```
BBBBBBBBBBBB        AAAAAAAAA       SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA       SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA       SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBBBBBBBBBBB       AAA         AAA     SSSSSSSSS    RRRRRRRRRRR       TTT        LLL
BBBBBBBBBBBB       AAA         AAA     SSSSSSSSS    RRRRRRRRRRR       TTT        LLL
BBBBBBBBBBBB       AAA         AAA     SSSSSSSSS    RRRRRRRRRRR       TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR         TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR         TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR         TTT        LLL
BBB          BBB   AAA         AAA           SSS  RRR      RRR      TTT        LLL
BBB          BBB   AAA         AAA           SSS  RRR      RRR      TTT        LLL
BBB          BBB   AAA         AAA           SSS  RRR      RRR      TTT        LLL
BBBBBBBBBBBB       AAA         AAA  SSSSSSSSSSS  RRR         RRR    TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB       AAA         AAA  SSSSSSSSSSS  RRR         RRR    TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB       AAA         AAA  SSSSSSSSSSS  RRR         RRR    TTT        LLLLLLLLLLLLLLL
```

```
BBBBBBBB      AAAAAA      SSSSSSSS  EEEEEEEEEE RRRRRRRR   RRRRRRRR     000000    RRRRRRRR
BBBBBBBB      AAAAAA      SSSSSSSS  EEEEEEEEEE RRRRRRRR   RRRRRRRR     000000    RRRRRRRR
BB      BB   AA      AA  SS        EE         RR      RR  RR      RR  00      00  RR      RR
BB      BB   AA      AA  SS        EE         RR      RR  RR      RR  00      00  RR      RR
BB      BB   AA      AA  SS        EE         RR      RR  RR      RR  00      00  RR      RR
BB      BB   AA      AA  SS        EE         RR      RR  RR      RR  00      00  RR      RR
BBBBBBBB     AA      AA    SSSSSS  EEEEEEEE   RRRRRRRR    RRRRRRRR   00      00  RRRRRRRR
BBBBBBBB     AA      AA    SSSSSS  EEEEEEEE   RRRRRRRR    RRRRRRRR   00      00  RRRRRRRR
BB      BB   AAAAAAAAAA          SS EE         RR  RR      RR  RR     00      00  RR  RR
BB      BB   AAAAAAAAAA          SS EE         RR  RR      RR  RR     00      00  RR  RR
BB      BB   AA      AA          SS EE         RR    RR    RR    RR   00      00  RR    RR      ....
BB      BB   AA      AA          SS EE         RR    RR    RR    RR   00      00  RR    RR      ....
BBBBBBBB     AA      AA  SSSSSSSS   EEEEEEEEEE RR      RR  RR      RR   000000   RR      RR    ....
BBBBBBBB     AA      AA  SSSSSSSS   EEEEEEEEEE RR      RR  RR      RR   000000   RR      RR    ....

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 BAS$ERROR (                          ! Error processing
;     2          0002  0                 IDENT = '1-074'            ! File: BASERROR.B32 Edit: MDL1074
;     3          0003  0                 ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                  *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
;    28          0028  1 !
;    29          0029  1 !
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 ! FACILITY:  VAX-11 BASIC Error Handling
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !
;    36          0036  1 !       This module contains the VAX-11 BASIC error handling logic.
;    37          0037  1 !       The error data base is OWN to this module.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:  VAX-11 user mode
;    40          0040  1 !
;    41          0041  1 ! AUTHOR: John Sauter, CREATION DATE: 17-Oct-78
;    42          0042  1 !
;    43          0043  1 ! MODIFIED BY:
;    44          0044  1 !
;    45          0045  1 ! 1-001 - Original.  JBS 27-NOV-78
;    46          0046  1 ! 1-002 - Remove BAS$$SIGNAL_IO and BAS$$STOP_IO.  They now live in
;    47          0047  1 !         their own module.  JBS 08-DEC-78
;    48          0048  1 ! 1-003 - Add global definitions of BAS$_abcmnoxyz symbols.  JBS 11-DEC-78
;    49          0049  1 ! 1-004 - Include severity in those definitions.  JBS 19-DEC-78
;    50          0050  1 ! 1-005 - If the compiled code does not do any error processing,
;    51          0051  1 !         either continue, restart the line or exit.  JBS 28-DEC-78
;    52          0052  1 ! 1-006 - Call BAS$$CB_CLEANUP to flush active I/O when unwinding.
;    53          0053  1 !         JBS 29-DEC-78
;    54          0054  1 ! 1-007 - Change BAS$$CB_CLEANUP to OTS$CLEANUP_IO.  JBS 09-JAN-1979
;    55          0055  1 ! 1-008 - When restarting an I/O statement, do an UNWIND to the
;    56          0056  1 !         beginning of the I/O statement.  JBS 26-JAN-1979
;    57          0057  1 ! 1-009 - Remove OTS$CLEANUP_IO, since we will do I/O cleanup using
```

```
;    58          0058  1 !                 a stack frame.  JBS 26-JAN-1979
;    59          0059  1 ! 1-010 - When searching for a line number corresponding to a PC,
;    60          0060  1 !                 look in the right place in the table.  JBS 30-JAN-1979
;    61          0061  1 ! 1-011 - When getting storage for the SIGNAL argument list, get enough
;    62          0062  1 !                 for the argument count and the two trailing longwords, even
;    63          0063  1 !                 though this may sometimes be a little more than is needed.
;    64          0064  1 !                 JBS 31-JAN-1979
;    65          0065  1 ! 1-012 - Purge the terminal output buffer before printing an error
;    66          0066  1 !                 message.  JBS 02-FEB-1979
;    67          0067  1 ! 1-013 - Add support for I/O lists and change the name of the prefix
;    68          0068  1 !                 for stack frames from BAS$ to BSF$.  JBS 08-FEB-1979
;    69          0069  1 ! 1-014 - Because control C puts some non-BASIC frames on the stack,
;    70          0070  1 !                 be cleverer about searching through stack frames for a
;    71          0071  1 !                 non-GOSUB frame.  JBS 20-FEB-1979
;    72          0072  1 ! 1-015 - In BAS$$SIGNAL, don't force the severity to SEVERE ERROR
;    73          0073  1 !                 by calling LIB$STOP.  JBS 20-FEB-1979
;    74          0074  1 ! 1-016 - Search the PC table from back to front so that the line numbers
;    75          0075  1 !                 from statements which generate no code, such as DATA statements,
;    76          0076  1 !                 will not appear.  JBS 22-FEB-1979
;    77          0077  1 ! 1-017 - Use OTS$$PUR_IO_ERR to purge I/O buffers, thus avoiding having
;    78          0078  1 !                 to REQUIRE all of the I/O data structures.  JBS 07-MAR-1979
;    79          0079  1 ! 1-018 - Concatenate a ?, % or space on the front of error messages
;    80          0080  1 !                 in BAS$ERT based on the severity of the error.  JBS 12-MAR-1979
;    81          0081  1 ! 1-019 - In BAS$ERT, don't clobber the length field of a dynamic
;    82          0082  1 !                 string.  JBS 22-MAR-1979
;    83          0083  1 ! 1-020 - Change name of ILLEGAL RESUME.  JBS 02-APR-1979
;    84          0084  1 ! 1-021 - Make BAS$$COND_VAL global, so BAS$$SIGNAL_IO can use it.
;    85          0085  1 !                 JBS 06-APR-1979
;    86          0086  1 ! 1-022 - Only restart statements after restartable I/O failures if
;    87          0087  1 !                 the I/O was to a terminal.  JBS 06-APR-1979
;    88          0088  1 ! 1-023 - RESUME with no line number will resume into another module.
;    89          0089  1 !                 JBS 12-APR-1979
;    90          0090  1 ! 1-024 - The compiled code can get SS$_SUBRNG.  JBS 15-APR-1979
;    91          0091  1 ! 1-025 - Correct an error in edit 022.  JBS 16-APR-1979
;    92          0092  1 ! 1-026 - Correct an error in unwinding from a RESUME with no
;    93          0093  1 !                 line number.  JBS 30-APR-1979
;    94          0094  1 ! 1-027 - If the line number is not found, take the line number
;    95          0095  1 !                 corresponding to the next earlier PC.  This is needed
;    96          0096  1 !                 because (contrary to the specification) the compiler
;    97          0097  1 !                 does not put its "fake line numbers" in the line number
;    98          0098  1 !                 table.  JBS 04-MAY-1979
;    99          0099  1 ! 1-028 - If we are restarting an I/O statement, call BAS$$RESTART_IO
;   100          0100  1 !                 to reinitialize the I/O data base.  JBS 07-MAY-1979
;   101          0101  1 ! 1-029 - If we are doing system handling on an INFO message, don't
;   102          0102  1 !                 promote it to a warning.  JBS 10-MAY-1979
;   103          0103  1 ! 1-030 - If we convert a system message to a BASIC message, be sure
;   104          0104  1 !                 the PC and PSL of the failure are reported.  JBS 11-MAY-1979
;   105          0105  1 ! 1-031 - Publish the PC and PSL for any converted message.
;   106          0106  1 !                 JBS 13-MAY-1979
;   107          0107  1 ! 1-032 - Include certain string error codes in the list of messages
;   108          0108  1 !                 which are converted to BASIC-specific errors.  JBS 16-MAY-1979
;   109          0109  1 ! 1-033 - Convert LIB$S and OTS$S to STR$.  JBS 21-MAY-1979
;   110          0110  1 ! 1-034 - Correct an error in BAS$$USER_HAND which prevented intercepting
;   111          0111  1 !                 an error that had once been through ON ERROR GO BACK.
;   112          0112  1 !                 JBS 29-MAY-1979
;   113          0113  1 ! 1-035 - Add BAS$$ERR_INIT.  JBS 04-JUN-1979
;   114          0114  1 ! 1-036 - Call BAS$$UNWIND when cutting back a frame.  JBS 06-JUN-1979
```

```
;  115          0115  1 ! 1-037 - Defer calling SYS$UNWIND to the top level handler.
;  116          0116  1 !           JBS 06-JUN-1979
;  117          0117  1 ! 1-038 - BAS$$ERR_INIT must clear SYSTEM_ERROR and GONE_BACK.
;  118          0118  1 !           JBS 07-JUN-1979
;  119          0119  1 ! 1-039 - RESUME to a line number must accumulate the number of frames
;  120          0120  1 !           to unwind.  JBS 10-JUL-1979
;  121          0121  1 ! 1-040 - Change call to STR$COPY.  JBS 16-JUL-1979
;  122          0122  1 ! 1-041 - Fix a bug which caused GONE_BACK to remain set after an UNWIND.
