          0967  1 !     1. Attempt to rebuild both the MDBSF and the SDBSF if we are going to
;  861          0968  1 !        exercise the medium and not save the old bad block data.
;  862          0969  1 !
;  863          0970  1 !     2. Attempt to find the "old" non last track format SDBSF if the medium
;  864          0971  1 !        is not going to be exercised and only a listing type of action
;  865          0972  1 !        was requested.  If successful, then we will change the context to
;  866          0973  1 !        non last track.
;  867          0974  1 !
;  868          0975  1 ! Routine Value:
;  869          0976  1 !
;  870          0977  1 !    TRUE       If we were successful in rebuilding the MDBSF and SDBSF or
;  871          0978  1 !               in locating the "old" format SDBSF and changing the analysis
;  872          0979  1 !               context to non last track.
;  873          0980  1 !
;  874          0981  1 !    FALSE      If we could not (for one of several reasons) rebuild the
;  875          0982  1 !               the bad block files.
;  876          0983  1 !
;  877          0984  1 !--
;  878          0985  2 BEGIN
;  879          0986  2 LOCAL
;  880          0987  2     return_status;
;  881          0988  2
;  882          0989  2 return_status = FALSE;                          ! Assume we can't rebuild the bad block files.
;  883          0990  2 IF .bad$gl_context [ctx_v_exercise]             ! Check the context bits to see if we can
;  884          0991  2 AND NOT .bad$gl_context [ctx_v_keep]            ! attempt to rebuild them.
;  885          0992  2 THEN
;  886          0993  3     BEGIN
;  887          0994  3     rebuild_dbsfs ();                          ! We can try ...
;  888          0995  3     return_status = TRUE;                      ! We have written them to the medium.
;  889          0996  3     END
;  890          0997  2 ELSE
;  891          0998  3     BEGIN                                      ! Otherwise, if we aren't exercising the
;  892          0999  3     IF NOT .bad$gl_context [ctx_v_exercise]    ! medium and no attempt will be made to
;  893          1000  3     AND NOT .bad$gl_context [ctx_v_badblocks]  ! modify a non last track format SDBSF
;  894          1001  3     THEN                                       ! if one is found, then go look for one.
;  895          1002  3         IF get_nlt_sdbsf ()
;  896          1003  3         THEN
;  897          1004  4             BEGIN
;  898          1005  4             bad$gl_context [ctx_v_ltdevice] = FALSE;    ! Found one, change to context.
;  899          1006  4             return_status = TRUE;                       ! Set the status to allow us to continue.
;  900          1007  3             END;
;  901          1008  2     END;
;  902          1009  2
;  903          1010  2 RETURN .return_status;
;  904          1011  2
;  905          1012  1 END;     ! of ROUTINE recover_dbsfs
```

```
                                        000C 00000 RECOVER_DBSFS:
                                                   .WORD    Save R2,R3                                  ; 0959
                    53      0000G   CF  9E 00002           MOVAB   BAD$GL_CONTEXT, R3
                                    52  D4 00007           CLRL    RETURN_STATUS                       ; 0989
                0F          63      02  E1 00009           BBC     #2, BAD$GL_CONTEXT, 2$              ; 0990
                07          63      06  E0 0000D           BBS     #6, BAD$GL_CONTEXT, 1$              ; 0991
                    0000V   CF      00  FB 00011           CALLS   #0, REBUILD_DBSFS                   ; 0994
                                    13  11 00016           BRB     3$                                  ; 0995
                12          63      02  E0 00018 1$:       BBS     #2, BAD$GL_CONTEXT, 4$              ; 0999
                            0F      63  E8 0001C 2$:       BLBS    BAD$GL_CONTEXT, 4$                  ; 1000
                    0000V   CF      00  FB 0001F           CALLS   #0, GET_NLT_SDBSF                   ; 1002
                            07      50  E9 00024           BLBC    R0, 4$
                      01    A3      01  8A 00027           BICB2   #1, BAD$GL_CONTEXT+1                ; 1005
                            52      01  D0 0002B 3$:       MOVL    #1, RETURN_STATUS                   ; 1006
                            50      52  D0 0002E 4$:       MOVL    RETURN_STATUS, R0                   ; 1010
                                        04 00031           RET                                         ; 1012

; Routine Size:  50 bytes,    Routine Base:  $CODE$ + 04AD

;  906          1013  1
```

```
;  908          1014  1 %SBTTL  'rebuild_dbsfs -- Rebuild the MDBSF and SDBSF on last track devices'
;  909          1015  1 ROUTINE rebuild_dbsfs : NOVALUE =
;  910          1016  1 !++
;  911          1017  1 !
;  912          1018  1 ! Functional Description:
;  913          1019  1 !
;  914          1020  1 !    This procedure will rebuild the MDBSF and SDBSF on last track devices.
;  915          1021  1 !    The context of the analysis MUST BE EXERCISE and NOKEEP.  This is to
;  916          1022  1 !    assure that no data will be unintentionally lost due to the use of the
;  917          1023  1 !    last track for the detected bad block files data.
;  918          1024  1 !
;  919          1025  1 ! Implicit Inputs:
;  920          1026  1 !
;  921          1027  1 !    Global Data.
;  922          1028  1 !
;  923          1029  1 ! Implicit Outputs:
;  924          1030  1 !
;  925          1031  1 !    The MDBSF and SDBSF will be recreated on the medium as empty files.
;  926          1032  1 !
;  927          1033  1 ! Side Effects:
;  928          1034  1 !
;  929          1035  1 !    If we did not manage to write the new MDBSF and SDBSF onto at least
;  930          1036  1 !    half of the last track, we will issue a diagnostic to the user warning
;  931          1037  1 !    him of the unreliable nature of the last track data.
;  932          1038  1 !
;  933          1039  1 !--
;  934          1040  2 BEGIN
;  935          1041  2 LOCAL
;  936          1042  2     cnt : INITIAL (0);
;  937          1043  2
;  938          1044  2 bad$gl_func = IO$_WRITELBLK;                          ! Set up for writes.
;  939          1045  2 CH$FILL (-1, bad$k_page_size, .bad$ga_mdbsf);         ! Clean the buffer out.
;  940          1046  2 bad$ga_mdbsf [ltk_l_serial_number] = bad$gl_serialnum = 1; ! Default Serial Number.
;  941          1047  2 bad$ga_mdbsf [ltk_w_unused] = 0;                      ! Clear unused field.
;  942          1048  2 bad$ga_mdbsf [ltk_w_cartridge] = 0;                   ! Define it as a Data Cartridge.
;  943          1049  2
;  944          1050  2 INCR offset FROM 0 TO .bad$gl_sectors / .bad$gb_block_fact DO
;  945          1051  3     IF NOT (bad$gl_status = bad$sync_io (.bad$ga_mdbsf,
;  946          1052  3                                         bad$k_page_size,
;  947          1053  3                                         .bad$gl_mdbsf_ptr + .offset))
;  948          1054  2     THEN
;  949          1055  2         cnt = .cnt + 1;                                                          ;
;  950          1056  2
;  951          1057  2 IF .cnt GEQ (.bad$gl_sectors / .bad$gb_block_fact) / 2  ! If we did not manage to write at
;  952          1058  2 THEN                                                   ! least half of the last track,
;  953          1059  2     SIGNAL (bad$_rebldwarn, 1, bad$ga_device);         ! send out a warning...
;  954          1060  2
;  955          1061  1 END;     ! of ROUTINE rebuild_dbsfs


                                 00FC 00000 REBUILD_DBSFS:
                                                .WORD   Save R2,R3,R4,R5,R6,R7                  ; 1015
                               57  D4 00002     CLRL    CNT                                     ; 1040
                     0000G  CF 20  D0 00004     MOVL    #32, BAD$GL_FUNC                        ; 1044
```

```
                                56      0000G   CF   D0 00009         MOVL    BAD$GA_MDBSF, R6                                ; 1045
    0200    8F      FF  8F              6E              00   2C 0000E         MOVC5   #0, (SP), #-1, #512, (R6)
                                                        66        00016
                            0000G   CF              01   D0 00017         MOVL    #1, BAD$GL_SERIALNUM                    ; 1046
                                66                  01   7D 0001C         MOVQ    #1, (R6)
                                53      0000G   CF  9A 0001F             MOVZBL  BAD$GB_BLOCK_FACT, R3                   ; 1050
                        53      0000G   CF          53   C7 00024         DIVL3   R3, BAD$GL_SECTORS, R3
                                52                  01   CE 0002A         MNEGL   #1, OFFSET
                                                    1F   11 0002D         BRB     2$
                                0000GDF42           9F   0002F 1$:       PUSHAB  @BAD$GL_MDBSF_PTR[OFFSET]               ; 1053
                                7E      0200    8F  3C   00034         MOVZWL  #512, -(SP)                             ; 1051
                                        0000G   CF  DD   00039         PUSHL   BAD$GA_MDBSF
                            000000000G  00          03   FB 0003D         CALLS   #3, BAD$SYNC_IO
                                0000G   CF          50   D0 00044         MOVL    R0, BAD$GL_STATUS
                                02                  50   E8 00049         BLBS    R0, 2$
                                                    57   D6 0004C         INCL    CNT                                     ; 1055
                        DD              52          53   F3 0004E 2$:     AOBLEQ  R3, OFFSET, 1$                          ; 1051
                                        50      0000G   CF  9A 00052     MOVZBL  BAD$GB_BLOCK_FACT, R0                   ; 1057
                        50      0000G   CF          50   C7 00057         DIVL3   R0, BAD$GL_SECTORS, R0
                                50                  02   C6 0005D         DIVL2   #2, R0
                                50                  57   D1 00060         CMPL    CNT, R0
                                                    13   19 00063         BLSS    3$
                                        0000G   CF  9F   00065         PUSHAB  BAD$GA_DEVICE                           ; 1059
                                                    01   DD 00069         PUSHL   #1
                                    000000000G      8F   DD 0006B         PUSHL   #BAD$_REBLDWARN
                            000000000G  00          03   FB 00071         CALLS   #3, LIB$SIGNAL
                                                        04   00078 3$:       RET                                             ; 1061

