```
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      000000
BBBBBBBB      AAAAAA    DDDDDDDD      IIIIII      000000
BB      BB  AA      AA  DD      DD      II      00      00
BB      BB  AA      AA  DD      DD      II      00      00
BB      BB  AA      AA  DD      DD      II      00      00
BB      BB  AA      AA  DD      DD      II      00      00
BBBBBBBB    AA      AA  DD      DD      II      00      00
BBBBBBBB    AA      AA  DD      DD      II      00      00
BB      BB  AAAAAAAAAA  DD      DD      II      00      00
BB      BB  AAAAAAAAAA  DD      DD      II      00      00
BB      BB  AA      AA  DD      DD      II      00      00      ....
BB      BB  AA      AA  DD      DD      II      00      00      ....
BBBBBBBB    AA      AA  DDDDDDDD      IIIIII      000000      ....
BBBBBBBB    AA      AA  DDDDDDDD      IIIIII      000000      ....


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 badio (%TITLE 'Analyze/Media Input/Output Procedures'
;     2        0002  0                 IDENT = 'V04-000') =
;     3        0003  1 BEGIN
;     4        0004  1 !
;     5        0005  1 !**************************************************************************
;     6        0006  1 !*                                                                        *
;     7        0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8        0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
;     9        0009  1 !*  ALL RIGHTS RESERVED.                                                 *
;    10        0010  1 !*                                                                        *
;    11        0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
;    12        0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
;    13        0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
;    14        0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
;    15        0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
;    16        0016  1 !*  TRANSFERRED.                                                          *
;    17        0017  1 !*                                                                        *
;    18        0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
;    19        0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
;    20        0020  1 !*  CORPORATION.                                                          *
;    21        0021  1 !*                                                                        *
;    22        0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
;    23        0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.               *
;    24        0024  1 !*                                                                        *
;    25        0025  1 !*                                                                        *
;    26        0026  1 !**************************************************************************
;    27        0027  1 !
;    28        0028  1 !
;    29        0029  1 !++
;    30        0030  1 !
;    31        0031  1 ! Facility:
;    32        0032  1 !
;    33        0033  1 !       Analyze/Media
;    34        0034  1 !
;    35        0035  1 ! Abstract:
;    36        0036  1 !
;    37        0037  1 !       This module contains the procedures required to perform input
;    38        0038  1 !       and output operations in both synchronous and asynchronous modes.
;    39        0039  1 !
;    40        0040  1 ! Environment:
;    41        0041  1 !
;    42        0042  1 !       VAX/VMS User Mode, Non-Privileged
;    43        0043  1 !
;    44        0044  1 ! Author:
;    45        0045  1 !
;    46        0046  1 !       Michael T. Rhodes,              Creation Date: July, 1982
;    47        0047  1 !
;    48        0048  1 ! Modified By:
;    49        0049  1 !
;    50        0050  1 !       V03-002 MTR0004         Michael T. Rhodes       1-Mar-1983
;    51        0051  1 !               Relocate routines 'bad$retry' and 'bad$verify_blocks' to here
;    52        0052  1 !               and add routine 'recoverable_error' to perform robust IO error
;    53        0053  1 !               recovery actions.
;    54        0054  1 !
;    55        0055  1 !       V03-001 MTR0001         Michael T. Rhodes       15-Dec-1982
;    56        0056  1 !               Modify the AST servicing structure.
;    57        0057  1 !
```

```
;   58          0058  1 !--
```

```
;    60          0059  1 %SBTTL 'Declarations'
;    61          0060  1 !
;    62          0061  1 ! Include Files:
;    63          0062  1 !
;    64          0063  1 REQUIRE 'lib$:baddef';                          ! Define BAD's structures.
;    65          0183  1 LIBRARY 'SYS$LIBRARY:LIB';                      ! Define VMS structures.
;    66          0184  1
;    67          0185  1 !
;    68          0186  1 ! Table of Contents:
;    69          0187  1 !
;    70          0188  1 FORWARD ROUTINE
;    71          0189  1     bad$ast_serv        : NOVALUE,              ! AST service routine.
;    72          0190  1     bad$async_io,                               ! Perform assynchronous IO.
;    73          0191  1     bad$sync_io,                                ! Perform synchronous IO.
;    74          0192  1     bad$start_io        : NOVALUE,              ! Starts the asychrounous IO operations
;    75          0193  1                                                 ! and waits for completion.
;    76          0194  1     bad$retry           : NOVALUE,              ! Examine error status and retry recoverable IO erro
;    77          0195  1     bad$verify_blocks   : NOVALUE,              ! Verify the contents of each block written to the m
;    78          0196  1     recoverable_error;                          ! If the error condition is recoverable, perform rec
;    79          0197  1
;    80          0198  1 !
;    81          0199  1 ! Private Storage
;    82          0200  1 !
;    83          0201  1 OWN
;    84          0202  1     current_lbn,                                ! Current lbn to be transferred.
;    85          0203  1     devadr              : VECTOR [2, LONG],     ! Device   iresses for requests.
;    86          0204  1     last_lbn;                                   ! Lbn of last whole buffer transfer.
;    87          0205  1
;    88          0206  1 !
;    89          0207  1 ! External references:
;    90          0208  1 !
;    91          0209  1 EXTERNAL ROUTINE
;    92          0210  1     bad$check_ltk       : ADDRESSING_MODE (GENERAL),    ! Insert bad block into SDBSF if not in MDBSF or SDB
;    93          0211  1     bad$check_nlt       : ADDRESSING_MODE (GENERAL),    ! Insert bad block into SDBSF if not in non last tra
;    94          0212  1     LIB$MATCH_COND      : ADDRESSING_MODE (GENERAL);    ! Match the condition value specified.
;    95          0213  1
;    96          0214  1 EXTERNAL
;    97          0215  1     bad$gl_chan,                                ! Channel number for device.
;    98          0216  1     bad$gl_context      : BITVECTOR [32],       ! Global control context flags.
;    99          0217  1     bad$ga_bufadr       : VECTOR [2, LONG],     ! Addresses of the data buffers.\
;   100          0218  1     bad$ga_device       : $BBLOCK [dsc$c_s_bln],! Address of device name descriptor.
;   101          0219  1     bad$gl_devtype,                             ! Device type.
;   102          0220  1     bad$gl_func,                                ! IO function code for $QIO.
;   103          0221  1     bad$gq_iosb         : VECTOR [4, LONG],     ! IOSB Vector for both transfers.
;   104          0222  1     bad$gl_pagcnt,                              ! Number of pages transfered.
;   105          0223  1     bad$gl_status,                              ! Global status.