;  123          0123  1 !           JBS 23-JUL-1979
;  124          0124  1 ! 1-042 - When unwinding to a frame, POP its I/O.  JBS 24-JUL-1979
;  125          0125  1 ! 1-043 - Change call to OTS$$TERM_IO.  JBS 26-JUL-1979
;  126          0126  1 ! 1-044 - Remove edit 023: don't allow RESUME into another module.
;  127          0127  1 !           JBS 26-JUL-1979
;  128          0128  1 ! 1-045 - Give error 31 (illegal byte count for I/O) in response to
;  129          0129  1 !           an attempt to do I/O to a closed file.  JBS 01-AUG-1979
;  130          0130  1 ! 1-046 - Don't try to build an argument list for LIB$SIGNAL longer
;  131          0131  1 !           than 255.  JBS 08-AUG-1979
;  132          0132  1 ! 1-047 - Correct a typo in edit 044.  JBS 20-AUG-1979
;  133          0133  1 ! 1-048 - Call BAS$$PUR_IO_ERR.  JBS 20-AUG-1979
;  134          0134  1 ! 1-049 - Translate MTH$_FLOOVEMAT into floating overflow, since it
;  135          0135  1 !           is produced by both the EXP and TAN functions.  JBS 20-AUG-1979
;  136          0136  1 ! 1-050 - Change BAS$HANDLER to BAS$$HANDLER for the sharable library.
;  137          0137  1 !           JBS 20-AUG-1979
;  138          0138  1 ! 1-051 - Move the definitions of the error codes to BAS$MSGDEF, for
;  139          0139  1 !           the sake of the shared library.  JBS 21-AUG-1979
;  140          0140  1 ! 1-052 - Remove the redundent RETURN statement, the BLISS compiler no
;  141          0141  1 !           longer needs it.  JBS 06-SEP-1979
;  142          0142  1 ! 1-053 - Add BAS$PUSH_ERR and BAS$POP_ERR.  JBS 10-SEP-1979
;  143          0143  1 ! 1-054 - Change IOL from I/O list to Immediate On-Line.  JBS 10-SEP-1979
;  144          0144  1 ! 1-055 - If a BASIC condition is signalled as INFO, don't promote
;  145          0145  1 !           it to a more severe condition.  This is needed for the
;  146          0146  1 !           two kinds of control C signals for the RUN command.
;  147          0147  1 !           JBS 14-SEP-1979
;  148          0148  1 ! 1-056 - Change MTH$_SINCOSSIG to MTH$_SIGLOSMAT.  JBS 19-SEP-1979
;  149          0149  1 ! 1-057 - Add STR$_STRTOOLON.  JBS 31-OCT-1979
;  150          0150  1 ! 1-058 - Make ERR, ERL and ERN$ retain their values after RESUME.
;  151          0151  1 !           JBS 07-NOV-1979
;  152          0152  1 ! 1-059 - Fix restarting an I/O statement to clear the error flag.
;  153          0153  1 !           JBS 08-NOV-1979
;  154          0154  1 ! 1-060 - Make sure that a user error handler doesn't try to handle
;  155          0155  1 !           INFO conditions.  This is a part of edit 055.  JBS 15-NOV-1979
;  156          0156  1 ! 1-061 - Handle correctly a main program with ON ERROR GO BACK getting
;  157          0157  1 !           a restartable error.  JBS 09-JAN-1980
;  158          0158  1 ! 1-062 - Handle delta PC values greater than 2^15.  JBS 12-FEB-1980
;  159          0159  1 ! 1-063 - Handle error trapping in a module without line numbers, except for
;  160          0160  1 !           RESUME with no line number.  JBS 07-MAR-1980
;  161          0161  1 ! 1-064 - Treat floating faults the same as traps IN BAS$$HANDLER.  SBL 10-Jun-1980
;  162          0162  1 ! 1-065 - Distiguish between a major and a minor frame in BAS$$USER_HAND, so
;  163          0163  1 !           when an error is ON ERROR GO BACK!TO 0 in a minor frame the major
;  164          0164  1 !           frame can handle the error. FM 13-FEB-81.
;  165          0165  1 ! 1-066 - Comments referring to SYSMSG.MPF are using an obsolete name; the name
;  166          0166  1 !           should be SYS$MESSAGE:SYSMSG.EXE. PL 26-Aug-81
;  167          0167  1 ! 1-067 - Convert SS$_DECOVF to equivalent BAS$ error.  PLL 5-Apr-1982
;  168          0168  1 ! 1-068 - Remove code that was a workaround for a bug in SYS$UNWIND (could not
;  169          0169  1 !           be called with an argument of zero).  In BAS$$USER_HAND, instead of
;  170          0170  1 !           patching the return PC of the frame that returns to the compiled code,
;  171          0171  1 !           do nothing and let BAS$$HANDLER call SYS$UNWIND.  This fixes a bug
```

```
;  172          0172  1 !           with integer overflow that occurs when the g, h emulator is invoked.
;  173          0173  1 !           PLL 26-Apr-82
;  174          0174  1 !  1-069 - Back out the last edit.  It broke STOP and possibly some other
;  175          0175  1 !           things.  Apparently calling SYS$UNWIND with zero does not always
;  176          0176  1 !           work as expected.  So put in a more obscure fix (in BAS$$HANDLER)
;  177          0177  1 !           for integer overflow with the g, h emulator.  PLL 11-May-1982
;  178          0178  1 !  1-070 - BAS$$HANDLER must check to see how many arguments in the signal
;  179          0179  1 !           argument list - sometimes there is no PSW.  PLL 11-May-1982
;  180          0180  1 !  1-071 - Give the new error message 'improper error handling' if a main
;  181          0181  1 !           program calls a subprogram in its error handler, and this sub-
;  182          0182  1 !           program does an ON ERROR GO BACK.  PLL 22-Jun-1982
;  183          0183  1 !  1-072 - make variables BAS$T_ERN, BAS$L_ERR and BAS$L_ERL globals, so
;  184          0184  1 !           that they are accessible from other routines (specifically BAS$CTRLC)
;  185          0185  1 !           also make BAS$$LINE & BAS$$MODULE global routines.  MDL 22-Jul-1982
;  186          0186  1 !  1-073 - BAS$$HANDLER should be prepared to restart an ANSI INPUT error.
;  187          0187  1 !           PLL 29-Jul-1982
;  188          0188  1 !  1-074 - don't set BAS$L_GOING_BACK in BAS$$USER_HAND if this is a restartable
;  189          0189  1 !           error.  MDL 29-Mar-1984
;  190          0190  1 !--
;  191          0191  1
;  192          0192  1 !<BLF/PAGE>
```

```
;  194          0193  1 !
;  195          0194  1 ! SWITCHES:
;  196          0195  1 !
;  197          0196  1
;  198          0197  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;  199          0198  1
;  200          0199  1 !
;  201          0200  1 ! LINKAGES:
;  202          0201  1 !
;  203          0202  1
;  204          0203  1 LINKAGE
;  205          0204  1     RESTART_LINK = JSB : NOTUSED (11, 10, 9)
;  206          0205  1     NOPRESERVE (8, 7, 6, 5, 4, 3, 2);
;  207          0206  1
;  208          0207  1 REQUIRE 'RTLIN:OTSLNK';                         ! Define common linkages
;  209          0636  1
;  210          0637  1 !
;  211          0638  1 ! TABLE OF CONTENTS:
;  212          0639  1 !
;  213          0640  1
;  214          0641  1 FORWARD ROUTINE
;  215          0642  1     BAS$$SIGNAL : NOVALUE,                      ! signal an error
;  216          0643  1     BAS$$STOP : NOVALUE,                        ! signal an error
;  217          0644  1     BAS$$COND_VAL,                              ! compute VAX/VMS cond value
;  218          0645  1     PC_TO_LINE_NO,                              ! convert PC to line number
;  219          0646  1     BAS$$LINE,                                  ! get the number of the current line
;  220          0647  1     BAS$$FUNCTION,                              ! get the name of the current function
;  221          0648  1     BAS$$MODULE,                                ! get the name of the current module
;  222          0649  1     HANDLER_HANDLER,                            ! handler for BAS$HANDLER
;  223          0650  1     RESTART : RESTART_LINK NOVALUE,             ! unwind target
;  224          0651  1     RESTART_IO : RESTART_LINK NOVALUE,          ! unwind target for I/O
;  225          0652  1     BAS$$USER_HAND,                             ! try to let user handle error
;  226          0653  1     BAS$RESUME,                                 ! resume from a condition handler
;  227          0654  1     BAS$RESUME_Z,                               ! likewise, but no line number
;  228          0655  1     BAS$ON_ERR_Z,                               ! ON ERROR GOTO 0
;  229          0656  1     BAS$ON_ERR_BK,                              ! ON ERROR GO BACK
;  230          0657  1     BAS$$HANDLER,                               ! handle a BASIC-PLUS-2 error
;  231          0658  1     BAS$ERL,                                    ! return error line number
;  232          0659  1     BAS$ERR,                                    ! return current error number
;  233          0660  1     BAS$ERN,                                    ! return module name of error
;  234          0661  1     BAS$ERT,                                    ! return text of error number
;  235          0662  1     BAS$ERROR : NOVALUE,                        ! signal an error from compiled code
;  236          0663  1     BAS$$ERR_INIT : NOVALUE,                    ! Initialize for the RUN command
;  237          0664  1     BAS$PUSH_ERR,                               ! Save error info
;  238          0665  1     BAS$POP_ERR;                                ! Restore error info
;  239          0666  1
;  240          0667  1 !
;  241          0668  1 ! INCLUDE FILES:
;  242          0669  1 !
;  243          0670  1
;  244          0671  1 LIBRARY 'RTLSTARLE';                            ! system symbols
;  245          0672  1
;  246          0673  1 REQUIRE 'RTLIN:RTLPSECT';                       ! macros to declare psects
;  247          0768  1
;  248          0769  1 REQUIRE 'RTLIN:BASFRAME';                       ! define frame structure
;  249          0972  1
;  250          0973  1 REQUIRE 'RTLIN:BASERRMSG';                      ! Define ERROR_LIST macro.
```

```
;  251          1596  1
;  252          1597  1 REQUIRE 'RTLML:OTSLUB';                          ! Define LUB
;  253          1737  1
;  254          1738  1 !
;  255          1739  1 ! MACROS:
;  256          1740  1 !
;  257          1741  1 !       NONE
;  258          1742  1 !
;  259          1743  1 ! EQUATED SYMBOLS:
;  260          1744  1 !