; Routine Size:  121 bytes,    Routine Base:  $CODE$ + 04DF

;  956          1062  1
```

```
;  958          1063  1 %SBTTL  'get_sdbsf -- Get a valid copy of the SDBSF'
;  959          1064  1 ROUTINE get_sdbsf : NOVALUE =
;  960          1065  1 !++
;  961          1066  1 !
;  962          1067  1 ! Functional Description:
;  963          1068  1 !
;  964          1069  1 !    Obtain a  alid copy of the Software Detected Bad Sector File.
;  965          1070  1 !
;  966          1071  1 ! Implic't Inputs:
;  967          1072  1 !
;  968          1073  1 !    bad$ga_mdbsf       the address of the MDBSF buffer.
;  969          1074  1 !    bad$ga_sdbsf       the address of the SDBSF buffer.
;  970          1075  1 !    bad$gl_mdbsf_ptr   the block number of the first block on the last track.
;  971          1076  1 !    bad$gl_sdbsf_ptr   the address of the first free entry in the SDBSF buffer.
;  972          1077  1 !                                                                                ;
;  973          1078  1 ! Side Effects:
;  974          1079  1 !
;  975          1080  1 !    If we are unable to read the SDBSF, if the SDBSF is FULL, or if it does
;  976          1081  1 !    not correspond the MDBSF we will stop the analysis by issuing a FATAL      ;
;  977          1082  1 !    diagnostic.
;  978          1083  1 !
;  979          1084  1 !--
;  980          1085  2 BEGIN
;  981          1086  2 LOCAL
;  982          1087  2     value_present;
;  983          1088  2
;  984          1089  2 value_present = FALSE;                                  ! The MDBSF was found and validated,
;  985          1090  2                                                        ! now look for the SDBSF.
;  986          1091  2 INCR block FROM 10 TO .bad$gl_sectors / .bad$gb_block_fact - 1 DO
;  987          1092  3     BEGIN                                              ! Search the remainder of the last track
;  988          1093  3     IF bad$gl_status = bad$sync_io (.bad$ga_sdbsf,     ! and validate the SDBSF's header.
;  989          1094  3                                     bad$k_page_size,
;  990          1095  3                                     .bad$gl_mdbsf_ptr + .block)
;  991          1096  3     THEN
;  992          1097  4         BEGIN
;  993          1098  4         IF .bad$ga_sdbsf [ltk_l_serial_number] EQL     ! Validate the serial numbers.
;  994          1099  4            .bad$ga_mdbsf [ltk_l_serial_number]
;  995          1100  4         THEN
;  996          1101  5             BEGIN
;  997          1102  5             value_present = TRUE;                     ! They matched, the rest of the
;  998          1103  5             EXITLOOP;                                 ! header should also be good.
;  999          1104  4             END;
; 1000          1105  3         END;
; 1001          1106  2     END;
; 1002          1107  2
; 1003          1108  2 IF NOT .value_present
; 1004          1109  2 THEN
; 1005          1110  2     SIGNAL_STOP (bad$_sdbsfrfail, 1, bad$ga_device);
; 1006          1111  2
; 1007          1112  2 !
; 1008          1113  2 ! Point to the first free entry in the SDBSF.
; 1009          1114  2 !
; 1010          1115  2 value_present = FALSE;
; 1011          1116  2 INCR entry FROM .bad$ga_sdbsf TO .bad$ga_sdbsf + bad$k_page_size - 4 BY 4
; 1012          1117  2 DO
; 1013          1118  3     BEGIN
; 1014          1119  3     IF  ..entry EQL -1                                 ! An entry of -1 implies
```

```
;  1015          1120  3     THEN                                          ! the end of the bad block
;  1016          1121  4         BEGIN                                     ! information.
;  1017          1122  4         value_present = TRUE;
;  1018          1123  4         bad$gl_sdbsf_ptr = .entry;                ! Set the address into
;  1019          1124  4         EXITLOOP;                                 ! the global pointer and
;  1020          1125  3         END;                                      ! return.
;  1021          1126  2     END;
;  1022          1127  2
;  1023          1128  2 IF NOT .value_present                            ! If the bad block file
;  1024          1129  2 THEN                                            ! is full, issue a diagnostic
;  1025          1130  2     SIGNAL_STOP (bad$_sdbsffull, 1, bad$ga_device);     ! and quit.
;  1026          1131  2
;  1027          1132  1 END;     ! of ROUTINE get_sdbsf


                                        007C 00000 GET_SDBSF:
                                                   .WORD   Save R2,R3,R4,R5,R6                 ; 1064
                      56 00000000G   00  9E 00002          MOVAB   LIB$STOP, R6
                      55     0000G   CF  9E 00009          MOVAB   BAD$GA_SDBSF, R5
                      54     0000G   CF  9A 0000E          MOVZBL  BAD$GB_BLOCK_FACT, R4        ; 1091
               54     0000G   CF      54  C7 00013          DIVL3   R4, BAD$GL_SECTORS, R4
                      52             09  7D 00019          MOVQ    #9, BLOCK
                                     28  11 0001C          BRB     2$
                             0000GDF 42  9F 0001E 1$:       PUSHAB  @BAD$GL_MDBSF_PTR[BLOCK]    ; 1095
                      7E     0200    8F  3C 00023          MOVZWL  #512, -(SP)                 ; 1093
                                     65  DD 00028          PUSHL   BAD$GA_SDBSF
                00000000G    00      03  FB 0002A          CALLS   #3, BAD$SYNC_IO
                    0000G    CF      50  D0 00031          MOVL    R0, BAD$GL_STATUS
                             0D      50  E9 00036          BLBC    R0, 2$
                    0000G    DF   00    B5  D1 00039          CMPL    @BAD$GA_SDBSF, @BAD$GA_MDBSF ; 1099
                                     05  12 0003F          BNEQ    2$
                             53              01  D0 00041          MOVL    #1, VALUE_PRESENT       ; 1102
                                     04  11 00044          BRB     3$                          ; 1101
               D4            52              54  F2 00046 2$:      AOBLSS  R4, BLOCK, 1$           ; 1091
                             0F              53  E8 0004A 3$:      BLBS    VALUE_PRESENT, 4$       ; 1108
                                 0000G       CF  9F 0004D          PUSHAB  BAD$GA_DEVICE           ; 1110
                                     01      DD 00051          PUSHL   #1
                                 00000000G   8F  DD 00053          PUSHL   #BAD$_SDBSFRFAIL
                             66              03  FB 00059          CALLS   #3, LIB$STOP
                                     53      D4 0005C 4$:      CLRL    VALUE_PRESENT           ; 1115
               51            65 000001FC     8F  C1 0005E          ADDL3   #508, BAD$GA_SDBSF, R1  ; 1116
                             50              65  D0 00066          MOVL    BAD$GA_SDBSF, ENTRY
                                     16  11 00069          BRB     7$
                  FFFFFFFF   8F              60  D1 0006B 5$:      CMPL    (ENTRY), #-1            ; 1119
                                     0A  12 00072          BNEQ    6$
                             53              01  D0 00074          MOVL    #1, VALUE_PRESENT       ; 1122
                    0000G    CF              50  D0 00077          MOVL    ENTRY, BAD$GL_SDBSF_PTR ; 1123
                                     08  11 0007C          BRB     8$                          ; 1121
                             50              04  C0 0007E 6$:      ADDL2   #4, ENTRY               ; 1116
                             51              50  D1 00081 7$:      CMPL    ENTRY, R1
                                     E5  15 00084          BLEQ    5$
                             0F              53  E8 00086 8$:      BLBS    VALUE_PRESENT, 9$       ; 1128
                                 0000G       CF  9F 00089          PUSHAB  BAD$GA_DEVICE           ; 1130
                                     01      DD 0008D          PUSHL   #1
```

```
                                 00000000G  8F   DD 0008F          PUSHL   #BAD$_SDBSFFULL                                 ;
                               66           03   FB 00095          CALLS   #3, LIB$STOP                                    ;
                                            04   00098 9$:        RET                                                     ; 1132