;   106          0224  1     bad$ga_tpb,                                 ! Address of the test pattern buffer.
;   107          0225  1     bad$gl_trnsfr_cnt;                          ! Number of bytes per transfer.
;   108          0226  1
;   109          0227  1 !
;   110          0228  1 ! Define message codes...
;   111          0229  1 !
;   112          0230  1 EXTERNAL LITERAL
;   113          0231  1     bad$_readerr,                               ! Read error.
;   114          0232  1     bad$_writeerr;                              ! Write error.
;   115          0233  1
```

```
;  117     0234  1 %SBTTL  'bad$ast_serv -- AST service routine'
;  118     0235  1 GLOBAL  ROUTINE bad$ast_serv (astprm, reg_0, reg_1, reg_pc, reg_psl) : NOVALUE =
;  119     0236  1 !++
;  120     0237  1 !
;  121     0238  1 ! Functional Description:
;  122     0239  1 !
;  123     0240  1 !    Upon IO completion, control is transfer here by VMS via AST delivery.
;  124     0241  1 !    We perform the following actions:
;  125     0242  1 !
;  126     0243  1 !       1.  Check the IO status returned in the IO status block selected
;  127     0244  1 !           by the AST parameter.
;  128     0245  1 !           If the status is unsuccessful, then retry the IO synchronously.
;  129     0246  1 !
;  130     0247  1 !       2.  Update the current block number.
;  131     0248  1 !
;  132     0249  1 !       3.  Check for remaining whole buffer transfers pending;
;  133     0250  1 !
;  134     0251  1 !           True:
;  135     0252  1 !               If the transfer was a read, initiate verification of
;  136     0253  1 !               the buffer just read, against the test pattern buffer.
;  137     0254  1 !               Set up the next transfer address.
;  138     0255  1 !               Issue the IO request.
;  139     0256  1 !
;  140     0257  1 !           False:
;  141     0258  1 !               Set the device completion event flag.
;  142     0259  1 !
;  143     0260  1 ! Inputs:
;  144     0261  1 !
;  145     0262  1 !    astprm     val     AST parameter value, used to indicate the IO request
;  146     0263  1 !                       parameters such as; the IO buffer, status block etc..
;  147     0264  1 !    reg_0      val     the contents of R0 at the time of AST delivery.
;  148     0265  1 !    reg_1      val     the contents of R1 at the time of AST delivery.
;  149     0266  1 !    reg_pc     val     the contents of the PC at the time of AST delivery.
;  150     0267  1 !    reg_psl    val     the contents of the PSL at the time of AST delivery.
;  151     0268  1 !
;  152     0269  1 !    NOTE:  ONLY the astprm is used, all other input parameters are unused.
;  153     0270  1 !
;  154     0271  1 !--
;  155     0272  2 BEGIN
;  156     0273  2 IF NOT .bad$gq_iosb [.astprm * 2]                          ! Was the transfer successful?
;  157     0274  2 THEN                                                       ! If not, retry the transfer synchronously.
;  158     0275  2     bad$retry (.devadr [.astprm], .bad$gl_trnsfr_cnt,
;  159     0276  2                bad$gq_iosb [.astprm * 2], .bad$ga_bufadr [.astprm]);
;  160     0277  2
;  161     0278  2 current_lbn = .current_lbn + .bad$gl_pagcnt;               ! Update the current LBN.
;  162     0279  2
;  163     0280  2 IF .current_lbn LEQ .last_lbn                              ! Have we finished all of the
;  164     0281  2 THEN                                                       ! whole buffer transfers?
;  165     0282  3     BEGIN                                                  ! NO...
;  166     0283  3     IF (.bad$gl_func AND %X'3F') EQL IO$_READLBLK          ! If read/verfiy phase, then request
;  167     0284  3     THEN                                                   ! buffer verification.
;  168     0285  3         bad$verify_blocks (.bad$ga_bufadr [.astprm],
;  169     0286  3                            .bad$gl_trnsfr_cnt,
;  170     0287  3                            .devadr [.astprm],
;  171     0288  3                             bad$gq_iosb [.astprm * 2]);
;  172     0289  3
;  173     0290  3     devadr [.astprm] = .current_lbn;                       ! Set up the new address.
```

```
;  174          0291  3      bad$async_io (.astprm);                         ! Queue the request.
;  175          0292  3      END
;  176          0293  2 ELSE
;  177          0294  2      $SETEF (EFN = 3);                               ! Yes, set the device completion semaphore.
;  178          0295  2
;  179          0296  2 RETURN
;  180          0297  2
;  181          0298  1 END;     ! of GLOBAL ROUTINE bad$ast_serv


                                                        .TITLE  BADIO Analyze/Media Input/Output Procedures
                                                        .IDENT  \V04-000\

                                                        .PSECT  $OWN$,NOEXE,2

                                                00000 CURRENT_LBN:
                                                        .BLKB   4
                                                00004 DEVADR:   .BLKB   8
                                                0000C LAST_LBN:
                                                        .BLKB   4

                                                        .EXTRN  BAD$CHECK_LTK, BAD$CHECK_NLT
                                                        .EXTRN  LIB$MATCH_COND, BAD$GL_CRAN
                                                        .EXTRN  BAD$GL_CONTEXT, BAD$GA_BUFADR
                                                        .EXTRN  BAD$GA_DEVICE, BAD$GL_DEVTYPE
                                                        .EXTRN  BAD$GL_FUNC, BAD$GQ_IOSB
                                                        .EXTRN  BAD$GL_PAGCNT, BAD$GL_STATUS
                                                        .EXTRN  BAD$GA_TPB, BAD$GL_TRNSFR_CNT
                                                        .EXTRN  BAD$_READERR, BAD$_WRITEERR
                                                        .EXTRN  SYS$SETEF

                                                        .PSECT  $CODE$,NOWRT,2

                                                    001C 00000      .ENTRY  BAD$AST_SERV, Save R2,R3,R4             ; 0235
                                            54      0000'  CF   9E  00002   MOVAB   DEVADR, R4
                                            52        04   AC   D0  00007   MOVL    ASTPRM, R2                      ; 0273
                                    50              52        01  78  0000B   ASHL    #1, R2, R0
                                            53      0000GCF40   DE  0000F   MOVAL   BAD$GQ_IOSB[R0], R3
                                            13              63  E8  00015   BLBS    (R3), 1$
                                                    0000GCF42   DD  00018   PUSHL   BAD$GA_BUFADR[R2]               ; 0276
                                                            53  DD  0001D   PUSHL   R3
                                                    0000G   CF  DD  0001F   PUSHL   BAD$GL_TRNSFR_CNT
                                                         6442   DD  00023   PUSHL   DEVADR[R2]
                                                0000V   CF          04  FB  00026   CALLS   #4, BAD$RETRY