;  261          1745  1 !+
;  262          1746  1 ! Define the special error codes used for I/O errors and traceback.
;  263          1747  1 !-
;  264          1748  1
;  265          1749  1 LITERAL
;  266          1750  1     ERR_TRACE_MAIN = 4089,                        ! main program
;  267          1751  1     ERR_TRACE_SUB = 4090,                         ! external subroutine
;  268          1752  1     ERR_TRACE_EXTF = 4091,                        ! external function
;  269          1753  1     ERR_TRACE_DEF = 4092,                         ! DEF procedure
;  270          1754  1     ERR_TRACE_DEFS = 4093,                        ! DEF* procedure
;  271          1755  1     ERR_TRACE_GOSB = 4094,                        ! GOSUB
;  272          1756  1     ERR_TRACE_ONER = 4095,                        ! condition handler
;  273          1757  1     ERR_TRACE_IOLST = 4087,                       ! Immediate mode code
;  274          1758  1     ERR_TRACE_PCPSL = 4086;                       ! user PC=!XL, PSL=!XL
;  275          1759  1
;  276          1760  1 !+
;  277          1761  1 ! Define the return values from BAS$$USER_HAND.
;  278          1762  1 !-
;  279          1763  1
;  280          1764  1 LITERAL
;  281          1765  1     USER_HAND_CONT = 0,                           ! Continue from point of error (or of unwind)
;  282          1766  1     USER_HAND_BACK = 1,                           ! Try caller's handler
;  283          1767  1     USER_HAND_FAIL = 2;                           ! Force system error processing
;  284          1768  1
;  285          1769  1 !+
;  286          1770  1 ! Define the return values from the user's error handler.
;  287          1771  1 ! None of those below implies RESUME with a line number.
;  288          1772  1 !-
;  289          1773  1
;  290          1774  1 LITERAL
;  291          1775  1     USER_ERR_RSUMZ = 0,                           ! RESUME with no line number
;  292          1776  1     USER_ERR_GOBK = 1,                            ! ON ERROR GO BACK
;  293          1777  1     USER_ERR_OEGZ = 2;                            ! ON ERROR GOTO 0
;  294          1778  1
;  295          1779  1 !+
;  296          1780  1 ! Define the coded values for system error handling.
;  297          1781  1 !-
;  298          1782  1
;  299          1783  1 LITERAL
;  300          1784  1     K_SYS_CONT = 1,                               ! Continue in line
;  301          1785  1     K_SYS_EXIT = 2,                               ! Exit the image (LIB$STOP)
;  302          1786  1     K_SYS_RESTART = 3;                            ! Restart the line which had the error
;  303          1787  1
;  304          1788  1 !
;  305          1789  1 ! PSECTS:
;  306          1790  1 !
;  307          1791  1 DECLARE_PSECTS (BAS);
```

```
;  308          1792  1 !
;  309          1793  1 ! OWN STORAGE:
;  310          1794  1 !
;  311          1795  1 GLOBAL
;  312          1796  1     BAS$T_ERN : BLOCK [8, BYTE] INITIAL (BYTE ( REP 8 OF (0))), ! descriptor for module name
;  313          1797  1     BAS$L_ERR : INITIAL (0),                                    ! current error code
;  314          1798  1     BAS$L_ERL : INITIAL (0);                                    ! line number of error
;  315          1799  1
;  316          1800  1 OWN
;  317          1801  1     BAS$L_ERRFLG : INITIAL (0),                                 ! 1 = error in progress
;  318          1802  1     HIGHEST_LEVEL : INITIAL (0),                                ! Level to unwind to on RESUME
;  319          1803  1     HIGHEST_FMP : INITIAL (0),                                  ! Frame to unwind to on RESUME
;  320          1804  1     ACCUM_LEVEL : INITIAL (0),                                  ! Level to unwind to on RESUME with a line number
;  321          1805  1     UNWIND_COUNT : INITIAL (0),                                 ! Level for top handler to unwind
;  322          1806  1     SYSTEM_ERROR : INITIAL (0),                                 ! Set for "fatal fatal" error
;  323          1807  1     GONE_BACK : INITIAL (0),                                    ! Set for ON ERROR GO BACK
;  324          1808  1     ERROR_STACK : VOLATILE VECTOR [2] INITIAL (0, 0),   ! Error stack
;  325          1809  1     ERROR_STACK_INI : VOLATILE INITIAL (0);     ! Init flag for ERROR_STACK
;  326          1810  1
;  327          1811  1 !+
;  328          1812  1 ! Some OWN storage is needed so that communication can take place
;  329          1813  1 ! between levels of BAS$$USER_HAND and to RESTART.
;  330          1814  1 !-
;  331          1815  1
;  332          1816  1 OWN
;  333          1817  1     BAS$A_CH_CUR_LN : INITIAL (0),                              ! restart PC
;  334          1818  1     BAS$L_GOING_BACK : INITIAL (0),                             ! 1 when "going back"
;  335          1819  1     BAS$A_RESTART : INITIAL (0);                                ! restart PC
;  336          1820  1
;  337          1821  1 !
;  338          1822  1 ! EXTERNAL REFERENCES:
;  339          1823  1 !
;  340          1824  1
;  341          1825  1 EXTERNAL ROUTINE
;  342          1826  1     LIB$MATCH_COND,                                             ! match condition codes
;  343          1827  1     LIB$SIGNAL : NOVALUE,                                       ! system error signaller
;  344          1828  1     LIB$STOP : NOVALUE,                                         ! system fatal error signaller
;  345          1829  1     SYS$UNWIND,                                                 ! unwind the stack
;  346          1830  1     LIB$FIXUP_FLT,                                              ! fix up reserved operands
;  347          1831  1     LIB$GET_VM,                                                 ! get storage
;  348          1832  1     LIB$FREE_VM,                                                ! free storage
;  349          1833  1     STR$CONCAT,                                                 ! Concatenate two strings
;  350          1834  1     STR$COPY_R,                                                 ! Copy a string by ref
;  351          1835  1     STR$COPY_DX,                                                ! Copy a string by desc
;  352          1836  1     SYS$GETMSG,                                                 ! get the message text for a signal condition
;  353          1837  1     BAS$INIT_ONERR,                                             ! run a condition handler
;  354          1838  1     BAS$$RESTART_IO,                                            ! Restart an I/O statement
;  355          1839  1     BAS$$PUR_IO_ERR : NOVALUE,                                  ! Purge I/O on an error
;  356          1840  1     OTS$$TERM_IO,                                               ! Test for terminal I/O
;  357          1841  1     BAS$$UNWIND : NOVALUE,                                      ! Purge a frame
;  358          1842  1     BAS$$UNWIND_IO : NOVALUE,                                   ! Purge a frame's I/O
;  359          1843  1     BAS$HANDLER;                                                ! Header for condition handler
;  360          1844  1
;  361          1845  1 !+
;  362          1846  1 ! The following symbols are defined in module BAS$MSGDEF
;  363          1847  1 !-
;  364          1848  1
```

```
;  365          1849  1 EXTERNAL LITERAL
;  366          1850  1     BAS$K_FAC_NO : UNSIGNED (12),                ! Facility code
;  367          1851  1     BAS$K_RESNO_ERR : UNSIGNED (8),              ! RESUME with no error
;  368          1852  1     BAS$K_ILLRESSUB : UNSIGNED (8),              ! Illegal RESUME to subroutine
;  369          1853  1     BAS$K_DIVBY_ZER : UNSIGNED (8),              ! Divide by zero
;  370          1854  1     BAS$K_IMASQUROO : UNSIGNED (8),              ! Imaginary square root
;  371          1855  1     BAS$K_ILLARGLOG : UNSIGNED (8),              ! Illegal argument to LOG
;  372          1856  1     BAS$K_INTERR : UNSIGNED (8),                 ! Integer error
;  373          1857  1     BAS$K_MEMMANVIO : UNSIGNED (8),              ! Memory management violation
;  374          1858  1     BAS$K_FLOPOIERR : UNSIGNED (8),              ! Floating point error
;  375          1859  1     BAS$K_SUBOUTRAN : UNSIGNED (8),              ! Subscript out of rnage
;  376          1860  1     BAS$K_MAXMEMEXC : UNSIGNED (8),              ! Maximum memory exceeded
;  377          1861  1     BAS$K_ILLBYTCOU : UNSIGNED (8),              ! Illegal byte count for I/O
;  378          1862  1     BAS$_ON_CHAFIL,                              !  " on channel n, file a"
;  379          1863  1     BAS$K_PROLOSSOR : UNSIGNED (8),              ! Program lost, sorry.
;  380          1864  1     BAS$K_STRTOOLON : UNSIGNED (8),              ! String too long
;  381          1865  1     BAS$K_DECERR : UNSIGNED (8),                 ! Decimal error or overflow
;  382          1866  1     BAS$K_IMPERRHAN : UNSIGNED (8);              ! Improper error handling
;  383          1867  1 !+
;  384          1868  1 ! The following VAX/VMS condition codes are used in this module
;  385          1869  1 !-
;  386          1870  1
;  387          1871  1 EXTERNAL LITERAL
;  388          1872  1 !+
;  389          1873  1 ! Attempt to compute the square root of a negative number.
;  390          1874  1 ! The result will be the reserved operand.
;  391          1875  1 !-
;  392          1876  1     MTH$_SQUROONEG,
;  393          1877  1 !+
;  394          1878  1 ! Attempt to compute the logarithm of 0, or of a negative number.
;  395          1879  1 ! The result will be the reserved operand.
;  396          1880  1 !-
;  397          1881  1     MTH$_LOGZERNEG,
;  398          1882  1 !+
;  399          1883  1 ! Attempt to raise E to a power so large that the result cannot
;  400          1884  1 ! be represented by the computer.  That power is about 88.  The
;  401          1885  1 ! result will be the reserved operand.
;  402          1886  1 !-
;  403          1887  1     MTH$_FLOOVEMAT,
;  404          1888  1 !+
;  405          1889  1 ! Attempt to raise a base, B, to a power, P, where this is undefined.
;  406          1890  1 ! For example, 0 raised to the 0 power is undefined.  The result
;  407          1891  1 ! will be the reserved operand.
;  408          1892  1 !-
;  409          1893  1     MTH$_UNDEXP,
;  410          1894  1 !+
;  411          1895  1 ! Attempt to take the SINE or COSINE of a number so large that,
;  412          1896  1 ! after taking the number modulo 2 PI, there is no information
;  413          1897  1 ! left.  This is caused by the fact that the computer keeps
;  414          1898  1 ! only the highest-order significant bits of a number.  The
;  415          1899  1 ! result will be the reserved operand.  Other functions in the
;  416          1900  1 ! math library may also signal this under similar conditions.