; Routine Size:  153 bytes,    Routine Base:  $CODE$ + 0558

; 1028          1133  1
```

```
;1030     1134  1 %SBTTL  'get_nlt_sdbsf -- Get a non last track SDBSF'
;1031     1135  1 ROUTINE get_nlt_sdbsf =
;1032     1136  1 !++
;1033     1137  1 !
;1034     1138  1 ! Functional Description:
;1035     1139  1 !
;1036     1140  1 !    This is a non last track device, so search the last 256 blocks of the
;1037     1141  1 !    device for the SDBSF.  Once its found validate it by checksumming (shared
;1038     1142  1 !    routine from F11AACP) and verification of the fixed value fields of the
;1039     1143  1 !    header.
;1040     1144  1 !
;1041     1145  1 ! Implicit Inputs:
;1042     1146  1 !
;1043     1147  1 !    bad$gl_maxblock    the total number of blocks on the device.
;1044     1148  1 !    bad$ga_sdbsf       the address of the SDBSF buffer.
;1045     1149  1 !
;1046     1150  1 ! Side Effects:
;1047     1151  1 !
;1048     1152  1 !    If there is no SDBSF on the medium, an informational message will be issued
;1049     1153  1 !    and a new SDBSF will be created.
;1050     1154  1 !
;1051     1155  1 !--
;1052     1156  2 BEGIN
;1053     1157  2 LOCAL
;1054     1158  2     value_present;
;1055     1159  2
;1056     1160  2 value_present = FALSE;
;1057     1161  2 DECR block FROM .bad$gl_maxblock TO .bad$gl_maxblock - 256 DO
;1058     1162  3     BEGIN
;1059     1163  3     bad$gl_status = bad$sync_io (.bad$ga_sdbsf,
;1060     1164  3                                  bad$k_page_size,
;1061     1165  3                                 .block);
;1062     1166  3     IF .bad$gl_status
;1063     1167  3     THEN
;1064     1168  4         BEGIN
;1065     1169  4         IF CHECKSUM2 (.bad$ga_sdbsf, bad$k_page_size - 2)        ! Sum of the entries, stored in the last wor
;1066     1170  4         AND .bad$ga_sdbsf [nlt_b_cntfldsiz] EQL nlt_k_cntfldsiz ! Size of the count field in bytes.
;1067     1171  4         AND .bad$ga_sdbsf [nlt_b_adrfldsiz] EQL nlt_k_adrfldsiz ! Size of the address field in bytes.
;1068     1172  4         AND .bad$ga_sdbsf [nlt_b_usedbbdsc] LEQ nlt_k_available ! Number of used bad block descriptor
;1069     1173  4         THEN                                                    ! (the count is in WORDs!)
;1070     1174  5             BEGIN
;1071     1175  5             value_present = TRUE;                               ! We found one, return.
;1072     1176  5             EXITLOOP;
;1073     1177  4             END;
;1074     1178  3         END;
;1075     1179  2     END;
;1076     1180  2
;1077     1181  2 IF NOT .value_present                                           ! If the SDBSF is not available, issue
;1078     1182  2 THEN                                                           ! a diagnostic and create a new one.
;1079     1183  3     BEGIN
;1080     1184  3     SIGNAL (bad$_nobadinfo, 1, bad$ga_device);
;1081     1185  3     init_sdbsf ();
;1082     1186  2     END;
;1083     1187  2
;1084     1188  2 !
;1085     1189  2 ! Point to the first free entry in the SDBSF before returning.
;1086     1190  2 !
```

```
;1087          1191  2 bad$gl_sdbsf_ptr = .bad$ga_sdbsf + nlt_k_headersiz +
;1088          1192  2                    (.bad$ga_sdbsf [nlt_b_usedbbdsc] * 2);
;1089          1193  2
;1090          1194  2 RETURN .value_present;
;1091          1195  2
;1092          1196  1 END;     ! of ROUTINE get_nlt_sdbsf
```

```
                                    003C 00000 GET_NLT_SDBSF:
                                                .WORD   Save R2,R3,R4,R5                ;1135
                        55      0000G   CF  9E 00002        MOVAB   BAD$GA_SDBSF, R5
                                        53  D4 00007        CLRL    VALUE_PRESENT       ;1160
                54      C000.   CF 00000100 8F C3 00009     SUBL3   #256, BAD$GL_MAXBLOCK, R4   ;1161
                        52      0000G   CF  D0 00013        MOVL    BAD$GL_MAXBLOCK, BLOCK
                                        4B  11 00018        BRB     3$
                                        52  DD 0001A 1$:     PUSHL   BLOCK               ;1165
                        7E      0200    8F  3C 0001C        MOVZWL  #512, -(SP)         ;1163
                                        65  DD 00021        PUSHL   BAD$GA_SDBSF
                00000000G       00      03  FB 00023        CALLS   #3, BAD$SYNC_IO
                    0000G       CF      50  D0 0002A        MOVL    R0, BAD$GL_STATUS
                        2F      0000G   CF  E9 0002F        BLBC    BAD$GL_STATUS, 2$   ;1166
                        7E      01FE    8F  3C 00034        MOVZWL  #510, -(SP)         ;1169
                                        65  DD 00039        PUSHL   BAD$GA_SDBSF
                00000000G       00      02  FB 0003B        CALLS   #2, CHECKSUM2
                                1E      50  E9 00042        BLBC    R0, 2$
                                01      00  B5 91 00045     CMPB    @BAD$GA_SDBSF, #1   ;1170
                                            18 12 00049     BNEQ    2$
                                50          65 D0 0004B     MOVL    BAD$GA_SDBSF, R0    ;1171
                                03      01  A0 91 0004E     CMPB    1(R0), #3
                                            0F 12 00052     BNEQ    2$
                                50          65 D0 00054     MOVL    BAD$GA_SDBSF, R0    ;1172
                            FC  8F      02  A0 91 00057     CMPB    2(R0), #252
                                            05 1A 0005C     BGTRU   2$                  ;
                                53          01 D0 0005E     MOVL    #1, VALUE_PRESENT   ;1175
                                            07 11 00061     BRB     4$                  ;1174
                                            52 D7 00063 2$: DECL    BLOCK               ;1161 ;
                                54          52 D1 00065 3$: CMPL    BLOCK, R4
                                            B0 18 00068     BGEQ    1$
                                18          53 E8 0006A 4$: BLBS    VALUE_PRESENT, 5$   ;1181
                                    0000G   CF  9F 0006D    PUSHAB  BAD$GA_DEVICE       ;1184
                                            01 DD 00071     PUSHL   #1
                                00000000G   8F  DD 00073    PUSHL   #BAD$_NOBADINFO
                        00000000G   00      03  FB 00079    CALLS   #3, LIB$SIGNAL
                            0000V   CF      00  FB 00080    CALLS   #0, INIT_SDBSF      ;1185
                                51          65 D0 00085 5$: MOVL    BAD$GA_SDBSF, R1    ;1191
                                50      02  A1 9A 00088     MOVZBL  2(R1), R0           ;1192
                            0000G   CF      04 A140 3E 0008C    MOVAW   4(R1)[R0], BAD$GL_SDBSF_PTR  ;1191
                                50          53 D0 00093     MOVL    VALUE_PRESENT, R0   ;1194
                                            04 00096        RET                         ;1196

; Routine Size:  151 bytes,    Routine Base:  $CODE$ + 05F1

;1093          1197  1
```

```
;1095     1198  1 %SBTTL  'init_sdbsf -- Create and initialize the header of the SDBSF'
;1096     1199  1 ROUTINE init_sdbsf : NOVALUE =
;1097     1200  1 !++
;1098     1201  1 !
;1099     1202  1 ! Functional Description:
;1100     1203  1 !
;1101     1204  1 !    This procedure will initialize the contents of a new SDBSF.  The pointer
;1102     1205  1 !    to the first free entry will also be set.
;1103     1206  1 !
;1104     1207  1 ! Implicit Inputs:
;1105     1208  1 !
;1106     1209  1 !    bad$ga_sdbsf       the address of the SDBSF buffer.
;1107     1210  1 !
;1108     1211  1 ! Side Effects:
;1109     1212  1 !
;1110     1213  1 !    The header fields of the selected format bad block buffer are initialized
;1111     1214  1 !    to the statically defined values.
;1112     1215  1 !
;1113     1216  1 !--
;1114     1217  2 BEGIN
;1115     1218  2
;1116     1219  2 IF .bad$gl_context [ctx_v_ltdevice]
;1117     1220  2 THEN
;1118     1221  3     BEGIN       ! Last track device (copy static fields from the MDBSF).
;1119     1222  3     bad$ga_sdbsf [ltk_l_serial_number] = .bad$ga_mdbsf [ltk_l_serial_number];
;1120     1223  3     bad$ga_sdbsf [ltk_w_unused] = .bad$ga_mdbsf [ltk_w_unused];
;1121     1224  3     bad$ga_sdbsf [ltk_w_cartridge] = .bad$ga_mdbsf [ltk_w_cartridge];
;1122     1225  3     bad$gl_sdbsf_ptr = .bad$ga_sdbsf + ltk_k_headersiz;
;1123     1226  3     END
;1124     1227  2 ELSE
;1125     1228  3     BEGIN       ! Non Last track device (use static defaults).
;1126     1229  3     bad$ga_sdbsf [nlt_b_cntfldsiz] = nlt_k_cntfldsiz;   ! Fill in the count field size.
;1127     1230  3     bad$ga_sdbsf [nlt_b_adrfldsiz] = nlt_k_adrfldsiz;   ! Fill in the address field size.
;1128     1231  3     bad$ga_sdbsf [nlt_b_usedbbdsc] = 0;                 ! No descriptors used.
;1129     1232  3     bad$ga_sdbsf [nlt_b_available] = nlt_k_available;   ! Total number of available descriptors.
;1130     1233  3     bad$gl_sdbsf_ptr = .bad$ga_sdbsf + nlt_k_headersiz;
;1131     1234  2     END;
;1132     1235  2
;1133     1236  1 END;     ! of ROUTINE init_sdbsf