                                                  FC    A4    0000G   CF  C0  0002B 1$: ADDL2   BAD$GL_PAGCNT, CURRENT_LBN  ; 0278
                                                  08    A4      FC    A4  D1  00031   CMPL    CURRENT_LBN, LAST_LBN   ; 0280
                                                                29  14  00036   BGTR    3$
    21          0000G   CF              06                    00  ED  00038   CMPZV   #0, #6, BAD$GL_FUNC, #33 ; 0283
                                                                13  12  0003F   BNEQ    2$
                                                            53  DD  00041   PUSHL   R3                              ; 0288
                                                         6442   DD  00043   PUSHL   DEVADR[R2]
                                                    0000G   CF  DD  00046   PUSHL   BAD$GL_TRNSFR_CNT
                                                    0000GCF42   DD  0004A   PUSHL   BAD$GA_BUFADR[R2]
                                                0000V   CF          04  FB  0004F   CALLS   #4, BAD$VERIFY_BLOCKS
                                                         6442       FC  A4  D0  00054 2$: MOVL    CURRENT_LBN, DEVADR[R2] ; 0290
                                                                52  DD  00059   PUSHL   R2                          ; 0291
                                                0000V   CF          01  FB  0005B   CALLS   #1, BAD$ASYNC_IO
```

```
                                 04 00060          RET                                          ; 0280
                              03 DD 00061 3$:      PUSHL   #3                                   ; 0294
                 00000000G  00 01 FB 00063         CALLS   #1, SYS$SETEF                        ;
                                 04 0006A          RET                                          ; 0298

; Routine Size:  107 bytes,    Routine Base:  $CODE$ + 0000

;  182          0299  1
```

```
;  184          0300  1 %SBTTL  'bad$start_io -- Start asynchronous IO transfers'
;  185          0301  1 GLOBAL  ROUTINE bad$start_io (last_block) : NOVALUE =
;  186          0302  1 !++
;  187          0303  1 !
;  188          0304  1 ! Functional Description:
;  189          0305  1 !
;  190          0306  1 !    This procedure is responsible for starting the asynchronous IO operations
;  191          0307  1 !    and waiting for the device completion semaphore to be set.  Once device
;  192          0308  1 !    completion is indicated, we return to our caller.
;  193          0309  1 !
;  194          0310  1 ! Inputs:
;  195          0311  1 !
;  196          0312  1 !    last_block val     The last block number to which a whole buffer
;  197          0313  1 !                       transfer may be made.
;  198          0314  1 !
;  199          0315  1 ! Side Effects:
;  200          0316  1 !
;  201          0317  1 !    event flag 3 is set indicating device comletion.
;  202          0318  1 !    asynchrounous io to the device is complete.
;  203          0319  1 !
;  204          0320  1 !--
;  205          0321  2 BEGIN
;  206          0322  2 $CLREF (EFN = 3);                                   ! Clear device completion semaphore.
;  207          0323  2 last_lbn = .last_block;                             ! Establish the last whole buffer transfer address.
;  208          0324  2 current_lbn = .bad$gl_pagcnt;                       ! Preset the current transfer block address.
;  209          0325  2 devadr [0] = 0;                                     ! Set the first device transfer address.
;  210          0326  2 devadr [1] = .current_lbn;                          ! Set the next transfer address.
;  211          0327  2 bad$async_io (0);                                   ! Initiate the IO operations.
;  212          0328  2 bad$async_io (1);                                   !
;  213          0329  2 $WAITFR (EFN = 3);                                  ! Wait for device completion semaphore.
;  214          0330  2
;  215          0331  1 END;     ! of GLOBAL ROUTINE bad$start_io

                                                        .EXTRN  SYS$CLREF, SYS$WAITFR

                                                0004 00000      .ENTRY  BAD$START_IO, Save R2                           ; 0301
                                52      0000'   CF   9E 00002  MOVAB   CURRENT_LBN, R2                                 ;
                                                03   DD 00007  PUSHL   #3                                               ; 0322
                00000000G       00              01   FB 00009  CALLS   #1, SYS$CLREF                                   ;
                        0C      A2      04      AC   D0 00010  MOVL    LAST_BLOCK, LAST_LBN                            ; 0323
                                62      0000G   CF   D0 00015  MOVL    BAD$GL_PAGCNT, CURRENT_LBN                      ; 0324
                                        04      A2   D4 0001A  CLRL    DEVADR                                          ; 0325
                        08      A2              62   D0 0001D  MOVL    CURRENT_LBN, DEVADR+4                           ; 0326
                                                7E   D4 00021  CLRL    -(SP)                                           ; 0327
                    0000V       CF              01   FB 00023  CALLS   #1, BAD$ASYNC_IO                                ;
                                                01   DD 00028  PUSHL   #1                                               ; 0328
                    0000V       CF              01   FB 0002A  CALLS   #1, BAD$ASYNC_IO                                ;
                                                03   DD 0002F  PUSHL   #3                                               ; 0329
                00000000G       00              01   FB 00031  CALLS   #1, SYS$WAITFR                                  ;
                                                04   00038  RET                                                        ; 0331

; Routine Size:  57 bytes,    Routine Base:  $CODE$ + 006B

;  216          0332  1
```

```
;  218          0333  1 %SBTTL  'bad$async_io -- Perform Asynchronous IO operation'
;  219          0334  1 GLOBAL  ROUTINE bad$async_io (selector) =
;  220          0335  1 !++
;  221          0336  1 !
;  222          0337  1 ! Functional Description:
;  223          0338  1 !
;  224          0339  1 !    This routine performs all asynchronous IO operations for the utility.
;  225          0340  1 !
;  226          0341  1 ! Inputs:
;  227          0342  1 !
;  228          0343  1 !    selector            Selects the IO buffer, IOSB, and device addresses.
;  229          0344  1 !                        It also doubles as the ASTPRM and we add 1 to form the EFN.
;  230          0345  1 !
;  231          0346  1 ! Implicit Inputs:
;  232          0347  1 !
;  233          0348  1 !    bad$gl_chan         IO channel for device
;  234          0349  1 !    bad$gl_func         IO function code
;  235          0350  1 !    bad$gq_iosb         IO status block Vector.
;  236          0351  1 !    bad$ga_bufadr       IO buffer address Vector.
;  237          0352  1 !    bad$gl_trnsfr_cnt   Number of bytes to be transfered.
;  238          0353  1 !    devadr              Device address vector (holds current io request addresses).