;  417          1901  1 !-
;  418          1902  1     MTH$_SIGLOSMAT,
;  419          1903  1 !+
;  420          1904  1 ! Attempt to allocate a dynamic string when there is
;  421          1905  1 ! not enough virtual memory left to hold it along with
```

```
;  422          1906  1 ! all of the other strings allocated.
;  423          1907  1 !-
;  424          1908  1     STR$_INSVIRMEM,
;  425          1909  1 !+
;  426          1910  1 ! Divide by zero in string arithmetic.
;  427          1911  1 !-
;  428          1912  1     STR$_DIVBY_ZER,
;  429          1913  1 !+
;  430          1914  1 ! Attempt to create a string longer than 65535 characters,
;  431          1915  1 ! the maximum length allowed by the VAX-11 string architecture.
;  432          1916  1 ! This can be the result of, for example, the concatenation of
;  433          1917  1 ! two 50,000 character strings.
;  434          1918  1 !-
;  435          1919  1     STR$_STRTOOLON,
;  436          1920  1 !+
;  437          1921  1 ! Attempt to continue to do I/O to a closed file.
;  438          1922  1 ! (That is, the file was closed between element
;  439          1923  1 ! transmitters, and another element transmission
;  440          1924  1 ! was attempted.)
;  441          1925  1 !-
;  442          1926  1     OTS$_IO_CONCLO;
;  443          1927  1 !+
;  444          1928  1 ! Attempt to compute a packed decimal result which the computer
;  445          1929  1 ! can not represent.
;  446          1930  1 !
;  447          1931  1 !   SS$_DECOVF                  ! (defined in RTLSTARLE)
;  448          1932  1
;  449          1933  1 !+
;  450          1934  1 ! Attempt to divide a real number by 0.
;  451          1935  1 !
;  452          1936  1 !   SS$_FLTDIV                  ! (defined in RTLSTARLE)
;  453          1937  1 !   SS$_FLTDIV_F (fault)        ! (defined in RTLSTARLE)
;  454          1938  1 !
;  455          1939  1 ! Attempt to divide an integer by 0.
;  456          1940  1 !
;  457          1941  1 !   SS$_INTDIV                  ! (defined in RTLSTARLE)
;  458          1942  1 !
;  459          1943  1 ! Attempt to compute a floating point result which the computer
;  460          1944  1 ! cannot represent.
;  461          1945  1 !
;  462          1946  1 !   SS$_FLTOVF                  ! (defined in RTLSTARLE)
;  463          1947  1 !   SS$_FLTOVF_F (fault)        ! (defined in RTLSTARLE)
;  464          1948  1 !
;  465          1949  1 ! Attempt to compute an integer result which the computer cannot
;  466          1950  1 ! represent.
;  467          1951  1 !
;  468          1952  1 !   SS$_INTOVF                  ! (defined in RTLSTARLE)
;  469          1953  1 !
;  470          1954  1 ! Reserved operand fault.  In the context of BASIC, this is usually
;  471          1955  1 ! caused by an attempt to refer to a reserved floating operand, but
;  472          1956  1 ! it can be caused by other errors.  Only the floating reserved
;  473          1957  1 ! operand case is handled by BASIC.
;  474          1958  1 !
;  475          1959  1 !   SS$_ROPRAND                 ! (define in RTLSTARLE)
;  476          1960  1 !
;  477          1961  1 ! Attempt to refer to an invalid address.  This can happen if
;  478          1962  1 ! range checking on array indicies is defeated.
```

```
;  479          1963  1 !
;  480          1964  1 !   SS$_ACCVIO                  ! (defined in RTLSTARLE)
;  481          1965  1 !
;  482          1966  1 ! Attempt to use an index outside its proper range.  This can happen
;  483          1967  1 ! when the compiler generates in-line array indexing.
;  484          1968  1 !
;  485          1969  1 !   SS$_SUBRNG                  ! (defined in RTLSTARLE)
;  486          1970  1 !
;  487          1971  1
;  488          1972  1 EXTERNAL
;  489          1973  1         OTS$$A_CUR_LUB : ADDRESSING_MODE (GENERAL);
;  490          1974  1                                  ! Addr of current LUB/ISB/RAB
;  491          1975  1 !<BLF/PAGE>
```

```
;  493          1976  1 !
;  494          1977  1 ! Define the severity code for each possible error.
;  495          1978  1 !
;  496          1979  1
;  497          1980  1 MACRO
;  498        M 1981  1     ERR (NUMBER, CODE, TEXT, SEVERITY, SYSTEM_HANDLING) =
;  499        M 1982  1          (%IF (%IDENTICAL (SEVERITY, WARNING)) %THEN STS$K_WARNING %ELSE
;  500        M 1983  1          %IF (%IDENTICAL (SEVERITY, SUCCESS)) %THEN STS$K_SUCCESS %ELSE
;  501        M 1984  1          %IF (%IDENTICAL (SEVERITY, ERROR)) %THEN STS$K_ERROR %ELSE
;  502        M 1985  1          %IF (%IDENTICAL (SEVERITY, INFO)) %THEN STS$K_INFO %ELSE
;  503        M 1986  1          %IF (%IDENTICAL (SEVERITY, SEVERE)) %THEN STS$K_SEVERE %ELSE
;  504        M 1987  1          257 %FI %FI %FI %FI %FI)
;  505          1988  1     %;
;  506          1989  1
;  507          1990  1 BIND
;  508          1991  1     ERR_SEVERITY = UPLIT BYTE(ERROR_LIST) : VECTOR [256, BYTE];
;  509          1992  1
;  510          1993  1 UNDECLARE %QUOTE
;  511          1994  1     ERR;
;  512          1995  1
;  513          1996  1 !
;  514          1997  1 ! Define the system handling option for each error.
;  515          1998  1 !
;  516          1999  1
;  517          2000  1 MACRO
;  518        M 2001  1     ERR (NUMBER, CODE, TEXT, SEVERITY, SYSTEM_HANDLING) =
;  519        M 2002  1          (%IF (%IDENTICAL (SYSTEM_HANDLING, CONT)) %THEN K_SYS_CONT %ELSE
;  520        M 2003  1          %IF (%IDENTICAL (SYSTEM_HANDLING, EXIT)) %THEN K_SYS_EXIT %ELSE
;  521        M 2004  1          %IF (%IDENTICAL (SYSTEM_HANDLING, RESTART)) %THEN K_SYS_RESTART %ELSE
;  522        M 2005  1          257 %FI %FI %FI)
;  523          2006  1     %;
;  524          2007  1
;  525          2008  1 BIND
;  526          2009  1     ERR_SYSTEM = UPLIT BYTE(ERROR_LIST) : VECTOR [256, BYTE];
;  527          2010  1
;  528          2011  1 UNDECLARE %QUOTE
;  529          2012  1     ERR;
;  530          2013  1
;  531          2014  1 !<BLF/PAGE>
```

```
;   533          2015  1 !+
;   534          2016  1 ! The following field set represents an item pushed onto the
;   535          2017  1 ! error stack.  It contains the entire state of the error system.
;   536          2018  1 ! It is used when it is necessary to save the error state to run
;   537          2019  1 ! the compiler to compile an immediate mode statement.
;   538          2020  1 !-
;   539          2021  1
;   540          2022  1 FIELD
;   541          2023  1     PUSH_ITEM =
;   542          2024  1         SET
;   543          2025  1         PUSH$A_NEXT = [0, 0, %BPVAL, 0],        ! Next item
;   544          2026  1         PUSH$A_PREV = [4, 0, %BPVAL, 0],        ! Previous item
;   545          2027  1         PUSH$L_ERRFLG = [8, 0, %BPVAL, 0],      ! 1 = error in progress
;   546          2028  1         PUSH$T_ERN = [12, 0, 0, 0],             ! Module name of error
;   547          2029  1         PUSH$L_ERR = [20, 0, %BPVAL, 0],        ! Error number
;   548          2030  1         PUSH$L_ERL = [24, 0, %BPVAL, 0],        ! Line number of error
;   549          2031  1         PUSH$L_HGH_LVL = [28, 0, %BPVAL, 0],    ! Level for RESUME
;   550          2032  1         PUSH$A_HGH_FMP = [32, 0, %BPADDR, 0],   ! Frame for RESUME
;   551          2033  1         PUSH$L_ACC_LVL = [36, 0, %BPVAL, 0],    ! Level for RESUME-line
;   552          2034  1         PUSH$L_UNW_CNT = [40, 0, %BPVAL, 0],    ! Amount to unwind at top
;   553          2035  1         PUSH$L_SYS_ERR = [44, 0, %BPVAL, 0],    ! 1 = "fatal fatal" error
;   554          2036  1         PUSH$L_GONE_BAK = [48, 0, %BPVAL, 0],   ! 1 = ON ERROR GO BACK
;   555          2037  1         PUSH$A_CUR_IN = [52, 0, %BPADDR, 0],    ! Restart PC
;   556          2038  1         PUSH$L_GOING_BACK = [56, 0, %BPVAL, 0], ! Restart flag
;   557          2039  1         PUSH$A_RESTART = [60, 0, %BPADDR, 0]    ! Real restart PC
;   558          2040  1         TES;
;   559          2041  1
;   560          2042  1 LITERAL
;   561          2043  1     PUSH$K_LENGTH = 64;                         ! Number of bytes to allocate
;   562          2044  1
```

```
;  564          2045  1 GLOBAL ROUTINE BAS$$SIGNAL (                          ! signal an error
;  565          2046  1         ERR_CODE                                      ! the BASIC error code
;  566          2047  1     ) : NOVALUE =
;  567          2048  1
;  568          2049  1 !++
;  569          2050  1 ! FUNCTIONAL DESCRIPTION:
;  570          2051  1 !
;  571          2052  1 !       Signal an error for BASIC-PLUS-2/VAX.  The argument is the
;  572          2053  1 !       BASIC-PLUS-2 error code.
;  573          2054  1 !
;  574          2055  1 ! FORMAL PARAMETERS:
;  575          2056  1 !
;  576          2057  1 !       ERR_CODE.rl.v   The BASIC-PLUS-2 error code.  The codes and
;  577          2058  1 !                       their meanings are listed in file BASERRMSG.REQ.
;  578          2059  1 !
;  579          2060  1 ! IMPLICIT INPUTS:
;  580          2061  1 !
;  581          2062  1 !       NONE
;  582          2063  1 !
;  583          2064  1 ! IMPLICIT OUTPUTS:
;  584          2065  1 !