                                        0000 00000 INIT_SDBSF:
                                                           .WORD   Save nothing                                ;1199
                                50      0000G   CF   D0 00002     MOVL    BAD$GA_SDBSF, R0                   ;1222
                                0F      0000G   CF   E9 00007     BLBC    BAD$GL_CONTEXT+1, 1$               ;1219
                                51      0000G   CF   D0 0000C     MOVL    BAD$GA_MDBSF, R1                   ;1222
                                60              61   7D 00011     MOVQ    (R1), (R0)                         ;
                          0000G CF        08    A0   9E 00014     MOVAB   8(R0), BAD$GL_SDBSF_PTR            ;1225
                                                04 0001A         RET                                        ;1219
                                60 FC000301     8F   D0 0001B 1$:  MOVL   #-67108095, (R0)                  ;1229
                          0000G CF        04    A0   9E 00022     MOVAB   4(R0), BAD$GL_SDBSF_PTR            ;1233
                                                04 00028         RET                                        ;1236

; Routine Size:  41 bytes,    Routine Base:  $CODE$ + 0688
```

```
; 1134          1237  1
```

```
;  1136          1238  1 %SBTTL  'get_pattern -- Process user supplied test pattern(s)'
;  1137          1239  1 ROUTINE get_pattern : NOVALUE =
;  1138          1240  1 !++
;  1139          1241  1 !
;  1140          1242  1 ! Functional Description:
;  1141          1243  1 !
;  1142          1244  1 !    Process the user supplied test pattern(s).
;  1143          1245  1 !
;  1144          1246  1 ! Implicit Inputs:
;  1145          1247  1 !
;  1146          1248  1 !    bad$gl_bad_term            the vector into which the pattern(s)
;  1147          1249  1 !                               will be loaded.
;  1148          1250  1 !
;  1149          1251  1 !    bad$gb_term_count          the number of longwords which make
;  1150          1252  1 !                               up the pattern.
;  1151          1253  1 !
;  1152          1254  1 ! Side Effects:
;  1153          1255  1 !
;  1154          1256  1 !    If the user neglects to supply at least one pattern, the context bit
;  1155          1257  1 !    (ctx_v_pattern) will be cleared, and the default pattern(s) will be used.
;  1156          1258  1 !
;  1157          1259  1 !--
;  1158          1260  2 BEGIN
;  1159          1261  2 bad$gl_context [ctx_v_pattern] = FALSE;                ! Assume the default.
;  1160          1262  2 bad$gl_bad_term [0] = bad$gl_bad_term [1] = 0;         ! Clear the vector.
;  1161          1263  2 bad$gl_bad_term [2] = bad$gl_bad_term [3] = 0;         !
;  1162          1264  2 bad$gb_term_count = 0;                                ! Clear the term count.
;  1163          1265  2
;  1164          1266  2 WHILE CLI$GET_VALUE (pattern, bad$ga_input_desc) DO   ! Process all supplied values.
;  1165          1267  3     BEGIN                                             !
;  1166          1268  3     bad$gl_context [ctx_v_pattern] = TRUE;            ! Flag that there are valid patterns available.
;  1167          1269  3     parse_value (number_tbl, numkey_tbl);             ! Parse the string into a value.
;  1168          1270  2     END;
;  1169          1271  2
;  1170          1272  1 END;     ! of ROUTINE get_pattern


                                        0000 00000 GET_PATTERN:
                                                   .WORD   Save nothing                                    ;  1239
                    0000G  CF                04    8A 00002          BICB2   #4, BAD$GL_CONTEXT+1          ;  1261
                                   0000G  CF   7C    7C 00007          CLRQ    BAD$GL_BAD_TERM               ;  1262
                                   0000G  CF   7C    7C 0000B          CLRQ    BAD$GL_BAD_TERM+8             ;  1263
                                   0000G  CF   94    94 0000F          CLRB    BAD$GB_TERM_COUNT             ;  1264
                                   0000G  CF   9F    9F 00013 1$:      PUSHAB  BAD$GA_INPUT_DESC             ;  1266
                                   0000'  CF   9F    9F 00017          PUSHAB  PATTERN                       ;
                00000000G  00              02    FB 0001B          CALLS   #2, CLI$GET_VALUE             ;
                           14              50    E9 00022          BLBC    R0, 2$                        ;
                    0000G  CF              04    88 00025          BISB2   #4, BAD$GL_CONTEXT+1          ;  1268
                                   0000'  CF   9F    9F 0002A          PUSHAB  NUMKEY_TBL                    ;  1269
                                   0000'  CF   9F    9F 0002E          PUSHAB  NUMBER_TBL                    ;
                    0000V  CF              02    FB 00032          CALLS   #2, PARSE_VALUE               ;
                                           DA    11 00037          BRB     1$                            ;  1266
                                                 04 00039 2$:      RET                                   ;  1272
```

```
; Routine Size:  58 bytes,    Routine Base:  $CODE$ + 06B1

; 1171          1273  1
```

```
;  1173          1274  1 %SBTTL  'process_badblocks -- Process user supplied bad block(s)'
;  1174          1275  1 ROUTINE process_badblocks =
;  1175          1276  1 !++
;  1176          1277  1 !
;  1177          1278  1 ! Functional Description:
;  1178          1279  1 !
;  1179          1280  1 !     Process the user supplied bad block information.  This is accomplished
;  1180          1281  1 !     by parsing the bad block(s) supplied by the user, validating them
;  1181          1282  1 !     against the device geometry, then if they are not already recorded in
;  1182          1283  1 !     a bad block file, they will be inserted.
;  1183          1284  1 !
;  1184          1285  1 ! Implicit Inputs:
;  1185          1286  1 !
;  1186          1287  1 !     bad$gl_bad_term             the vector containing the terms used to describe
;  1187          1288  1 !                                 the bad block(s).
;  1188          1289  1 !     bad$gb_term_count           the number of term used to describe the bad
;  1189          1290  1 !                                 block(s).                                          ;
;  1190          1291  1 !
;  1191          1292  1 ! Side Effects:
;  1192          1293  1 !                                                                                    ;
;  1193          1294  1 !     If the block(s) specified are not already recorded in the bad block file(s)
;  1194          1295  1 !     they will be added.  If there are duplicate bad blocks specified, they will
;  1195          1296  1 !     cause a warning message to be displayed.
;  1196          1297  1 !
;  1197          1298  1 !     If we are prompting for the bad blocks, then if an invalid block number or
;  1198          1299  1 !     grammar error is encountered, we will issue a diagnostic, disregard the
;  1199          1300  1 !     last entry and prompt again.  If not prompting for bad blocks (they were
;  1200          1301  1 !     specified in the command), then we will issue a FATAL error diagnostic,
;  1201          1302  1 !     terminating the analysis.
;  1202          1303  1 !
;  1203          1304  1 !--
;  1204          1305  2 BEGIN
;  1205          1306  2 OWN
;  1206          1307  2     hibound,
;  1207          1308  2     lobound;
;  1208          1309  2
;  1209          1310  2 hibound = lobound = 0;                                  ! Assure we start out fresh.
;  1210          1311  2 bad$gb_term_count = 0;                                  ! Reset the count of terms
;  1211          1312  2 bad$gl_bad_term [0] = bad$gl_bad_term [1] = 0;          ! and each of the terms also.
;  1212          1313  2 bad$gl_bad_term [2] = bad$gl_bad_term [3] = 0;
;  1213          1314  2 parse_value (state_tbl, key_tbl);                       ! Parse the bad block info.
;  1214          1315  2
;  1215          1316  2 IF .bad$gb_term_count EQL 0                            ! Did we encounter a syntax error?
;  1216          1317  2 THEN                                                    ! Yes, so return for more input
;  1217          1318  2     RETURN FALSE;                                       ! (if any) to process.
;  1218          1319  2 !
;  1219          1320  2 ! Validate the address(es).  If validation fails, the validation routine
;  1220          1321  2 ! will depending on the processing context (interactive or command)
;  1221          1322  2 ! signal an error or halt execution of the image via a modification of
;  1222          1323  2 ! the message severity (set to STS$K_SEVERE).
;  1223          1324  2 !
;  1224          1325  2 IF .bad$gb_term_count GEQ 3                            ! Is the address in physical
;  1225          1326  2 THEN                                                    ! or logical address format?
;  1226          1327  3     BEGIN
;  1227          1328  3     IF NOT val_phy_adr (.bad$gl_bad_term [0],          ! Sector
;  1228          1329  3                         .bad$gl_bad_term [1],          ! Track
;  1229          1330  3                         .bad$gl_bad_term [2],          ! Cylinder
```