;  239          0354  1 !
;  240          0355  1 ! Implicit Outputs:
;  241          0356  1 !
;  242          0357  1 !    The data has been transfered to/from the data buffer.
;  243          0358  1 !
;  244          0359  1 ! Routine Value:
;  245          0360  1 !
;  246          0361  1 !    Success     If transfer request was queued successfully.
;  247          0362  1 !
;  248          0363  1 !    Failure     If transfer request failed to be queued.
;  249          0364  1 !
;  250          0365  1 ! Side Effects:
;  251          0366  1 !
;  252          0367  1 !    IO errors are signalled here, rather than by the caller alone.
;  253          0368  1 !
;  254          0369  1 !--
;  255          0370  2 BEGIN
;  256        P 0371  3 IF NOT (bad$gl_status = $QIO    (EFN = .selector + 1,
;  257        P 0372  3                                 CHAN =  .bad$gl_chan,
;  258        P 0373  3                                 FUNC =  IF .bad$gl_context [ctx_v_retry]
;  259        P 0374  3                                         THEN    .bad$gl_func
;  260        P 0375  3                                         ELSE    .bad$gl_func OR IO$M_INHRETRY,
;  261        P 0376  3                                 IOSB =  bad$gq_iosb [.selector * 2],
;  262        P 0377  3                                 ASTADR = bad$ast_serv,
;  263        P 0378  3                                 ASTPRM = .selector,
;  264        P 0379  3                                 P1 =    .bad$ga_bufadr [.selector],
;  265        P 0380  3                                 P2 =    .bad$gl_trnsfr_cnt,
;  266          0381  3                                 P3 =    .devadr [.selector]))
;  267          0382  2 THEN
;  268          0383  2     SIGNAL (IF (.bad$gl_func AND %X'3F') EQLU IO$_WRITELBLK
;  269          0384  2             THEN bad$_writeerr
;  270          0385  2             ELSE bad$_readerr,
;  271          0386  2             1, bad$ga_device,
;  272          0387  2             .bad$gl_status);
;  273          0388  2
;  274          0389  2 RETURN    .bad$gl_status;
```

```
;  275          0390  2
;  276          0391  1 END;    ! of GLOBAL ROUTINE bad$async_io


                                                                        .EXTRN  SYS$QIO

                                                                        .ENTRY  BAD$ASYNC_IO, Save R2,R3                        ; 0334
                                53      0000G   CF   9E 00002           MOVAB   BAD$GL_STATUS, R3                               ;
                                52      0000G   CF   9E 00007           MOVAB   BAD$GL_FUNC, R2                                 ;
                                                7E   7C 0000C           CLRQ    -(SP)                                           ; 0381
                                                7E   D4 0000E           CLRL    -(SP)                                           ;
                                51        04    AC   D0 00010           MOVL    SELECTOR, R1                                    ;
                                        0000'CF41   DD 00014           PUSHL   DEVADR[R1]                                      ;
                                        0000G   CF   DD 00019           PUSHL   BAD$GL_TRNSFR_CNT                               ;
                                        0000GCF41   DD 0001D           PUSHL   BAD$GA_BUFADR[R1]                               ;
                                                51   DD 00022           PUSHL   R1                                              ;
                                        FF34    CF   9F 00024           PUSHAB  BAD$AST_SERV                                    ;
                    50                  51          01   78 00028       ASHL    #1, R1, R0                                      ;
                                        0000GCF40   DF 0002C           PUSHAL  BAD$GQ_IOSB[R0]                                 ;
                    04      0000G   CF              03   E1 00031       BBC     #3, BAD$GL_CONTEXT+1, 1$                        ;
                                                    62   DD 00037       PUSHL   BAD$GL_FUNC                                     ;
                                                    0A   11 00039       BRB     2$                                              ;
                    50                  62 00008000 8F   C9 0003B 1$:   BISL3   #32768, BAD$GL_FUNC, R0                         ;
                                                    50   DD 00043       PUSHL   R0                                              ;
                                        0000G   CF   DD 00045 2$:       PUSHL   BAD$GL_CHAN                                     ;
                                          01    A1   9F 00049           PUSHAB  1(R1)                                           ;
                        00000000G       00          0C   FB 0004C       CALLS   #12, SYS$QIO                                    ;
                                        63          50   D0 00053       MOVL    R0, BAD$GL_STATUS                               ;
                                        24          50   E8 00056       BLBS    R0, 5$                                          ;
                                                    63   DD 00059       PUSHL   BAD$GL_STATUS                                   ; 0387
                                        0000G   CF   9F 0005B           PUSHAB  BAD$GA_DEVICE                                   ; 0383
                                                    01   DD 0005F       PUSHL   #1                                              ;
            20                  62              06       00   ED 00061   CMPZV   #0, #6, BAD$GL_FUNC, #32                       ;
                                                         08   12 00066   BNEQ    3$                                              ;
                                           00000000G     8F   DD 00068   PUSHL   #BAD$_WRITEERR                                  ;
                                                         06   11 0006E   BRB     4$                                              ;
                                           00000000G     8F   DD 00070 3$:   PUSHL   #BAD$_READERR                               ;
                        00000000G       00              04   FB 00076 4$:   CALLS   #4, LIB$SIGNAL                             ;
                                        50              63   D0 0007D 5$:   MOVL    BAD$GL_STATUS, R0                        ; 0389
                                                        04 00080           RET                                                     ; 0391

; Routine Size:  129 bytes,    Routine Base:  $CODE$ + 00A4

;  277          0392  1
```

```
;  279          0393  1 %SBTTL  'bad$sync_io -- Perform Synchronous IO operation'
;  280          0394  1 GLOBAL  ROUTINE bad$sync_io (bufadr, bytcnt, addr, iosbadr) =
;  281          0395  1 !++
;  282          0396  1 !
;  283          0397  1 ! Functional Description:
;  284          0398  1 !
;  285          0399  1 !    This routine performs all IO operations for the utility.
;  286          0400  1 !
;  287          0401  1 ! Inputs:
;  288          0402  1 !
;  289          0403  1 !    bufadr              Address of the data buffer
;  290          0404  1 !    bytcnt              Number of bytes to transfer
;  291          0405  1 !    addr                Device address for data transfer
;  292          0406  1 !    iosbadr             Address of the IO status block (see Side Effects)
;  293          0407  1 !
;  294          0408  1 ! Implicit Inputs:
;  295          0409  1 !
;  296          0410  1 !    bad$gl_chan         IO channel for device
;  297          0411  1 !    bad$gl_func         IO function code
;  298          0412  1 !