;  585          2066  1 !       NONE
;  586          2067  1 !
;  587          2068  1 ! ROUTINE VALUE:
;  588          2069  1 !
;  589          2070  1 !       NONE
;  590          2071  1 !
;  591          2072  1 ! COMPLETION CODES:
;  592          2073  1 !
;  593          2074  1 !       NONE
;  594          2075  1 !
;  595          2076  1 ! SIDE EFFECTS:
;  596          2077  1 !
;  597          2078  1 !       May never return to the caller.
;  598          2079  1 !
;  599          2080  1 !--
;  600          2081  1
;  601          2082  2     BEGIN
;  602          2083  2
;  603          2084  2     LOCAL
;  604          2085  2         VAX_11_COND_VAL : BLOCK [4, BYTE];      ! 32-bit VAX/VMS condition value
;  605          2086  2
;  606          2087  2     VAX_11_COND_VAL = BAS$$COND_VAL (.ERR_CODE);
;  607          2088  2 !+
;  608          2089  2 ! The line number, module name and function name are added in
;  609          2090  2 ! BAS$HANDLER for each level that this signal goes through.
;  610          2091  2 !-
;  611          2092  2     LIB$SIGNAL (.VAX_11_COND_VAL);
;  612          2093  1     END;                                            ! of BAS$$SIGNAL


                                                        .TITLE  BAS$ERROR
                                                        .IDENT  \1-074\

                                                        .PSECT  _BAS$DATA,NOEXE,  PIC,2

                                                00# 00000 BAS$T_ERN::
```

```
                                                                     .BYTE    0[8]                                          ;
                                                00000000  00008 BAS$L_ERR::
                                                                     .LONG    0                                             ;
                                                00000000  0000C BAS$L_ERL::
                                                                     .LONG    0                                             ;
                                                00000000  00010 BAS$L_ERRFLG:
                                                                     .LONG    0                                             ;
                                                00000000  00014 HIGHEST_LEVEL:
                                                                     .LONG    0                                             ;
                                                00000000  00018 HIGHEST_FMP:
                                                                     .LONG    0                                             ;
                                                00000000  0001C ACCUM_LEVEL:
                                                                     .LONG    0                                             ;
                                                00000000  00020 UNWIND_COUNT:
                                                                     .LONG    0                                             ;
                                                00000000  00024 SYSTEM_ERROR:
                                                                     .LONG    0                                             ;
                                                00000000  00028 GONE_BACK:
                                                                     .LONG    0                                             ;
                                      00000000  00000000  0002C ERROR_STACK:
                                                                     .LONG    0, 0                                          ;
                                                00000000  00034 ERROR_STACK_INI:
                                                                     .LONG    0                                             ;
                                                00000000  00038 BAS$A_CH_CUR_LN:
                                                                     .LONG    0                                             ;
                                                00000000  0003C BAS$L_GOING_BACK:
                                                                     .LONG    0                                             ;
                                                00000000  00040 BAS$A_RESTART:
                                                                     .LONG    0                                             ;

                                                                     .PSECT   _BAS$CODE,NOWRT,  SHR,  PIC,2

02 02 02 02 02 02 02 02 02 02 02 02 02 02 03  00000 P.AAA:   .BYTE    3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02  0000F                   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
02 02 02 02 02 02 04 04 04 04 04 02 02 02 02  0001E                   2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 2, 2, 2, - ;
02 02 02 02 02 02 02 02 02 00 02 02 02 02 02  0002D                   2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, - ;
04 04 04 04 04 04 04 02 02 02 02 02 02 02 02  0003C                   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, - ;
04 04 04 02 04 04 04 04 04 04 04 04 04 04 04  0004B                   4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, - ;
04 04 04 04 04 04 04 04 04 04 04 04 04 04 00  0005A                   4, 4, 2, 4, 4, 4, 0, 4, 4, 4, 4, 4, 4, 4, - ;
04 04 04 04 04 04 04 04 04 04 03 04 04 00 02  00069                   4, 4, 4, 4, 4, 4, 4, 2, 0, 4, 4, 3, 4, 4, - ;
02 02 02 02 02 02 02 04 04 04 04 03 04 04 04  00078                   4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, - ;
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02  00087                   4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02  00096                   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
02 02 00 00 02 02 02 02 02 02 04 02 02 02 02  000A5                   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
04 04 04 04 04 00 02 04 04 04 04 02 02 02 02  000B4                   2, 4, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, - ;
04 04 04 04 04 04 04 04 04 04 04 04 04 04 04  000C3                   2, 2, 4, 4, 4, 4, 2, 0, 4, 4, 4, 4, 4, 4, - ;
04 04 04 04 04 04 04 04 04 04 04 04 04 04 04  000D2                   4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, - ;
02 02 02 02 02 02 02 02 02 02 04 02 02 04 04  000E1                   4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, - ;
04 02 02 02 02 02 02 04 04 04 02 02 02 02 02  000F0                   4, 4, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, - ;
                                          04  000FF                   2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 2, 2, 2, 2, - ;
                                                                      2, 2, 4, 4                                  ;
02 02 02 02 02 02 02 02 02 02 02 02 02 02 01  00100 P.AAB:   .BYTE    1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02  0010F                   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02  0011E                   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
02 02 02 02 02 02 02 03 02 03 02 02 02 02 02  0012D                   2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 3, 2, 2, 2, - ;
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02  0013C                   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02  0014B                   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, - ;
```

```
02  02  02  02  02  02  02  02  02  02  02  02  02  02  01  0015A          2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, -  ;
02  02  02  02  02  02  02  02  02  02  01  02  02  01  02  00169          2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, -  ;
02  02  02  02  02  02  02  02  02  02  02  01  02  02  02  00178          2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, -  ;
02  02  02  02  02  02  02  02  02  02  02  02  02  02  02  00187          2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, -  ;
02  02  02  02  02  02  02  02  02  02  02  02  02  02  02  00196          2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, -  ;
02  02  03  01  02  02  02  02  02  02  02  02  02  02  02  001A5          2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, -  ;
02  02  02  02  02  03  02  02  02  02  02  02  02  02  02  001B4          2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, -  ;
02  02  02  02  02  02  02  02  02  02  02  02  02  02  02  001C3          2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, -  ;
02  02  02  02  02  02  02  02  02  02  02  02  02  02  02  001D2          2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, -  ;
02  02  02  02  02  02  02  02  02  02  02  02  02  02  02  001E1          2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, -  ;
02  02  02  02  02  02  02  02  02  02  02  02  02  02  02  001F0          2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, -  ;
                                    02  001FF          2, 2, 2, 2

                                                ERR_SEVERITY=       P.AAA
                                                ERR_SYSTEM=         P.AAB
                                                        .EXTRN  LIB$MATCH_COND, LIB$SIGNAL
                                                        .EXTRN  LIB$STOP, SYS$UNWIND
                                                        .EXTRN  LIB$FIXUP_FLT, LIB$GET_VM
                                                        .EXTRN  LIB$FREE_VM, STR$CONCAT
                                                        .EXTRN  STR$COPY_R, STR$COPY_DX
                                                        .EXTRN  SYS$GETMSG, BAS$INIT_ONERR
                                                        .EXTRN  BAS$$RESTART_IO
                                                        .EXTRN  BAS$$PUR_IO_ERR
                                                        .EXTRN  OTS$$TERM_IO, BAS$$UNWIND
                                                        .EXTRN  BAS$$UNWIND_IO, BAS$HANDLER
                                                        .EXTRN  BAS$K_FAC_NO, BAS$K_RESNO_ERR
                                                        .EXTRN  BAS$K_ILLRESSUB
                                                        .EXTRN  BAS$K_DIVBY_ZER
                                                        .EXTRN  BAS$K_IMASQUROO
                                                        .EXTRN  BAS$K_ILLARGLOG
                                                        .EXTRN  BAS$K_INTERR, BAS$K_MEMMANVIO
                                                        .EXTRN  BAS$K_FLOPOIERR
                                                        .EXTRN  BAS$K_SUBOUTRAN
                                                        .EXTRN  BAS$K_MAXMEMEXC
                                                        .EXTRN  BAS$K_ILLBYTCOU
                                                        .EXTRN  BAS$_ON_CHAFIL, BAS$K_PROLOSSOR
                                                        .EXTRN  BAS$R_STRTOOLON
                                                        .EXTRN  BAS$K_DECERR, BAS$K_IMPERRHAN
                                                        .EXTRN  MTH$_SQUROONEG, MTH$_LOGZERNEG
                                                        .EXTRN  MTH$_FLOOVEMAT, MTH$_UNDEXP
                                                        .EXTRN  MTH$_SIGLOSMAT, STR$_INSVIRMEM
                                                        .EXTRN  STR$_DIVBY_ZER, STR$_STRTOOLON
                                                        .EXTRN  OTS$_IO_CONCLO, OTS$$A_CUR_LUB

                                    0000 00000          .ENTRY  BAS$$SIGNAL, Save nothing                       ; 2045
                            04  AC  DD 00002          PUSHL   ERR_CODE                                        ; 2087
                      0000V  CF     01  FB 00005          CALLS   #1, BAS$$COND_VAL                           ;
                                    50  DD 0000A          PUSHL   VAX_11_COND_VAL                             ; 2092
                00000000G  00       01  FB 0000C          CALLS   #1, LIB$SIGNAL                              ;
                                        04 00013          RET                                                 ; 2093

; Routine Size:  20 bytes,    Routine Base:  _BAS$CODE + 0200

;  613          2094  1
```

```
;  615          2095  1 GLOBAL ROUTINE BAS$$STOP (                        ! signal an error
;  616          2096  1         ERR_CODE                                  ! the BASIC error code
;  617          2097  1     ) : NOVALUE =
;  618          2098  1
;  619          2099  1 !++
;  620          2100  1 ! FUNCTIONAL DESCRIPTION:
;  621          2101  1 !
;  622          2102  1 !       Signal an error for BASIC-PLUS-2/VAX.  The argument is the
;  623          2103  1 !       BASIC-PLUS-2 error code.
;  624          2104  1 !
;  625          2105  1 ! FORMAL PARAMETERS:
;  626          2106  1 !
;  627          2107  1 !       ERR_CODE.rl.v   The BASIC-PLUS-2 error code.  The codes and
;  628          2108  1 !                       their meanings are listed in file BASERRMSG.REQ.
;  629          2109  1 !                       The severity must be ERROR or SEVERE ERROR.
;  630          2110  1 !
;  631          2111  1 ! IMPLICIT INPUTS:
;  632          2112  1 !
;  633          2113  1 !       NONE
;  634          2114  1 !