```
;  1230          1331  3                               .bad$gl_bad_term [3])           ! Count
;  1231          1332  3      THEN
;  1232          1333  3          RETURN FALSE;
;  1233          1334  3
;  1234          1335  3      lobound = bad$cvt_phy_log (.bad$gl_bad_term [0],     ! This is a valid address, form
;  1235          1336  3                                 .bad$gl_bad_term [1],     ! its equivalent logical block
;  1236          1337  3                                 .bad$gl_bad_term [2]);    ! number.
;  1237          1338  3
;  1238          1339  3      hibound = .lobound + .bad$gl_bad_term [3] - 1;
;  1239          1340  3      END
;  1240          1341  2  ELSE
;  1241          1342  3      BEGIN
;  1242          1343  3      IF NOT val_log_adr (.bad$gl_bad_term [0],           ! Logical Block Number
;  1243          1344  3                         .bad$gl_bad_term [1])           ! Count
;  1244          1345  3      THEN
;  1245          1346  3          RETURN FALSE;
;  1246          1347  3
;  1247          1348  3      lobound = .bad$gl_bad_term [0];                     ! Set the address bounds.
;  1248          1349  3      hibound = .lobound + .bad$gl_bad_term [1] - 1;
;  1249          1350  2      END;
;  1250          1351  2
;  1251          1352  2  IF .hibound LSS .lobound                               ! Prevent mess up for one block,
;  1252          1353  2  THEN                                                  ! the hi must be same as low.
;  1253          1354  2      hibound = .lobound;
;  1254          1355  2  !
;  1255          1356  2  ! At this point we are satisfied with the addresses specified.  Determine
;  1256          1357  2  ! if the media is last track or non last track and look for a duplicate block.
;  1257          1358  2  ! If no duplicate is found, the block(s) will be entered in the SDBSF.
;  1258          1359  2  !
;  1259          1360  2  IF .bad$gl_context [ctx_v_ltdevice]
;  1260          1361  2  THEN
;  1261          1362  2      bad$check_ltk (.lobound, .hibound)
;  1262          1363  2  ELSE
;  1263          1364  2      bad$check_nlt (.lobound, .hibound);
;  1264          1365  2
;  1265          1366  2  RETURN TRUE;
;  1266          1367  2
;  1267          1368  1  END;     !of ROUTINE process_badblocks


                                                .PSECT  $OWN$,NOEXE,2

                                    00000 HIBOUND:.BLKB   4
                                    00004 LOBOUND:.BLKB   4


                                                .PSECT  $CODE$,NOWRT,2

                                    000C 00000 PROCESS_BADBLOCKS:
                                                .WORD   Save R2,R3                      ;  1275
                                53      0000G   CF  9E 00002    MOVAB   BAD$GL_BAD_TERM, R3     ;
                                52      0000'   CF  9E 00007    MOVAB   HIBOUND, R2             ;
                                                62  7C 0000C    CLRQ    HIBOUND                 ;  1310
                                        0000G   CF  94 0000E    CLRB    BAD$GB_TERM_COUNT       ;  1311
                                                63  7C 00012    CLRQ    BAD$GL_BAD_TERM         ;  1312
```

```
                                 08   A3  7C 00014          CLRQ    BAD$GL_BAD_TERM+8                       ; 1313
                           0000'  CF  9F 00017          PUSHAB  KEY_TBL                                 ; 1314
                           0000'  CF  9F 0001B          PUSHAB  STATE_TBL                               ;
                    0000V  CF         02  FB 0001F          CALLS   #2, PARSE_VALUE                         ;
                           50      0000G  CF  9A 00024          MOVZBL  BAD$GB_TERM_COUNT, R0           ; 1316
                                          73  13 00029          BEQL    6$                              ;
                           03             50  91 0002B          CMPB    R0, #3                          ; 1325
                                          28  1F 0002E          BLSSU   1$                              ;
                           7E      08     A3  7D 00030          MOVQ    BAD$GL_BAD_TERM+8, -(SP)        ; 1330
                           7E             63  7D 00034          MOVQ    BAD$GL_BAD_TERM, -(SP)          ; 1328
                    0000V  CF             04  FB 00037          CALLS   #4, VAL_PHY_ADR                 ;
                           5F             50  E9 0003C          BLBC    R0, 6$                          ;
                           7E      04     A3  7D 0003F          MOVQ    BAD$GL_BAD_TERM+4, -(SP)        ; 1336
                                          63  DD 00043          PUSHL   BAD$GL_BAD_TERM                 ; 1335
              00000000G    00             03  FB 00045          CALLS   #3, BAD$CVT_PHY_LOG             ;
                      04   A2             50  D0 0004C          MOVL    R0, LOBOUND                     ;
                50    04   A2      0C     A3  C1 00050          ADDL3   BAD$GL_BAD_TERM+12, LOBOUND, R0 ; 1339
                                          15  11 00056          BRB     2$                              ;
                           7E             63  7D 00058  1$:     MOVQ    BAD$GL_BAD_TERM, -(SP)          ; 1343
                    0000V  CF             02  FB 0005B          CALLS   #2, VAL_LOG_ADR                 ;
                           3B             50  E9 00060          BLBC    R0, 6$                          ;
                      04   A2             63  D0 00063          MOVL    BAD$GL_BAD_TERM, LOBOUND        ; 1348
                50    04   A2      04     A3  C1 00067          ADDL3   BAD$GL_BAD_TERM+4, LOBOUND, R0  ; 1349
                           62      FF     A0  9E 0006D  2$:     MOVAB   -1(R0), HIBOUND                 ;
                           50      04     A2  D0 00071          MOVL    LOBOUND, R0                     ; 1352
                           50             62  D1 00075          CMPL    HIBOUND, R0                     ;
                                          03  18 00078          BGEQ    3$                              ;
                           62             50  D0 0007A          MOVL    R0, HIBOUND                     ; 1354
                           0D      0000G  CF  E9 0007D  3$:     BLBC    BAD$GL_CONTEXT+1, 4$            ; 1360
                                          62  DD 00082          PUSHL   HIBOUND                         ; 1362
                                          50  DD 00084          PUSHL   R0                              ;
              00000000G    00             02  FB 00086          CALLS   #2, BAD$CHECK_LTK               ;
                                          0B  11 0008D          BRB     5$                              ;
                                          62  DD 0008F  4$:     PUSHL   HIBOUND                         ; 1364
                                          50  DD 00091          PUSHL   R0                              ;
              00000000G    00             02  FB 00093          CALLS   #2, BAD$CHECK_NLT               ;
                           50             01  D0 0009A  5$:     MOVL    #1, R0                          ; 1366
                                          04  00 0009D          RET                                     ;
                                          50  D4 0009E  6$:     CLRL    R0                              ; 1368
                                          04  00 000A0          RET                                     ;

; Routine Size:  161 bytes,    Routine Base:  $CODE$ + 06EB

; 1268          1369  1
```

```
;1270     1370  1 %SBTTL  'parse_value -- Parse and process user supplied value(s)'
;1271     1371  1 ROUTINE parse_value (state, key) : NOVALUE =
;1272     1372  1 !++
;1273     1373  1 !
;1274     1374  1 ! Functional Description:
;1275     1375  1 !
;1276     1376  1 !    This routine is responsible for parsing user supplied values which
;1277     1377  1 !    are used by /BAD_BLOCKS, /EXERCISE=PATTERN, etc..
;1278     1378  1 !
;1279     1379  1 ! Inputs:
;1280     1380  1 !
;1281     1381  1 !    state       adr     address of the state table to use.
;1282     1382  1 !    key         adr     address of the key table to use.
;1283     1383  1 !
;1284     1384  1 ! Implicit Inputs:
;1285     1385  1 !
;1286     1386  1 !    The address of the TPARSE control block.
;1287     1387  1 !
;1288     1388  1 !    The descriptor pointer to the string to be parsed (bad$ga_input_desc).
;1289     1389  1 !
;1290     1390  1 ! Implicit Outputs:
;1291     1391  1 !
;1292     1392  1 !    The information vector bad$gl_bad_term [x] will contain the user
;1293     1393  1 !    supplied data values.
;1294     1394  1 !
;1295     1395  1 !--
;1296     1396  2 BEGIN
;1297     1397  2 LOCAL
;1298     1398  2     tparse_blk  : BLOCK [TPA$K_LENGTH0, BYTE];          ! TPARSE control block.
;1299     1399  2
;1300     1400  2 IF bad$str_trim NEQ 0                                  ! Trim the input line if we are stand alone
;1301     1401  2 THEN    bad$str_trim (bad$ga_input_desc);               ! to preclude erroneous errors during the parse.
;1302     1402  2
;1303     1403  2 tparse_blk [tpa$l_count] = TPA$K_COUNT0;               ! Initial longword count.
;1304     1404  2 tparse_blk [tpa$l_options] = TPA$M_ABBREV;             ! Allow abbreviations.
;1305     1405  2 tparse_blk [tpa$l_stringcnt] = .bad$ga_input_desc [dsc$w_length];
;1306     1406  2 tparse_blk [tpa$l_stringptr] = .bad$ga_input_desc [dsc$a_pointer];
;1307     1407  2
;1308     1408  2 LIB$TPARSE (tparse_blk, .state, .key);                  ! Parse the value.
;1309     1409  2
;1310     1410  2 IF bad$str_trim NEQ 0                                  ! Restore the buffer size if we are stand alone
;1311     1411  2 THEN    bad$ga_input_desc [dsc$w_length] = bad$k_input_len;
;1312     1412  2
;1313     1413  1 END;    ! of ROUTINE parse_value