;  299          0413  1 ! Implicit Outputs:
;  300          0414  1 !
;  301          0415  1 !    The data has been transfered to/from the data buffer.
;  302          0416  1 !
;  303          0417  1 ! Routine Value:
;  304          0418  1 !
;  305          0419  1 !    The completion status and byte count of the IO (as extracted from
;  306          0420  1 !    the iosb).
;  307          0421  1 !
;  308          0422  1 ! Side Effects:
;  309          0423  1 !
;  310          0424  1 !    If the iosbadr parameter is omitted, we default to the use of
;  311          0425  1 !    bad$gq_iosb [0].
;  312          0426  1 !
;  313          0427  1 !    IO errors are signalled here, rather than by the caller alone.
;  314          0428  1 !
;  315          0429  1 !--
;  316          0430  2 BEGIN
;  317          0431  2
;  318          0432  2 BUILTIN
;  319          0433  2     NULLPARAMETER;
;  320          0434  2
;  321          0435  2 MAP
;  322          0436  2     iosbadr : REF VECTOR [2, LONG];
;  323          0437  2
;  324        P 0438  3 IF NOT (bad$gl_status = $QIOW   (CHAN = .bad$gl_chan,
;  325        P 0439  3                                  FUNC = IF .bad$gl_context [ctx_v_retry]
;  326        P 0440  3                                         THEN    .bad$gl_func
;  327        P 0441  3                                         ELSE    .bad$gl_func OR IO$M_INHRETRY,
;  328        P 0442  3                                  IOSB =  IF NULLPARAMETER (4)
;  329        P 0443  3                                         THEN    bad$gq_iosb [0]
;  330        P 0444  3                                         ELSE    iosbadr [0],
;  331        P 0445  3                                  P1 =    .bufadr,
;  332        P 0446  3                                  P2 =    .bytcnt,
;  333          0447  3                                  P3 =    .addr))
;  334          0448  2 THEN
;  335          0449  2     SIGNAL (IF (.bad$gl_func AND %X'3F') EQLU IO$_WRITELBLK
```

```
;  336     0450  2              THEN bad$_writeerr
;  337     0451  2              ELSE bad$_readerr,
;  338     0452  2              1, bad$ga_device,
;  339     0453  2              .bad$gl_status);
;  340     0454  2
;  341     0455  2 RETURN  IF NULLPARAMETER (4)
;  342     0456  2         THEN    .bad$gq_iosb [0]
;  343     0457  2         ELSE    .iosbadr [0];
;  344     0458  2
;  345     0459  1 END;    ! of GLOBAL ROUTINE bad$sync_io
```

```
                                                        .EXTRN  SYS$QIOW

                                        0004 00000      .ENTRY  BAD$SYNC_IO, Save R2                    ; 0394
                        52      0000G   CF   9E 00002  MOVAB   BAD$GL_FUNC, R2
                                        7E   7C 00007  CLRQ    -(SP)                                   ; 0447
                                        7E   D4 00009  CLRL    -(SP)
                        7E        08    AC   7D 0000B  MOVQ    BYTCNT, -(SP)
                                  04    AC   DD 0000F  PUSHL   BUFADR
                                        7E   7C 00012  CLRQ    -(SP)
                        04              6C   91 00014  CMPB    (AP), #4
                                        05   1F 00017  BLSSU   1$
                                  10    AC   D5 00019  TSTL    16(AP)
                                        09   12 0001C  BNEQ    2$
                        50      0000G   CF   9E 0001E  1$:     MOVAB   BAD$GQ_IOSB, R0
                                        50   DD 00023  PUSHL   R0
                                        03   11 00025  BRB     3$
                                  10    AC   DD 00027  2$:     PUSHL   IOSBADR
                  04      0000G   CF         03   E1 0002A  3$:     BBC     #3, BAD$GL_CONTEXT+1, 4$
                                              62   DD 00030  PUSHL   BAD$GL_FUNC
                                              0A   11 00032  BRB     5$
                  50                    62 00008000   8F   C9 00034  4$:     BISL3   #32768, BAD$GL_FUNC, R0
                                              50   DD 0003C  PUSHL   R0
                                    0000G   CF   DD 0003E  5$:     PUSHL   BAD$GL_CHAN
                                              7E   D4 00042  CLRL    -(SP)
                      00000000G     00         0C   FB 00044  CALLS   #12, SYS$QIOW
                          0000G     CF         50   D0 0004B  MOVL    R0, BAD$GL_STATUS
                                    26         50   E8 00050  BLBS    R0, 8$
                                    0000G      CF   DD 00053  PUSHL   BAD$GL_STATUS                   ; 0423
                                    0000G      CF   9F 00057  PUSHAB  BAD$GA_DEVICE                   ; 0424
                                               01   DD 0005B  PUSHL   #1
          20            62                    06   ED 0005D  CMPZV   #0, #6, BAD$GL_FUNC, #32
                                               08   12 00062  BNEQ    6$
                                 00000000G     8F   DD 00064  PUSHL   #BAD$_WRITEERR
                                               06   11 0006A  BRB     7$
                                 00000000G     8F   DD 0006C  6$:     PUSHL   #BAD$_READERR
                      00000000G     00         04   FB 00072  7$:     CALLS   #4, LIB$SIGNAL
                                    04         6C   91 00079  8$:     CMPB    (AP), #4                ; 0455
                                               05   1F 0007C  BLSSU   9$
                                         10    AC   D5 0007E  TSTL    16(AP)
                                               06   12 00081  BNEQ    10$
                                    50      0000G  CF   D0 00083  9$:     MOVL    BAD$GQ_IOSB, R0         ; 0456
                                                         04 00088  RET
                                    50        10   BC   D0 00089  10$:    MOVL    @IOSBADR, R0            ; 0457
                                                         04 0008D  RET                                   ; 0459
```

```
; Routine Size:  142 bytes,    Routine Base:  $CODE$ + 0125

;  346          0460  1
```

```
;  348     0461  1 %SBTTL  'bad$retry -- Retry bad block'
;  349     0462  1 GLOBAL ROUTINE  bad$retry (lbn, cnt, iosb, bufadr) : NOVALUE =
;  350     0463  1 !++
;  351     0464  1 !
;  352     0465  1 ! Functional Description:
;  353     0466  1 !
;  354     0467  1 !    An error has been detected during the IO transfer.
;  355     0468  1 !    We will examine the IOSB to determine the number of blocks
;  356     0469  1 !    successfully transfered.  With this we can 'skip' the bad
;  357     0470  1 !    block and continue to write/read the remainder of the blocks
;  358     0471  1 !    (recursively if need be!).
;  359     0472  1 !