;  635          2115  1 ! IMPLICIT OUTPUTS:
;  636          2116  1 !
;  637          2117  1 !       NONE
;  638          2118  1 !
;  639          2119  1 ! ROUTINE VALUE:
;  640          2120  1 !
;  641          2121  1 !       NONE
;  642          2122  1 !
;  643          2123  1 ! COMPLETION CODES:
;  644          2124  1 !
;  645          2125  1 !       NONE
;  646          2126  1 !
;  647          2127  1 ! SIDE EFFECTS:
;  648          2128  1 !
;  649          2129  1 !       Never returns to the caller.
;  650          2130  1 !
;  651          2131  1 !--
;  652          2132  1
;  653          2133  2     BEGIN
;  654          2134  2
;  655          2135  2     LOCAL
;  656          2136  2         VAX_11_COND_VAL : BLOCK [4, BYTE];      ! 32-bit VAX/VMS condition value
;  657          2137  2
;  658          2138  2     VAX_11_COND_VAL = BAS$$COND_VAL (.ERR_CODE);
;  659          2139  2 !+
;  660          2140  2 ! The line number, module name and function name are added in
;  661          2141  2 ! BAS$HANDLER for each level that this signal goes through.
;  662          2142  2 !-
;  663          2143  2     LIB$STOP (.VAX_11_COND_VAL);
;  664          2144  1     END;                                          ! of BAS$$STOP


                                0000 00000          .ENTRY  BAS$$STOP, Save nothing                 ; 2095
                        04   AC  DD 00002          PUSHL   ERR_CODE                                ; 2138
```

```
                    0000V  CF          01  FB 00005        CALLS   #1, BAS$$COND_VAL                        ;
                                       50  DD 0000A        PUSHL   VAX_11_COND_VAL                         ; 2143
                00000000G  00          01  FB 0000C        CALLS   #1, LIB$STOP                            ;
                                           04 00013        RET                                             ; 2144

; Routine Size:  20 bytes,    Routine Base:  _BAS$CODE + 0214

;  665          2145  1
```

```
;  667     2146  1 GLOBAL ROUTINE BAS$$COND_VAL (                    ! Compute condition value
;  668     2147  1         ERR_CODE                                   ! BASIC error code
;  669     2148  1     ) =
;  670     2149  1
;  671     2150  1 !++
;  672     2151  1 ! FUNCTIONAL DESCRIPTION:
;  673     2152  1 !
;  674     2153  1 !       Convert a BASIC error code to its 32-bit VAX/VMS error code.
;  675     2154  1 !       Conversion is done by copying the BASIC error number to the
;  676     2155  1 !       code field, setting the severity field based on the entry in
;  677     2156  1 !       the severity table for the code, and setting the facility
;  678     2157  1 !       to BAS$K_FAC_NO.  The facility specific bit is also set.
;  679     2158  1 !
;  680     2159  1 ! FORMAL PARAMETERS:
;  681     2160  1 !
;  682     2161  1 !       ERR_CODE.rl.v   The BASIC-PLUS-2 error code.  The codes and
;  683     2162  1 !                       their meanings are listed in file BASERRMSG.REQ.
;  684     2163  1 !
;  685     2164  1 ! IMPLICIT INPUTS:
;  686     2165  1 !
;  687     2166  1 !       NONE
;  688     2167  1 !
;  689     2168  1 ! IMPLICIT OUTPUTS:
;  690     2169  1 !
;  691     2170  1 !       NONE
;  692     2171  1 !
;  693     2172  1 ! ROUTINE VALUE:
;  694     2173  1 !
;  695     2174  1 !       The 32-bit VAX/VMS error code.
;  696     2175  1 !
;  697     2176  1 ! COMPLETION CODES:
;  698     2177  1 !
;  699     2178  1 !       NONE
;  700     2179  1 !
;  701     2180  1 ! SIDE EFFECTS:
;  702     2181  1 !
;  703     2182  1 !       NONE
;  704     2183  1 !
;  705     2184  1 !--
;  706     2185  1
;  707     2186  2     BEGIN
;  708     2187  2
;  709     2188  2     LOCAL
;  710     2189  2         RESULT : BLOCK [4, BYTE];                  ! 32-bit VAX/VMS condition value
;  711     2190  2
;  712     2191  2     RESULT = 0;
;  713     2192  2     RESULT [STS$V_SEVERITY] = (IF (.ERR_CODE GTRU 255) THEN STS$K_INFO ELSE .ERR_SEVERITY [.ERR_CODE]);
;  714     2193  2     RESULT [STS$V_CODE] = .ERR_CODE;
;  715     2194  2     RESULT [STS$V_FAC_SP] = 1;
;  716     2195  2     RESULT [STS$V_FAC_NO] = BAS$K_FAC_NO;
;  717     2196  2     RETURN (.RESULT);
;  718     2197  1     END;                                           ! of BAS$$COND_VAL
```

```
                                                  0000 00000          .ENTRY  BAS$$COND_VAL, Save nothing                     ; 2146
                                              51  D4 00002          CLRL    RESULT                                          ; 2191
                          000000FF    8F      04  AC  D1 00004          CMPL    ERR_CODE, #255                                  ; 2192
                                                  05  1B 0000C          BLEQU   1$                                              ;
                                      50          03  D0 0000E          MOVL    #3, R0                                          ;
                                                  0A  11 00011          BRB     2$                                              ;
                                      50      FDC1    CF  9E 00013 1$:      MOVAB   ERR_SEVERITY, R0                        ;
                                      50        04 BC40   9A 00018          MOVZBL  @ERR_CODE[R0], R0                       ;
            51            03                    00        50  F0 0001D 2$:      INSV    R0, #0, #3, RESULT                      ;
            51            0C                    03        04  AC F0 00022          INSV    ERR_CODE, #3, #12, RESULT               ; 2193
                                              51      8000    8F  A8 00028          BISW2   #32768, RESULT                          ; 2194
            51            0C                  10 00000000G    8F  F0 0002D          INSV    #BAS$K_FAC_NO, #16, #12, RESULT         ; 2195
                                      50                  51  D0 00036          MOVL    RESULT, R0                              ; 2196
                                                          04 00039          RET                                             ; 2197

; Routine Size:  58 bytes,    Routine Base:  _BAS$CODE + 0228

;  719          2198  1
```

```
;   721          2199  1 ROUTINE PC_TO_LINE_NO (                       ! Convert PC to line number
;   722          2200  1         FMP,                                   ! Frame containing line
;   723          2201  1         PC                                     ! PC to be converted
;   724          2202  1     ) =
;   725          2203  1
;   726          2204  1 !++
;   727          2205  1 ! FUNCTIONAL DESCRIPTION:
;   728          2206  1 !
;   729          2207  1 !       Compute the BASIC line number corresponding to a PC value.
;   730          2208  1 !
;   731          2209  1 ! FORMAL PARAMETERS:
;   732          2210  1 !
;   733          2211  1 !       FMP.ra.v        Address of the frame from which we want the
;   734          2212  1 !                       line number.
;   735          2213  1 !       PC.rlu.v        The program counter corresponding to the
;   736          2214  1 !                       line number.  If no exact match is found, use
;   737          2215  1 !                       the next lower PC value.
;   738          2216  1 !
;   739          2217  1 ! IMPLICIT INPUTS:
;   740          2218  1 !
;   741          2219  1 !       The (delta PC, line number) table, pointed to by the FCD
;   742          2220  1 !       for the main procedure.
;   743          2221  1 !
;   744          2222  1 ! IMPLICIT OUTPUTS:
;   745          2223  1 !
;   746          2224  1 !       NONE
;   747          2225  1 !
;   748          2226  1 ! ROUTINE VALUE:
;   749          2227  1 !
;   750          2228  1 !       The line number, as a 32-bit binary value.
;   751          2229  1 !
;   752          2230  1 ! COMPLETION CODES:
;   753          2231  1 !
;   754          2232  1 !       NONE
;   755          2233  1 !
;   756          2234  1 ! SIDE EFFECTS:
;   757          2235  1 !
;   758          2236  1 !       NONE
;   759          2237  1 !
;   760          2238  1 !--
;   761          2239  1
;   762          2240  2     BEGIN
;   763          2241  2
;   764          2242  2     MAP
;   765          2243  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;   766          2244  2
;   767          2245  2     LOCAL
;   768          2246  2         BSF$A_MAJOR_STG : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME),
;   769          2247  2         PC_DELTA_TABLE : REF VECTOR,
;   770          2248  2         SEARCH_ARG;
;   771          2249  2
;   772          2250  2 !+
;   773          2251  2 ! If the PC is zero, the cell that held it must not have been
;   774          2252  2 ! set up.  This means that we are trying to find the PC for
;   775          2253  2 ! a routine in which the first statement has not yet started
;   776          2254  2 ! execution.  Return a zero to indicate this.
;   777          2255  2 !-
```

```
;  778          2256  2
;  779          2257  2      IF (.PC EQLA 0) THEN RETURN (0);
;  780          2258  2
;  781          2259  2  !+
;  782          2260  2  ! Load the pointer to the major procedure's frame.
;  783          2261  2  !-
;  784          2262  2      BSF$A_MAJOR_STG = .FMP [BSF$A_BASE_R11];
;  785          2263  2  !+
;  786          2264  2  ! Get a pointer to the (PC delta, line number) table.
;  787          2265  2  ! This requires skipping over the text of the module name.
;  788          2266  2  !-
;  789          2267  2      PC_DELTA_TABLE = .BSF$A_MAJOR_STG [BSF$A_PROC_INFO] +       !
;  790          2268  2      ((.(.BSF$A_MAJOR_STG [BSF$A_PROC_INFO])) AND 255) + 1;
;  791          2269  2  !+
;  792          2270  2  ! Compute the PC relative to the beginning of the code.
;  793          2271  2  !-
;  794          2272  2      SEARCH_ARG = .PC - .BSF$A_MAJOR_STG [BSF$A_CODE_BEG];
;  795          2273  2  !+
;  796          2274  2  ! Search the table.  If an exact match cannot be found, use the
;  797          2275  2  ! line number just before the PC.