                         000C 00000 PARSE_VALUE:
                                    .WORD   Save R2,R3                                  ;1371
53      0000G   CF  9E 00002        MOVAB   BAD$GA_INPUT_DESC, R3                       ;
5E              24  C2 00007        SUBL2   #36, SP                                     ;
50      0000G   CF  9E 0000A        MOVAB   BAD$STR_TRIM, R0                            ;1400
                52  D4 0000F        CLRL    R2                                          ;
                50  D5 00011        TSTL    R0                                          ;
                09  13 00013        BEQL    1$                                          ;
```

```
                                                52  D6 00015          INCL    R2                                                  ;
                                                53  DD 00017          PUSHL   R3                                                  ; 1401
                                      0000G     CF  01  FB 00019          CALLS   #1, BAD$STR_TRIM                                    ;
                                                6E  08  D0 0001E 1$:      MOVL    #8, TPARSE_BLK                                      ; 1403
                                          04    AE  02  D0 00021          MOVL    #2, TPARSE_BLK+4                                    ; 1404
                                          08    AE  63  3C 00025          MOVZWL  BAD$GA_INPUT_DESC, TPARSE_BLK+8                     ; 1405
                                          0C    AE  04  A3  D0 00029      MOVL    BAD$GA_INPUT_DESC+4, TPARSE_BLK+12                  ; 1406
                                                7E  04  AC  7D 0002E      MOVQ    STATE, -(SP)                                        ; 1408
                                                    08  AE  9F 00032      PUSHAB  TPARSE_BLK                                          ;
                                  00000000G     00      03  FB 00035      CALLS   #3, LIB$TPARSE                                      ;
                                                04      52  E9 0003C      BLBC    R2, 2$                                              ; 1410
                                                63  84  8F  9B 0003F      MOVZBW  #132, BAD$GA_INPUT_DESC                             ; 1411
                                                        04  00043 2$:      RET                                                         ; 1413

; Routine Size:  68 bytes,    Routine Base:  $CODE$ + 078C

; 1314          1414  1
```

```
;  1316          1415  1 %SBTTL 'TPARSE action routines'
;  1317          1416  1 ROUTINE count_term =
;  1318          1417  1 !++
;  1319          1418  1 !
;  1320          1419  1 ! Functional Description:
;  1321          1420  1 !
;  1322          1421  1 !    This routine is called by TPARSE to indicate the number
;  1323          1422  1 !    of terms which describe the user supplied bad block(s).
;  1324          1423  1 !
;  1325          1424  1 ! Implicit Outputs:
;  1326          1425  1 !
;  1327          1426  1 !    bad$gl_bad_term          The appropriate longword in this vector is
;  1328          1427  1 !                             loaded with the current value of the parse.
;  1329          1428  1 !
;  1330          1429  1 !    bad$gb_term_count        The count value is used after parsing the
;  1331          1430  1 !                             current term to describe the form of bad block
;  1332          1431  1 !                             information specified.
;  1333          1432  1 !
;  1334          1433  1 !                             1 => lbn
;  1335          1434  1 !                             2 => lbn:count
;  1336          1435  1 !                             3 => sector.track.cylinder
;  1337          1436  1 !                             4 => sector.track.cylinder:count
;  1338          1437  1 !
;  1339          1438  1 !--
;  1340          1439  2 BEGIN
;  1341          1440  2
;  1342          1441  2 def_tparse_args;
;  1343          1442  2
;  1344          1443  2 bad$gl_bad_term [.bad$gb_term_count] = .tparse_args [TPA$L_NUMBER];
;  1345          1444  2 bad$gb_term_count = .bad$gb_term_count + 1;
;  1346          1445  2
;  1347          1446  2 IF .bad$gb_term_count GTR 4
;  1348          1447  2 THEN RETURN FALSE;
;  1349          1448  2
;  1350          1449  2 RETURN TRUE;
;  1351          1450  2
;  1352          1451  1 END;     ! of ROUTINE count_term


                                 0004 00000 COUNT_TERM:
                                                .WORD   Save R2                                         ;  1416
                       52  0000G   CF  9E 00002          MOVAB   BAD$GB_TERM_COUNT, R2
                       50          62  9A 00007          MOVZBL  BAD$GB_TERM_COUNT, R0                   ;  1443
                0000GCF40    1C    AC  D0 0000A          MOVL    28(TPARSE_ARGS), BAD$GL_BAD_TERM[R0]
                                   62  96 00011          INCB    BAD$GB_TERM_COUNT                       ;  1444
                       04          62  91 00013          CMPB    BAD$GB_TERM_COUNT, #4                   ;  1446
                                   04  1A 00016          BGTRU   1$
                       50          01  D0 00018          MOVL    #1, R0                                  ;  1449
                                   04  0001B          RET
                                   50  D4 0001C 1$:       CLRL    R0                                      ;  1451
                                   04  0001E          RET

; Routine Size:  31 bytes,    Routine Base:  $CODE$ + 07D0
```

; 1353 1452 1

```
;  1355          1453  1 ROUTINE syntax_err =
;  1356          1454  1 !++
;  1357          1455  1 !
;  1358          1456  1 ! Functional Description:
;  1359          1457  1 !
;  1360          1458  1 !    This routine issues a diagnotic to inform the user of an invalid
;  1361          1459  1 !    argument in the bad block specification.  The term count is reset
;  1362          1460  1 !    to indicate an invalid bad block specification was encountered.
;  1363          1461  1 !
;  1364          1462  1 ! Implicit Outputs:
;  1365          1463  1 !
;  1366          1464  1 !    bad$gb_term_count          is reset, to indicate no value present.
;  1367          1465  1 !
;  1368          1466  1 ! Side Effects:
;  1369          1467  1 !
;  1370          1468  1 !    Since there was a missing argument in this block spec, it constitutes
;  1371          1469  1 !    an unrecoverable error.  If we are prompting for bad blocks, issue the
;  1372          1470  1 !    diagnostic and prompt for the next block.  If we are processing bad
;  1373          1471  1 !    blocks issued from the command line, then terminate the analysis after
;  1374          1472  1 !    issuing the diagnostic.
;  1375          1473  1 !
;  1376          1474  1 !--
;  1377          1475  2 BEGIN
;  1378          1476  2 bad$gb_term_count = 0;
;  1379          1477  2
;  1380          1478  2 SIGNAL (IF .bad$gl_context [ctx_v_interactive]
;  1381          1479  2         THEN bad$_ivblkent
;  1382          1480  2         ELSE bad$_ivblkent + STS$K_SEVERE,
;  1383          1481  2         1, bad$ga_input_desc);
;  1384          1482  2
;  1385          1483  2 RETURN  TRUE;
;  1386          1484  2
;  1387          1485  1 END;    ! of ROUTINE syntax_err


                                        0000 00000 SYNTAX_ERR:
                                                           .WORD   Save nothing                                    ;  1453
                                    0000G   CF   94 00002          CLRB    BAD$GB_TERM_COUNT                       ;  1476
                                    0000G   CF   9F 00006          PUSHAB  BAD$GA_INPUT_DESC                       ;  1478
                                            01   DD 0000A          PUSHL   #1                                      ;
                          08        0000G   CF                    05   E1 0000C          BBC     #5, BAD$GL_CONTEXT, 1$  ;
                                                00000000G   8F     DD 00012          PUSHL   #BAD$_IVBLKENT          ;
                                                            06     11 00018          BRB     2$                      ;
                                                00000000G   8F     DD 0001A 1$:      PUSHL   #BAD$_IVBLKENT+4        ;  1480
                             00000000G  00                  03     FB 00020 2$:      CALLS   #3, LIB$SIGNAL          ;  1478
                                        50                  01     D0 00027          MOVL    #1, R0                  ;  1483
                                                            04     0002A          RET                             ;  1485