;  360     0473  1 ! Inputs:
;  361     0474  1 !
;  362     0475  1 !    lbn         val     starting logical block number for the transfers.
;  363     0476  1 !    cnt         val     number of bytes requested by the IO.
;  364     0477  1 !    iosb        adr     address of the IO status block to use.
;  365     0478  1 !    bufadr      adr     address of the data buffer to use.
;  366     0479  1 !
;  367     0480  1 ! Side Effects:
;  368     0481  1 !
;  369     0482  1 !    The block which caused the IO operation to unsuccessfully complete
;  370     0483  1 !    will be retired to the SDBSF (if its not already!).
;  371     0484  1 !
;  372     0485  1 !--
;  373     0486  2 BEGIN
;  374     0487  2 MAP
;  375     0488  2     iosb : REF VECTOR [4, WORD];                    ! Allow reference to status/byte count etc..
;  376     0489  2
;  377     0490  2 LOCAL
;  378     0491  2     bad_block,                                      ! Logical Block Number of the bad block.
;  379     0492  2     blocks_done,                                    ! Number of blocks successfully transferred.
;  380     0493  2     byte_count,                                     ! Number of bytes remaining to be transferred.
;  381     0494  2     next_bufadr;                                    ! The address within the buffer to use during IO.
;  382     0495  2
;  383     0496  2 IF NOT recoverable_error (.iosb [0])                ! Can we recover from this error condition?
;  384     0497  2 THEN SIGNAL_STOP (                                  ! No, inform the user of the problem.
;  385     0498  2         IF (.bad$gl_func AND %X'3F') EQLU IO$_WRITELBLK ! Choose the message (read or write error)
;  386     0499  2         THEN bad$_writeerr
;  387     0500  2         ELSE bad$_readerr,
;  388     0501  2         1, bad$ga_device,                           ! Specify the device in question...
;  389     0502  2         .iosb [0]);                                 ! and the specific problem.
;  390     0503  2
;  391     0504  2 blocks_done = .iosb [1] / bad$k_page_size;          ! Number of blocks successfully transferred.
;  392     0505  2 bad_block = .lbn + .blocks_done;                    ! LBN of the bad block.
;  393     0506  2 byte_count = .cnt - bad$k_page_size -               ! Bytes remaining to be transferred.
;  394     0507  2             (.blocks_done * bad$k_page_size);       ! (round the number of bytes off to page boundary)
;  395     0508  2 next_bufadr = .bufadr + (.cnt - .byte_count);       ! Beginning address within buffer to use during IO.
;  396     0509  2
;  397     0510  2 IF .bad$gl_context [ctx_v_ltdevice]                 ! Check for this block in the
;  398     0511  2 THEN                                                ! appropriate file format.  If it
;  399     0512  2     bad$check_ltk (.bad_block, .bad_block)          ! has not already been recorded,
;  400     0513  2 ELSE                                                ! create a new entry in the SDBSF
;  401     0514  2     bad$check_nlt (.bad_block, .bad_block);         ! for this block.
;  402     0515  2
;  403     0516  2 IF .iosb [1] LSS .cnt                               ! Has the IO actually been completed?
;  404     0517  2 AND .byte_count GTR 0                               ! Do we actually have anything to transfer?
```

```
;  405     0518  2 THEN                                                  ! If so, skip past the bad block and finish up.
;  406     0519  3     BEGIN                                             !
;  407     0520  3     iosb [1] = 0;                                     ! Reset the byte count.
;  408     0521  3     bad$sync_io (.next_bufadr,                        ! Use the correct buffer address to complete the IO.
;  409     0522  3                  .byte_count,                         ! Transfer the remaining bytes.
;  410     0523  3                  .bad_block + 1,                      ! Skip past the bad block to continue.
;  411     0524  3                  iosb [0]);                           ! Reuse the old IO status block.
;  412     0525  3
;  413     0526  3     IF NOT .iosb [0]                                  ! ERROR during IO?!
;  414     0527  3     THEN                                              ! Recursively go past bad block(s).
;  415     0528  3         bad$retry (.bad_block + 1,                    ! Starting LBN.
;  416     0529  3                    .byte_count,                       ! Number of bytes we wanted to transfer.
;  417     0530  3                    iosb [0],                          ! Address of the IO status block.
;  418     0531  3                    .next_bufadr);                     ! Beginning buffer address.
;  419     0532  2     END;
;  420     0533  2
;  421     0534  1 END;    ! of GLOBAL ROUTINE bad$retry


                                                  003C 00000          .ENTRY  BAD$RETRY, Save R2,R3,R4,R5                     ; 0462
                                        53      0C    AC  D0 00002    MOVL    IOSB, R3                                        ; 0496
                                        7E            63  3C 00006    MOVZWL  (R3), -(SP)                                     ;
                                  0000V CF            01  FB 00009    CALLS   #1, RECOVERABLE_ERROR                           ;
                                        27            50  E8 0000E    BLBS    R0, 3$                                          ;
                                        7E            63  3C 00011    MOVZWL  (R3), -(SP)                                     ; 0502
                                             0000G    CF  9F 00014    PUSHAB  BAD$GA_DEVICE                                   ; 0497
                                                      01  DD 00018    PUSHL   #1                                              ;
              20      0000G  CF           06          00  ED 0001A    CMPZV   #0, #6, BAD$GL_FUNC, #32                        ; 0498
                                                      08  12 00021    BNEQ    1$                                              ;
                                         00000000G    8F  DD 00023    PUSHL   #BAD$_WRITEERR                                  ;
                                                      06  11 00029    BRB     2$                                              ;
                                         00000000G    8F  DD 0002B 1$: PUSHL   #BAD$_READERR                                  ;
                                   00000000G  00      04  FB 00031 2$: CALLS   #4, LIB$STOP                                  ;
                                        50      02    A3  3C 00038 3$: MOVZWL  2(R3), BLOCKS_DONE                             ; 0504
                                        50  00000200  8F  C6 0003C    DIVL2   #512, BLOCKS_DONE                               ;
                                  54                50      04    AC  C1 00043    ADDL3   LBN, BLOCKS_DONE, BAD_BLOCK             ; 0505
                                  50                50            09  78 00048    ASHL    #9, R0, R0                              ; 0507
                                  52          08    AC            50  C3 0004C    SUBL3   R0, CNT, R2                             ;
                                              52    FE00          C2  9E 00051    MOVAB   -512(R2), BYTE_COUNT                    ; 0506
                                  50          08    AC            52  C3 00056    SUBL3   BYTE_COUNT, CNT, R0                     ; 0508
                                  55                50      10    AC  C1 0005B    ADDL3   BUFADR, R0, NEXT_BUFADR                 ;
                                                    0D    0000G   CF  E9 00060    BLBC    BAD$GL_CONTEXT+1, 4$                    ; 0510
                                                                  54  DD 00065    PUSHL   BAD_BLOCK                               ; 0512
                                                                  54  DD 00067    PUSHL   BAD_BLOCK                               ;
                                   00000000G  00                  02  FB 00069    CALLS   #2, BAD$CHECK_LTK                   ;