;  798          2276  2  !-
;  799          2277  2
;  800          2278  2      DECR TABLE_INDEX FROM .PC_DELTA_TABLE [0] TO 1 DO
;  801          2279  3          BEGIN
;  802          2280  3
;  803          2281  4          IF ((((.PC_DELTA_TABLE [.TABLE_INDEX])^-16) AND 65535) LEQU .SEARCH_ARG)
;  804          2282  3          THEN
;  805          2283  3              RETURN ((.(PC_DELTA_TABLE [.TABLE_INDEX])) AND 65535);
;  806          2284  3
;  807          2285  3          END;
;  808          2286  2
;  809          2287  2  !+
;  810          2288  2  ! We get here only if the number cannot be found in the table.  This
;  811          2289  2  ! means that the PC was stored by a fake line number before the first
;  812          2290  2  ! real line number.  This is so unreasonable that it is more likely
;  813          2291  2  ! due to a bug in either the compiler or the RTL.  To make the problem
;  814          2292  2  ! more visible, return a -1.
;  815          2293  2  !-
;  816          2294  2      RETURN (-1);
;  817          2295  1      END;                                        ! of PC_TO_LINE_NO


                                000C 00000 PC_TO_LINE_NO:
                                           .WORD   Save R2,R3                                          ; 2199
                             08 AC D5 00002        TSTL    PC                                           ; 2257
                                3C 13 00005        BEQL    3$                                           ;
                          50    04 AC D0 00007        MOVL    FMP, R0                                      ; 2262
                          50    F4 A0 D0 0000B        MOVL    -12(R0), BSF$A_MAJOR_STG                     ;
                          51 00AB D0 9A 0000F        MOVZBL  @171(BSF$A_MAJOR_STG), R1                    ; 2268
                          51 00AB C0 C0 00014        ADDL2   171(BSF$A_MAJOR_STG), R1                     ;
                                51 D6 00019        INCL    PC_DELTA_TABLE                               ;
                53    08 AC 0083 C0 C3 0001B        SUBL3   13T(BSF$A_MAJOR_STG), PC, SEARCH_ARG         ; 2272
                50       61      01 C1 00022        ADDL3   #1, (PC_DELTA_TABLE), TABLE_INDEX            ; 2281
                                14 11 00026        BRB     2$                                           ;
```

```
                52        6140      F0   8F   78 00028 1$:     ASHL    #-16, (PC_DELTA_TABLE)[TABLE_INDEX], R2  ;
          53    52          10           00   ED 0002E          CMPZV   #0, #16, R2, SEARCH_ARG                  ;
                                           07   1A 00033          BGTRU   2$                                       ;
                                      6140 DF   00035          PUSHAL  (PC_DELTA_TABLE)[TABLE_INDEX]            ; 2283
                                        50 9E   3C 00038          MOVZWL  @(SP)+, R0                               ;
                                           04   0003B          RET                                              ;
                                        E9      50 F5 0003C 2$:  SOBGTR  TABLE_INDEX, 1$                         ; 2278
                                        50      01 CE 0003F      MNEGL   #1, R0                                  ; 2294
                                           04   00042          RET                                              ;
                                           50   D4 00043 3$:     CLRL    R0                                      ; 2295
                                           04   00045          RET                                              ;

; Routine Size:  70 bytes,    Routine Base:  _BAS$CODE + 0262
```

```
;  819          2296  1 GLOBAL ROUTINE BAS$$LINE (                        ! Get current line number
;  820          2297  1                 FMP                               ! Current frame
;  821          2298  1     ) =
;  822          2299  1
;  823          2300  1 !++
;  824          2301  1 ! FUNCTIONAL DESCRIPTION:
;  825          2302  1 !
;  826          2303  1 !       Compute the number of the line now in execution.  At the
;  827          2304  1 !       beginning of each line, the BASIC compiler issues
;  828          2305  1 !
;  829          2306  1 !               MOVAB   -3(PC),BSF$A_MARK(FP)
;  830          2307  1 !
;  831          2308  1 !       which stores the PC of the MOVAB.  This routine then
;  832          2309  1 !       scans the PC delta table (also produced by the compiler)
;  833          2310  1 !       to find the correct line number.
;  834          2311  1 !
;  835          2312  1 ! FORMAL PARAMETERS:
;  836          2313  1 !
;  837          2314  1 !       FMP.ra.v        Address of the frame from which we want the
;  838          2315  1 !                       line number.
;  839          2316  1 !
;  840          2317  1 ! IMPLICIT INPUTS:
;  841          2318  1 !
;  842          2319  1 !       NONE
;  843          2320  1 !
;  844          2321  1 ! IMPLICIT OUTPUTS:
;  845          2322  1 !
;  846          2323  1 !       NONE
;  847          2324  1 !
;  848          2325  1 ! ROUTINE VALUE:
;  849          2326  1 !
;  850          2327  1 !       The line number, as a 32-bit binary value.
;  851          2328  1 !
;  852          2329  1 ! COMPLETION CODES:
;  853          2330  1 !
;  854          2331  1 !       NONE
;  855          2332  1 !
;  856          2333  1 ! SIDE EFFECTS:
;  857          2334  1 !
;  858          2335  1 !       NONE
;  859          2336  1 !
;  860          2337  1 !--
;  861          2338  1
;  862          2339  2     BEGIN
;  863          2340  2
;  864          2341  2     MAP
;  865          2342  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  866          2343  2
;  867          2344  2     RETURN (PC_TO_LINE_NO (.FMP, .FMP [BSF$A_MARK]));
;  868          2345  1     END;                                            ! of BAS$$LINE



                        0000 00000         .ENTRY  BAS$$LINE, Save nothing                 ; 2296
                50  04  AC  D0 00002       MOVL    FMP, R0                                 ; 2344
```

```
                     FC   A0  DD 00006          PUSHL   -4(R0)                                ;
                          50  DD 00009          PUSHL   R0                                   ;
                AB   AF        02  FB 0000B     CALLS   #2, PC_TO_LINE_NO                    ;
                                   04 0000F     RET                                          ; 2345

; Routine Size:  16 bytes,    Routine Base:  _BAS$CODE + 02A8
```

```
;  870          2346  1 ROUTINE BAS$$FUNCTION (                          ! Get current function name
;  871          2347  1         FMP                                      ! Current frame
;  872          2348  1     ) =
;  873          2349  1
;  874          2350  1 !++
;  875          2351  1 ! FUNCTIONAL DESCRIPTION:
;  876          2352  1 !
;  877          2353  1 !       Get the name of the function now in execution.  It is
;  878          2354  1 !       obtained from the BSF$A_PROC_ID field of the frame.
;  879          2355  1 !       The format depends on the frame type.  For the types
;  880          2356  1 !       which have no name, the "name" is returned as a line
;  881          2357  1 !       number, and the caller expects this.  In other cases,
;  882          2358  1 !       the name is returned as a pointer to a counted string.
;  883          2359  1 !
;  884          2360  1 ! FORMAL PARAMETERS:
;  885          2361  1 !
;  886          2362  1 !       FMP.ra.v        Address of the frame from which we want the
;  887          2363  1 !                       function name.
;  888          2364  1 !
;  889          2365  1 ! IMPLICIT INPUTS:
;  890          2366  1 !
;  891          2367  1 !       NONE
;  892          2368  1 !
;  893          2369  1 ! IMPLICIT OUTPUTS:
;  894          2370  1 !
;  895          2371  1 !       NONE
;  896          2372  1 !
;  897          2373  1 ! ROUTINE VALUE:
;  898          2374  1 !
;  899          2375  1 !       The function name, as a pointer to a counted string for
;  900          2376  1 !       main procedures, subprograms, external functions, DEFs
;  901          2377  1 !       and DEF*s.  The name is returned as a 32-bit line number
;  902          2378  1 !       for GOSUBs and condition handlers.
;  903          2379  1 !
;  904          2380  1 ! COMPLETION CODES:
;  905          2381  1 !
;  906          2382  1 !       NONE
;  907          2383  1 !
;  908          2384  1 ! SIDE EFFECTS:
;  909          2385  1 !
;  910          2386  1 !       NONE
;  911          2387  1 !
;  912          2388  1 !--
;  913          2389  1
;  914          2390  2     BEGIN
;  915          2391  2
;  916          2392  2     MAP
;  917          2393  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  918          2394  2
;  919          2395  2     RETURN (CASE .FMP [BSF$B_PROC_CODE] FROM BSF$K_PROC_MAIN TO BSF$K_PROC_IOL OF
;  920          2396  3             SET
;  921          2397  3             [BSF$K_PROC_MAIN, BSF$K_PROC_SUB, BSF$K_PROC_EXTF, BSF$K_PROC_DEF, BSF$K_PROC_DEFS] :        !
;  922          2398  3                 .FMP [BSF$A_PROC_ID];
;  923          2399  3             [BSF$K_PROC_GOSB, BSF$K_PROC_ONER, BSF$K_PROC_IOL] :         !
;  924          2400  3                 PC_TO_LINE_NO (.FMP, .FMP [BSF$A_PROC_ID]);
;  925          2401  3             [OUTRANGE] : 0;
;  926          2402  2             TES);
```

```
; 927          2403  1      END;                                        ! of BAS$$FUNCTION



                                             0000 00000 BAS$$FUNCTION:
                                                              .WORD    Save nothing                        ; 2346
                                 50      04   AC   D0 00002   MOVL     FMP, R0                             ; 2395
                       07        01      E5   A0   8F 00006   CASEB    -27(R0), #1, #7                     ;
    0013               0013      0013        0013     0000B 1$: .WORD  2$-1$,-                             ;
    0018               0018      0018        0013     00013            2$-1$,-                             ;
                                                                       2$-1$,-                             ;
                                                                       2$-1$,-                             ;
                                                                       2$-1$,-                             ;
                                                                       3$-1$,-                             ;
                                                                       3$-1$,-                             ;
                                                                       3$-1$                               ;
                                             50   D4 0001B    CLRL     R0                                  ;
                                                  04 0001D    RET                                          ;
                                 50      E8   A0   D0 0001E 2$: MOVL   -24(R0), R0                         ; 2398
                                                  04 00022    RET                                          ;
                                         E8   A0   DD 00023 3$: PUSHL  -24(R0)                             ; 2400
                                                  50   DD 00026  PUSHL R0                                  ;
                       FF7D   CF              02   FB 00028   CALLS    #2, PC_TO_LINE_NO                   ;
                                                  04 0002D    RET                                          ; 2403

; Routine Size:  46 bytes,    Routine Base:  _BAS$CODE + 02B8
```

```
;  929          2404  1 GLOBAL ROUTINE BAS$$MODULE (                  ! Get current module name
;  930          2405  1         FMP                                   ! Current frame
;  931          2406  1     ) =
;  932          2407  1
;  933          2408  1 !++
;  934          2409  1 ! FUNCTIONAL DESCRIPTION:
;  935          2410  1 !
;  936          2411  1 !       Get the name of the module now in execution.  It is
;  937          2412  1 !       obtained from the BSF$A_PROC_ID field of the frame.