; Routine Size:  43 bytes,    Routine Base:  $CODE$ + 07EF

;  1388          1486  1
```

```
;1390        1487  1 ROUTINE bad_value =
;1391        1488  1 !++
;1392        1489  1 !
;1393        1490  1 ! Functional Description:
;1394        1491  1 !
;1395        1492  1 !    This routine issues a diagnotic to inform the user of an invalid
;1396        1493  1 !    value on the PATTERN keyword.
;1397        1494  1 !
;1398        1495  1 ! Side Effects:
;1399        1496  1 !
;1400        1497  1 !    Since there was an invalid pattern value, it constitutes an unrecoverable
;1401        1498  1 !    error.  Since we are processing the value from the command line, terminate
;1402        1499  1 !    the analysis after issuing the diagnostic.
;1403        1500  1 !
;1404        1501  1 !--
;1405        1502  2 BEGIN
;1406        1503  2
;1407        1504  2 SIGNAL (bad$_badvalue, 1, bad$ga_input_desc);
;1408        1505  2 RETURN TRUE;
;1409        1506  2
;1410        1507  1 END;    ! of ROUTINE bad_value


                                0000 00000 BAD_VALUE:
                                           .WORD   Save nothing                                    ;1487
                          0000G CF  9F 00002       PUSHAB  BAD$GA_INPUT_DESC                       ;1504
                                01  DD 00006       PUSHL   #1
                    00000000G   8F  DD 00008       PUSHL   #BAD$_BADVALUE
          00000000G   00            03  FB 0000E   CALLS   #3, LIB$SIGNAL
                      50            01  D0 00015   MOVL    #1, R0                                  ;1505
                                    04  00018     RET                                             ;1507

; Routine Size:  25 bytes,    Routine Base:  $CODE$ + 081A

;1411        1508  1
```

```
;1413           1509  1 %SBTTL  'val_phy_adr -- Validate Physical Address'
;1414           1510  1 ROUTINE val_phy_adr (sec, trk, cyl, cnt) =
;1415           1511  1 !++
;1416           1512  1 !
;1417           1513  1 ! Functional Description:
;1418           1514  1 !
;1419           1515  1 !    Validate Physical Address Range
;1420           1516  1 !
;1421           1517  1 ! Inputs:
;1422           1518  1 !
;1423           1519  1 !    sec         val     the sector number supplied by the user.
;1424           1520  1 !    trk         val     the track number supplied by the user.
;1425           1521  1 !    cyl         val     the cylinder number supplied by the user.
;1426           1522  1 !    cnt         val     the count of contigous sectors to mark bad.
;1427           1523  1 !
;1428           1524  1 ! Implicit Inputs:
;1429           1525  1 !
;1430           1526  1 !    bad$gl_cylinders    the number of cylinders on the device.
;1431           1527  1 !    bad$gl_maxblock     the total number of blocks on the device.
;1432           1528  1 !    bad$gl_sectors      the number of sectors per track on the device.
;1433           1529  1 !    bad$gl_tracks       the number of tracks per cylinder on the device.
;1434           1530  1 !
;1435           1531  1 ! Side Effects:
;1436           1532  1 !
;1437           1533  1 !    If any of the parts which constitute the physical address are not
;1438           1534  1 !    valid, we will issue a diagnostic and select on of the following actions.
;1439           1535  1 !
;1440           1536  1 !       If prompting, disregard the last entry and prompt for more input.
;1441           1537  1 !
;1442           1538  1 !       If processing blocks from the command line, the severity will be set
;1443           1539  1 !       to FATAL, resulting in termination of the analysis.
;1444           1540  1 !
;1445           1541  1 !--
;1446           1542  2 BEGIN
;1447           1543  2 OWN
;1448           1544  2     hibound,
;1449           1545  2     status;
;1450           1546  2
;1451           1547  2 status = TRUE;                                          ! Assume address is ok.
;1452           1548  2
;1453           1549  2 IF .sec GTR .bad$gl_sectors - 1                        ! Is the sector number
;1454           1550  2 THEN                                                   ! within range?
;1455           1551  2     status = addr_error (sector, .sec);
;1456           1552  2
;1457           1553  2 IF .trk GTR .bad$gl_tracks - 1                        ! Is the track number
;1458           1554  2 THEN                                                   ! within range?
;1459           1555  2     status = addr_error (track, .trk);
;1460           1556  2
;1461           1557  2 IF .cyl GTR .bad$gl_cylinders - 1                     ! Is the cylinder number
;1462           1558  2 THEN                                                   ! within range?
;1463           1559  2     status = addr_error (cylinder, .cyl);
;1464           1560  2
;1465           1561  2 IF .cnt NEQ 0                                         ! Should we consider the count field?
;1466           1562  2 THEN
;1467           1563  3     BEGIN                                             ! Yes.
;1468           1564  3     hibound = bad$cvt_phy_log(.sec, .trk, .cyl);      ! Convert to logical address.
;1469           1565  3     IF .hibound + .cnt GTR .bad$gl_maxblock - 1       ! Does the COUNT field
```

```
; 1470     1566  3     THEN                                                    ! push us over the end
; 1471     1567  3         status = addr_error (count, .cnt);                  ! of the media?
; 1472     1568  2     END;                                                    ! Inform user of the
; 1473     1569  2                                                             ! invalid address generated.
; 1474     1570  2 RETURN .status;
; 1475     1571  2
; 1476     1572  1 END;     ! of ROUTINE val_phy_adr


                                                        .PSECT  $OWN$,NOEXE,2

                                                00008 HIBOUND:.BLKB   4
                                                0000C STATUS: .BLKB   4


                                                        .PSECT  $CODE$,NOWRT,2

                                                001C 00000 VAL_PHY_ADR:
                                                        .WORD   Save R2,R3,R4                           ; 1510
                                    54      0000V   CF  9E 00002        MOVAB   ADDR_ERROR, R4          :
                                    53      0000'   CF  9E 00007        MOVAB   STATUS, R3              :
                                    63              01  D0 0000C        MOVL    #1, STATUS              ; 1547
                            50      0000G   CF      01  C3 0000F        SUBL3   #1, BAD$GL_SECTORS, R0  ; 1549
                                    50        04    AC  D1 00015        CMPL    SEC, R0                 :
                                                    0D  15 00019        BLEQ    1$                      :
                                              04    AC  DD 0001B        PUSHL   SEC                     ; 1551
                                            0000'   CF  9F 0001E        PUSHAB  SECTOR                  :
                                    64              02  FB 00022        CALLS   #2, ADDR_ERROR          :
                                    63              50  D0 00025        MOVL    R0, STATUS              :
                            50      0000G   CF      01  C3 00028 1$:     SUBL3   #1, BAD$GL_TRACKS, R0   ; 1553
                                    50        08    AC  D1 0002E        CMPL    TRK, R0                 :
                                                    0D  15 00032        BLEQ    2$                      :
                                              08    AC  DD 00034        PUSHL   TRK                     ; 1555
                                            0000'   CF  9F 00037        PUSHAB  TRACK                   :
                                    64              02  FB 0003B        CALLS   #2, ADDR_ERROR          :
                                    63              50  D0 0003E        MOVL    R0, STATUS              :
                            50      0000G   CF      01  C3 00041 2$:     SUBL3   #1, BAD$GL_CYLINDERS, R0 ; 1557
                                    50        0C    AC  D1 00047        CMPL    CYL, R0                 :
                                                    0D  15 0004B        BLEQ    3$                      :
                                              0C    AC  DD 0004D        PUSHL   CYL                     ; 1559
                                            0000'   CF  9F 00050        PUSHAB  CYLINDER                :
                                    64              02  FB 00054        CALLS   #2, ADDR_ERROR          :
                                    63              50  D0 00057        MOVL    R0, STATUS              :
                                    52        10    AC  D0 0005A 3$:     MOVL    CNT, R2                 ; 1561
                                                    2E  13 0005E        BEQL    4$                      :
                                    7E        08    AC  7D 00060        MOVQ    TRK, -(SP)              ; 1564
                                              04    AC  DD 00064        PUSHL   SEC                     :
                        00000000G   00              03  FB 00067        CALLS   #3, BAD$CVT_PHY_LOG     :
                              FC    A3              50  D0 0006E        MOVL    R0, HIBOUND             :
                    51              52        FC    A3  C1 00072        ADDL3   HIBOUND, R2, R1         ; 1565
                    50      0000G   CF              01  C3 00077        SUBL3   #1, BAD$GL_MAXBLOCK, R0 :
                                    50              51  D1 0007D        CMPL    R1, R0                  :
                                                    0C  15 00080        BLEQ    4$                      :
                                                    52  DD 00082        PUSHL   R2                      ; 1567
                                            0000'   CF  9F 00084        PUSHAB  COUNT                   :
```

```
                                64          02  FB 00088          CALLS   #2, ADDR_ERROR                                ;
                                63          50  D0 0008B          MOVL    R0, STATUS                                    ;
                                50          63  D0 0008E 4$:      MOVL    STATUS, R0                                    ; 1570
                                                04 00091          RET                                                   ; 1572