                                                                  0B  11 00070    BRB     5$                                      ;
                                                                  54  DD 00072 4$: PUSHL   BAD_BLOCK                           ; 0514
                                                                  54  DD 00074    PUSHL   BAD_BLOCK                           ;
                                   00000000G  00                  02  FB 00076    CALLS   #2, BAD$CHECK_NLT                   ;
        08    AC        02    A3              10                  00  ED 0007D 5$: CMPZV   #0, #16, 2(R3), CNT                ; 0516
                                                                  22  18 00084    BGEQ    6$                                      ;
                                                                  52  D5 00086    TSTL    BYTE_COUNT                              ; 0517
                                                                  1E  15 00088    BLEQ    6$                                      ;
                                                        02        A3  B4 0008A    CLRW    2(R3)                                   ; 0520
```

```
                                        53  DD 0008D          PUSHL   R3                                          ; 0524
                                    01  A4  9F 0008F          PUSHAB  1(BAD_BLOCK)                                ; 0523
                                        52  DD 00092          PUSHL   BYTE_COUNT                                  ; 0524
                                        55  DD 00094          PUSHL   NEXT_BUFADR                                 ;
                                FED7    CF  04  FB 00096      CALLS   #4, BAD$SYNC_IO                             ;
                                        0A  63  E8 0009B      BLBS    (R3), 6$                                    ; 0526
                                        2C  BB 0009E          PUSHR   #^M<R2,R3,R5>                               ; 0530
                                    01  A4  9F 000A0          PUSHAB  1(BAD_BLOCK)                                ; 0528
                                FF58    CF  04  FB 000A3      CALLS   #4, BAD$RETRY                               ; 0530
                                            04  000A8 6$:      RET                                                 ; 0534

; Routine Size:  169 bytes,    Routine Base:  $CODE$ + 01B3

;  422          0535  1
```

```
;  424          0536  1 %SBTTL  'bad$verify_blocks -- Verify contents of blocks against test pattern'
;  425          0537  1 GLOBAL ROUTINE  bad$verify_blocks (bufadr, bufsiz, devadr, iosb) : NOVALUE =
;  426          0538  1 !++
;  427          0539  1 !
;  428          0540  1 ! Functional Description:
;  429          0541  1 !
;  430          0542  1 !    Compare the contents of the data buffer block by block against the test
;  431          0543  1 !    test pattern buffer.  If a mismatch occurs, we will look up the block
;  432          0544  1 !    in the appropriate bad block file.  If it has not been recorded previously
;  433          0545  1 !    we will create a new entry and retire it to the bad block file (SDBSF).
;  434          0546  1 !
;  435          0547  1 ! Inputs:
;  436          0548  1 !
;  437          0549  1 !    bufadr      adr     address of the data buffer.
;  438          0550  1 !    bufsiz      val     size in bytes of the data buffer.
;  439          0551  1 !    devadr      val     starting logical block number for the transfer.
;  440          0552  1 !    iosb        adr     address of the IO status block.
;  441          0553  1 !
;  442          0554  1 ! Implicit Inputs:
;  443          0555  1 !
;  444          0556  1 !    bad$ga_tpb          address of the current test pattern buffer.
;  445          0557  1 !
;  446          0558  1 !--
;  447          0559  2 BEGIN
;  448          0560  2 LOCAL
;  449          0561  2     bad_block;                                          ! Bad block number (LBN).
;  450          0562  2
;  451          0563  2 INCR block FROM .bufadr TO .bufadr + .bufsiz - 4 BY bad$k_page_size DO
;  452          0564  2     IF NOT CH$EQL (bad$k_page_size, .block,
;  453          0565  2                    bad$k_page_size, .bad$ga_tpb)
;  454          0566  2     THEN
;  455          0567  3         BEGIN                                           ! The current block has failed.
;  456          0568  3         bad_block = .devadr +                           ! Compute the LBN of the failure.
;  457          0569  3             ((.block - .bufadr) / bad$k_page_size);
;  458          0570  3
;  459          0571  3         IF .bad$gl_context [ctx_v_ltdevice]             ! Check for this block in the
;  460          0572  3         THEN                                            ! appropriate file format.  If it
;  461          0573  3             bad$check_ltk (.bad_block, .bad_block)      ! has not already been recorded,
;  462          0574  3         ELSE                                            ! create a new entry in   ? SDBSF
;  463          0575  3             bad$check_nlt (.bad_block, .bad_block);     ! for this block.
;  464          0576  2         END;
;  465          0577  2
;  466          0578  1 END;    ! of GLOBAL ROUTINE bad$verify_blocks
```

```
                                                 007C 00000          .ENTRY  BAD$VERIFY_BLOCKS, Save R2,R3,R4,R5,R6      ; 0537
        50      04      AC          08       AC   C1 00002          ADDL3   BUFSIZ, BUFADR, R0                          ; 0563
                56      FC      A0  9E 00008                       MOVAB   -4(R0), R6                                  ;
                54      04      AC  D0 0000C                       MOVL    BUFADR, BLOCK                               ;
                                    3D 11 00010                    BRB     4$                                          ;
0000G   DF              64    0200  8F 29 00012 1$:                CMPC3   #512, (BLOCK), @BAD$GA_TPB                  ; 0564
                                    2E 13 0001A                    BEQL    3$                                          ;
        50      54      04      AC  C3 0001C                       SUBL3   BUFADR, BLOCK, R0                           ; 0569
                50 00000200     8F  C6 00021                       DIVL2   #512, R0                                    ;
```

```
                55              50      0C     AC  C1 00028          ADDL3   DEVADR, R0, BAD_BLOCK                   ;
                                0D      0000G  CF  E9 0002D          BLBC    BAD$GL_CONTEXT+T, 2$                    ; 0571
                                               55  DD 00032          PUSHL   BAD_BLOCK                               ; 0573
                                               55  DD 00034          PUSHL   BAD_BLOCK                               ;
                    00000000G   00             02  FB 00036          CALLS   #2, BAD$CHECK_LTK                       ;
                                               0B  11 0003D          BRB     3$                                      ;
                                               55  DD 0003F 2$:      PUSHL   BAD_BLOCK                               ; 0575
                                               55  DD 00041          PUSHL   BAD_BLOCK                               ;
                    00000000G   00             02  FB 00043          CALLS   #2, BAD$CHECK_NLT                       ;
                                54      0200   C4  9E 0004A 3$:      MOVAB   512(R4), BLOCK                          ; 0564
                                56             54  D1 0004F 4$:      CMPL    BLOCK, R6                               ;
                                               BE  15 00052          BLEQ    1$                                      ;
                                                   04 00054          RET                                             ; 0578

; Routine Size:  85 bytes,    Routine Base:  $CODE$ + 025C

;  467          0579  1
```

```
;  469          0580  1 %SBTTL  'recoverable_error -- Perform recovery action if possible'
;  470          0581  1 ROUTINE recoverable_error (status) =
;  471          0582  1 !++
;  472          0583  1 !
;  473          0584  1 ! Functional Description:
;  474          0585  1 !
;  475          0586  1 !     This procedure is responsible for determining the severity of an IO error.
;  476          0587  1 !     This 'severity' is returned to the caller as the routine value and is
;  477          0588  1 !     categorized into two classes.  RECOVERABLE [true] which indicates that a
;  478          0589  1 !     recovery action can be performed and NONRECOVERABLE [false] which indicates
;  479          0590  1 !     that no recovery action can be performed.
;  480          0591  1 !
;  481          0592  1 !     The selection for the severity/action is accomplished by matching the
;  482          0593  1 !     status supplied by the caller against a list of 'known recoverable
;  483          0594  1 !     error conditions'.  A match generally indicates that the condition is
;  484          0595  1 !     recoverable and all others are considered nonrecoverable.  An index is
;  485          0596  1 !     generated which becomes a dispatch argument to CASE to the appropriate
;  486          0597  1 !     action routine.
;  487          0598  1 !
;  488          0599  1 ! Inputs:
;  489          0600  1 !
;  490          0601  1 !     status     val     the actual IO error condition returned from the $QIO.
;  491          0602  1 !
;  492          0603  1 ! Routine Value:
;  493          0604  1 !
;  494          0605  1 !     true               the error was recoverable.
;  495          0606  1 !     false              the error was not recoverable.
;  496          0607  1 !
;  497          0608  1 !--
;  498          0609  2 BEGIN
;  499          0610  2
;  500          0611  2 LITERAL
;  501          0612  2     opincompl = 1,                                  ! First error condition label.
;  502          0613  2     parity =    2;                                  ! Last error condition label.
;  503          0614  2
;  504          0615  2 LOCAL
;  505          0616  2     index,                                          ! Index to action routine.
;  506          0617  2     recoverable : INITIAL (FALSE);                  ! Condition severity returned to the caller.
;  507          0618  2
;  508          0619  2 index = LIB$MATCH_COND (status,                     ! Find the action routine which matchs this conditio
;  509          0620  2                         %REF (ss$_opincompl),       ! Operation incomplete.
;  510          0621  2                         %REF (ss$_parity));         ! Parity error.
;  511          0622  2
;  512          0623  2 CASE .index FROM opincompl TO parity OF
;  513          0624  2 SET
;  514          0625  2 [opincompl]:    IF .bad$gl_devtype EQL dt$_rp07     ! Operation Incomplete.  Special Case, for the RP07
;  515          0626  2                 THEN recoverable = true;            ! this condition generally indicates a media flaw.
;  516          0627  2                                                     ! However, for all others it indicates a hardware er
;  517          0628  2
;  518          0629  2 [parity]:       recoverable = true;                 ! Parity error.  Mark the block bad and continue.
;  519          0630  2 TES;
;  520          0631  2
;  521          0632  2 RETURN .recoverable;                                ! Return the severity of this condition.
;  522          0633  2
;  523          0634  1 END;     ! of ROUTINE recoverable_error
```

```
                                                0004 00000 RECOVERABLE_ERROR:
                                                           .WORD   Save R2                                          ; 0581
                                 5E                 08  C2 00002          SUBL2   #8, SP                          ;
                                                    52  D4 00005          CLRL    RECOVERABLE                     ; 0609
                           04    AE          01F4    8F  3C 00007          MOVZWL  #500, 4(SP)                     ; 0621
                                 04    AE       04   9F  0000D          PUSHAB  4(SP)                           ;
                           04    AE          02D4    8F  3C 00010          MOVZWL  #724, 4(SP)                     ; 0620
                                 04    AE       04   9F  00016          PUSHAB  4(SP)                           ;
                                 04    AC       04   9F  00019          PUSHAB  STATUS                          ; 0619
                00000000G        00                 03  FB 0001C          CALLS   #3, LIB$MATCH_COND              ;
                01               01                 50  CF 00023          CASEL   INDEX, #1, #1                   ; 0623
                                      000B                0004    00027 1$:     .WORD   2$-1$,-                  ;
                                                                                 3$-1$                          ;
                                 07          0000G  CF  D1 0002B 2$:     CMPL    BAD$GL_DEVTYPE, #7              ; 0625
                                                    03  12 00030          BNEQ    4$                              ;
                                 52                 01  D0 00032 3$:     MOVL    #1, RECOVERABLE                 ; 0629
                                 50                 52  D0 00035 4$:     MOVL    RECOVERABLE, R0                 ; 0632
                                                        04 00038          RET                                     ; 0634

; Routine Size:  57 bytes,    Routine Base:  $CODE$ + 02B1

;  524          0635  1
;  525          0636  1 END     ! of MODULE badio
;  526          0637  0 ELUDOM
```

```
                                                           .EXTRN  LIB$SIGNAL, LIB$STOP

;                                PSECT SUMMARY

;     Name                      Bytes                           Attributes

; $OWN$                          16  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
; $CODE$                        746  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                              Library Statistics

;                                        -------- Symbols --------      Pages       Processing
;     File                               Total    Loaded   Percent      Mapped      Time

; _$255$DUA28:[SYSLIB]LIB.L32;1           18619       15         0      1000          00:01.5
```

```
;                                       COMMAND QUALIFIERS

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

; Size:          746 code + 16 data bytes
; Run Time:          00:10.4
; Elapsed Time:      00:49.4
; Lines/CPU Min:      3664
; Lexemes/CPU-Min:  21394
; Memory Used:  103 pages
; Compilation Complete
```

BADREPORT
LIS

BADBLK

SCANFILE
LIS

BADBLOCK
MAP

BADMAIN
LIS

BADIO
LIS

GETREQ
LIS

BADMSG
LIS

BADSTASYS
LIS