; Routine Size:  146 bytes,    Routine Base:  $CODE$ + 0833

; 1477          1573  1
```

```
;1479        1574  1 %SBTTL 'val_log_adr -- Validate Logical Address'
;1480        1575  1 ROUTINE val_log_adr (lbn, cnt) =
;1481        1576  1 !++
;1482        1577  1 !
;1483        1578  1 ! Functional Description:
;1484        1579  1 !
;1485        1580  1 !    Validate Logical Block Number Address range.
;1486        1581  1 !
;1487        1582  1 ! Inputs:
;1488        1583  1 !
;1489        1584  1 !    lbn         val     the block number to validate.
;1490        1585  1 !    cnt         val     the number of contiguous block to validate.
;1491        1586  1 !
;1492        1587  1 ! Implicit Inputs:
;1493        1588  1 !
;1494        1589  1 !    bad$gl_maxblock    the total number of blocks on the device.
;1495        1590  1 !
;1496        1591  1 ! Side Effects:
;1497        1592  1 !
;1498        1593  1 !    If any of the logical block number or the count force us to generate an
;1499        1594  1 !    invalid address, we will issue a diagnostic and select on of the following
;1500        1595  1 !    actions.
;1501        1596  1 !
;1502        1597  1 !       If prompting, disregard the last entry and prompt for more input.
;1503        1598  1 !
;1504        1599  1 !       If processing blocks from the command line, the severity will be set
;1505        1600  1 !       to FATAL, resulting in termination of the analysis.
;1506        1601  1 !
;1507        1602  1 !--
;1508        1603  2 BEGIN
;1509        1604  2 LOCAL
;1510        1605  2     status;
;1511        1606  2
;1512        1607  2 status = TRUE;                                   ! Assume address is ok.
;1513        1608  2
;1514        1609  2 IF .lbn GTR .bad$gl_maxblock - 1                 ! Is this a valid lbn?
;1515        1610  2 THEN
;1516        1611  2     status = addr_error (logical_blk, .lbn);
;1517        1612  2
;1518        1613  2 IF .cnt NEQ 0                                    ! Should we consider the count?
;1519        1614  2 THEN                                            ! Yes.
;1520        1615  2     IF .lbn + .cnt GTR .bad$gl_maxblock - 1      ! Does the count push
;1521        1616  2     THEN                                        ! us off the end of the
;1522        1617  2         status = addr_error (count, .cnt);      ! media.
;1523        1618  2
;1524        1619  2 RETURN .status;
;1525        1620  2
;1526        1621  1 END;     ! of ROUTINE val_log_adr


                                000C 00000 VAL_LOG_ADR:
                                           .WORD   Save R2,R3                                  ;1575
                      50        01  D0 00002        MOVL    #1, STATUS                         ;1607
 51     0000G  CF     01  C3 00005        SUBL3   #1, BAD$GL_MAXBLOCK, R1                    ;1609
```

```
                         51      04     AC  D1 0000B          CMPL    LBN, R1                                          ;
                                        0C  15 0000F          BLEQ    1$                                               ;
                                 04     AC  DD 00011          PUSHL   LBN                                              ; 1611
                               0000'    CF  9F 00014          PUSHAB  LOGICAL_BLK                                      ;
                      0000V    CF             02  FB 00018    CALLS   #2, ADDR_ERROR                                   ;
                               52      08     AC  D0 0001D 1$: MOVL    CNT, R2                                          ; 1613
                                        1B  13 00021          BEQL    2$                                               ;
                   53          52      04     AC  C1 00023    ADDL3   LBN, R2, R3                                      ; 1615
                   51  0000G  CF             01  C3 00028    SUBL3   #1, BAD$GL_MAXBLOCK, R1                          ;
                               51             53  D1 0002E    CMPL    R3, R1                                           ;
                                        0B  15 00031          BLEQ    2$                                               ;
                                        52  DD 00033          PUSHL   R2                                               ; 1617
                               0000'    CF  9F 00035          PUSHAB  COUNT                                            ;
                      0000V    CF             02  FB 00039    CALLS   #2, ADDR_ERROR                                   ;
                                        04  0003E 2$:          RET                                                      ; 1621

; Routine Size:  63 bytes,    Routine Base:  $CODE$ + 08C5

; 1527          1622  1
```

```
;1529        1623  1 %SBTTL  'addr_error -- Address error handler'
;1530        1624  1 ROUTINE addr_error (desc, val) =
;1531        1625  1 !+
;1532        1626  1 !
;1533        1627  1 ! Functional Description:
;1534        1628  1 !
;1535        1629  1 !    There is an error in the address specification, inform the user of the
;1536        1630  1 !    problem area.
;1537        1631  1 !
;1538        1632  1 ! Side Effects:
;1539        1633  1 !
;1540        1634  1 !    If prompting, disregard the last entry and prompt for more input.
;1541        1635  1 !
;1542        1636  1 !    If processing blocks from the command line, the severity will be set
;1543        1637  1 !    to FATAL, resulting in termination of the analysis.
;1544        1638  1 !
;1545        1639  1 !--
;1546        1640  2 BEGIN
;1547        1641  2 bad$gb_term_count = 0;                          ! Reset the term count.
;1548        1642  2
;1549        1643  2 If bad$str_trim NEQ 0                          ! If we are stand alone, trim the string
;1550        1644  2 THEN    bad$str_trim (bad$ga_input_desc);       ! of trailing blanks, tabs and carriage return.
;1551        1645  2
;1552        1646  2 SIGNAL (IF .bad$gl_context [ctx_v_interactive] ! If the user is interactively entering
;1553        1647  2         THEN bad$_ivblknum                    ! bad blocks, reject this entry.
;1554        1648  2         ELSE bad$_ivblknum + STS$K_SEVERE,    ! Otherwise, stop the analysis here.
;1555        1649  2         3,                                    ! Three pieces of information...
;1556        1650  2         bad$ga_input_desc,                    ! The current input spec,
;1557        1651  2         .desc,                                ! the part of the bad block spec out of range,
;1558        1652  2         .val);                                ! and the offending value.
;1559        1653  2
;1560        1654  2 IF bad$str_trim NEQ 0                          ! Restore the buffer size if we are stand alone.
;1561        1655  2 THEN    bad$ga_input_desc [dsc$w_length] = bad$k_input_len;
;1562        1656  2
;1563        1657  2 RETURN FALSE;
;1564        1658  2
;1565        1659  1 END;    ! of ROUTINE addr_error


                                 000C 00000 ADDR_ERROR:
                                                .WORD   Save R2,R3                                      ;1624
                      53    0000G   CF   9E 00002     MOVAB   BAD$GA_INPUT_DESC, R3
                            0000G   CF   94 00007     CLRB    BAD$GB_TERM_COUNT                         ;1641
                      50    0000G   CF   9E 0000B     MOVAB   BAD$STR_TRIM, R0                          ;1643
                                    52   D4 00010     CLRL    R2
                                    50   D5 00012     TSTL    R0
                                    09   13 00014     BEQL    1$
                                    52   D6 00016     INCL    R2
                                    53   DD 00018     PUSHL   R3                                        ;1644
                 0000G  CF                 01   FB 0001A     CALLS   #1, BAD$STR_TRIM
                        7E       04       AC   7D 0001F 1$:  MOVQ    DESC, -(SP)                       ;1651
                                          53   DD 00023     PUSHL   R3                                ;1646
                                          03   DD 00025     PUSHL   #3
           08   0000G  CF                 05   E1 00027     BBC     #5, BAD$GL_CONTEXT, 2$
```

```
                                00000000G  8F  DD 0002D          PUSHL   #BAD$_IVBLKNUM                          ;
                                           06  11 00033          BRB     3$                                      ;
                                00000000G  8F  DD 00035 2$:      PUSHL   #BAD$_IVBLKNUM+4                        ; 1648
                     00000000G  00         05  FB 0003B 3$:      CALLS   #5, LIB$SIGNAL                          ; 1646
                                04         52  E9 00042          BLBC    R2, 4$                                  ; 1654
                                63     84  8F  9B 00045          MOVZBW  #132, BAD$GA_INPUT_DESC                 ; 1655
                                       50  D4 00049 4$:         CLRL    R0                                      ; 1657
                                           04 0004B              RET                                             ; 1659

; Routine Size:  76 bytes,    Routine Base:  $CODE$ + 0904

; 1566          1660  1
; 1567          1661  1 END     ! of MODULE badinit
; 1568          1662  0 ELUDOM



                                                                  .EXTRN  LIB$SIGNAL, LIB$STOP

;                                PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $PLIT$                        340  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  _LIB$KEY0$                      0  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _LIB$STATE$                   194  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  $CODE$                       2384  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                          16  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        29         0        1000        00:01.5
;  _$255$DUA28:[SYSLIB]TPAMAC.L32;1          42        27        64          14        00:00.1



;                                    COMMAND QUALIFIERS

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

; Size:         2384 code + 550 data bytes
; Run Time:       00:38.7
; Elapsed Time:   02:55.8
; Lines/CPU Min:   2579
; Lexemes/CPU-Min: 57142
```

```
; Memory Used:  326 pages
; Compilation Complete
```

BADBLOCKS
LIS

BAD

STABAD
MAP

BADDATA
LIS

ANALYZBAD
MAP

ANALYZFMD
CLD

BADDEF
SDL

WRITESAVE
LIS

BADINIT
LIS

ANALYZCMD
LIS

BADREPORT
LIS

BADBLK

SCANFILE
LIS

BADBLOCK
MAP

BADMAIN
LIS

BADIO
LIS

GETREQ
LIS

BADMSG
LIS

BADSTASYS
LIS