;  938          2413  1 !       It is returned as a pointer to a counted string.
;  939          2414  1 !
;  940          2415  1 ! FORMAL PARAMETERS:
;  941          2416  1 !
;  942          2417  1 !       FMP.ra.v        Address of the frame from which we want the
;  943          2418  1 !                       module name.
;  944          2419  1 !
;  945          2420  1 ! IMPLICIT INPUTS:
;  946          2421  1 !
;  947          2422  1 !       NONE
;  948          2423  1 !
;  949          2424  1 ! IMPLICIT OUTPUTS:
;  950          2425  1 !
;  951          2426  1 !       NONE
;  952          2427  1 !
;  953          2428  1 ! ROUTINE VALUE:
;  954          2429  1 !
;  955          2430  1 !       The module name, as a pointer to a counted string.
;  956          2431  1 !
;  957          2432  1 ! COMPLETION CODES:
;  958          2433  1 !
;  959          2434  1 !       NONE
;  960          2435  1 !
;  961          2436  1 ! SIDE EFFECTS:
;  962          2437  1 !
;  963          2438  1 !       NONE
;  964          2439  1 !
;  965          2440  1 !--
;  966          2441  1
;  967          2442  2     BEGIN
;  968          2443  2
;  969          2444  2     MAP
;  970          2445  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  971          2446  2
;  972          2447  2     LOCAL
;  973          2448  2         BSF$A_MAJOR_STG : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME);
;  974          2449  2
;  975          2450  2 !+
;  976          2451  2 ! Load the pointer to the major procedure's frame.
;  977          2452  2 !-
;  978          2453  2     BSF$A_MAJOR_STG = .FMP [BSF$A_BASE_R11];
;  979          2454  2 !+
;  980          2455  2 ! Its procedure information starts with the name of the module.
;  981          2456  2 !-
;  982          2457  2     RETURN (.BSF$A_MAJOR_STG [BSF$A_PROC_INFO]);
;  983          2458  1     END;                                          ! of BAS$$MODULE
```

```
                          0000 00000          .ENTRY  BAS$$MODULE, Save nothing                       ; 2404
                50    04  AC   D0 00002      MOVL    FMP, R0                                         ; 2453
                50    F4  A0   D0 00006      MOVL    -12(R0), BSF$A_MAJOR_STG                        ;
                50  00AB  C0   D0 0000A      MOVL    171(BSF$A_MAJOR_STG), R0                        ; 2457
                              04 0000F      RET                                                     ; 2458

; Routine Size:  16 bytes,    Routine Base:  _BAS$CODE + 02E6
```

```
;  985          2459  1 ROUTINE HANDLER_HANDLER (                       ! Handle unwind for BAS$HANDLER
;  986          2460  1           SIG,                                   ! signal args
;  987          2461  1           MECH,                                  ! mechanism args
;  988          2462  1           ENBL                                   !variables passed from BAS$HANDLER
;  989          2463  1       ) =
;  990          2464  1
;  991          2465  1 !++
;  992          2466  1 ! FUNCTIONAL DESCRIPTION:
;  993          2467  1 !
;  994          2468  1 !       Handle the unwind which is likely to be done to BAS$HANDLER
;  995          2469  1 !       by freeing its heap storage.  The length and address of the
;  996          2470  1 !       heap storage are passed in the third argument to this routine.
;  997          2471  1 !
;  998          2472  1 ! FORMAL PARAMETERS:
;  999          2473  1 !
; 1000          2474  1 !       SIG.rl.a        Address of the signal vector.  This contains
; 1001          2475  1 !                       the condition.
; 1002          2476  1 !       MECH.rl.a       Address of the mechanism vector.  This contains
; 1003          2477  1 !                       the status of the frame that signalled.
; 1004          2478  1 !       ENBL.rl.a       Address of the enable vector.  This contains
; 1005          2479  1 !                       pointers to the variables used to free the
; 1006          2480  1 !                       heap storage for BAS$HANDLER.
; 1007          2481  1 !
; 1008          2482  1 ! IMPLICIT INPUTS:
; 1009          2483  1 !
; 1010          2484  1 !       NONE
; 1011          2485  1 !
; 1012          2486  1 ! IMPLICIT OUTPUTS:
; 1013          2487  1 !
; 1014          2488  1 !       NONE
; 1015          2489  1 !
; 1016          2490  1 ! ROUTINE VALUE:
; 1017          2491  1 !
; 1018          2492  1 !       NONE
; 1019          2493  1 !
; 1020          2494  1 ! COMPLETION CODES:
; 1021          2495  1 !
; 1022          2496  1 !       Always SS$_RESIGNAL, but this is ingored when we are
; 1023          2497  1 !       unwinding.
; 1024          2498  1 !
; 1025          2499  1 ! SIDE EFFECTS:
; 1026          2500  1 !
; 1027          2501  1 !       May call LIB$FREE_VM to return storage to the free pool.
; 1028          2502  1 !
; 1029          2503  1 !--
; 1030          2504  1
; 1031          2505  2     BEGIN
; 1032          2506  2
; 1033          2507  2     MAP
; 1034          2508  2         SIG : REF VECTOR,                         ! signal vector
; 1035          2509  2         MECH : REF VECTOR,                        ! mechanism vector
; 1036          2510  2         ENBL : REF VECTOR;                        ! enable vector
; 1037          2511  2
; 1038          2512  2 !+
; 1039          2513  2 ! First check for the unwinding condition.  If it is not, resignal.
; 1040          2514  2 !-
; 1041          2515  2
```

```
2      IF ( NOT (LIB$MATCH_COND (SIG [1], %REF (SS$_UNWIND)))) THEN RETURN (SS$_RESIGNAL);
2
2   !+
2   ! We are unwinding.  If any heap storage has been allocated, free it.
2   !-
2
2      IF (.(.ENBL [2]) NEQA 0) THEN LIB$FREE_VM (%REF (.(.ENBL [1])*%UPVAL), .ENBL [2]);
2
2   !+
2   ! All done.
2   !-
2      RETURN (SS$_RESIGNAL);
1      END;                                            ! of HANDLER_HANDLER


                                   0000 00000 HANDLER_HANDLER:
                                                      .WORD   Save nothing                       ; 2459
                 7E      0920   8F   3C 00002          MOVZWL  #2336, -(SP)                       ; 2516
                                     5E   DD 00007     PUSHL   SP                                 ;
        7E       04      AC     04   C1 00009          ADDL3   #4, SIG, -(SP)                     ;
           00000000G     00     02   FB 0000E          CALLS   #2, LIB$MATCH_COND                 ;
                         1C     50   E9 00015          BLBC    R0, 1$                             ;
                         50  0C AC   D0 00018          MOVL    ENBL, R0                           ; 2522
                             08 B0   D5 0001C          TSTL    @8(R0)                             ;
                                13   13 0001F          BEQL    1$                                 ;
                             08 A0   DD 00021          PUSHL   8(R0)                              ;
   04    AE      04      B0     02   78 00024          ASHL    #2, @4(R0), 4(SP)                  ;
                             04 AE   9F 0002A          PUSHAB  4(SP)                              ;
           00000000G     00     02   FB 0002D          CALLS   #2, LIB$FREE_VM                    ;
                 50      0918   8F   3C 00034 1$:      MOVZWL  #2328, R0                          ; 2527
                                     04 00039          RET                                        ; 2528

; Routine Size:  58 bytes,    Routine Base:  _BAS$CODE + 02F6
```

```
;  1056          2529  1 ROUTINE RESTART : RESTART_LINK NOVALUE =
;  1057          2530  1
;  1058          2531  1 !++
;  1059          2532  1 ! FUNCTIONAL DESCRIPTION:
;  1060          2533  1 !
;  1061          2534  1 !       This is a short routine which restores SP and branches to the
;  1062          2535  1 !       user's code.  It has to be a routine so its address can be
;  1063          2536  1 !       passed to SYS$UNWIND.
;  1064          2537  1 !       Before branching to the user's code it POPs any I/O that may
;  1065          2538  1 !       be in progress.
;  1066          2539  1 !
;  1067          2540  1 ! FORMAL PARAMETERS:
;  1068          2541  1 !
;  1069          2542  1 !       NONE
;  1070          2543  1 !
;  1071          2544  1 ! IMPLICIT INPUTS:
;  1072          2545  1 !
;  1073          2546  1 !       BAS$A_RESTART.ra        The PC to branch to, which will be the
;  1074          2547  1 !                               the first instruction of a line.
;  1075          2548  1 !
;  1076          2549  1 ! IMPLICIT OUTPUTS:
;  1077          2550  1 !
;  1078          2551  1 !       NONE
;  1079          2552  1 !
;  1080          2553  1 ! ROUTINE VALUE:
;  1081          2554  1 !
;  1082          2555  1 !       NONE
;  1083          2556  1 !
;  1084          2557  1 ! COMPLETION CODES:
;  1085          2558  1 !
;  1086          2559  1 !       NONE
;  1087          2560  1 !
;  1088          2561  1 ! SIDE EFFECTS:
;  1089          2562  1 !
;  1090          2563  1 !       Never returns to its 'caller'
;  1091          2564  1 !
;  1092          2565  1 !--
;  1093          2566  1
;  1094          2567  2     BEGIN
;  1095          2568  2
;  1096          2569  2     REGISTER
;  1097          2570  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  1098          2571  2
;  1099          2572  2     BUILTIN
;  1100          2573  2         FP,
;  1101          2574  2         SP;
;  1102          2575  2
;  1103          2576  2     FMP = .FP;
;  1104          2577  2     BAS$$UNWIND_IO (.FMP);                  ! POP this frame's I/O
;  1105          2578  2     SP = .FMP [BSF$A_BASE_SP];              ! Restore SP
;  1106          2579  2     SP = .SP - %UPVAL;
;  1107          2580  2     .SP = .BAS$A_RESTART;                   ! Specify place to go
;  1108          2581  2     RETURN;                                 ! Go there, in effect.
;  1109          2582  1     END;                                    ! end of RESTART
```

```
                         52              5D  D0 00000 RESTART:MOVL    FP, FMP                                 ; 2576
                                         52  DD 00003         PUSHL   FMP                                     ; 2577
                00000000G  00            01  FB 00005         CALLS   #1, BAS$$UNWIND_IO                      ;
                         5E        F8     A2  D0 0000C         MOVL    -8(FMP), SP                             ; 2578
                         5E              04  C2 00010         SUBL2   #4, SP                                  ; 2579
                         6E 00000000'    EF  D0 00013         MOVL    BAS$A_RESTART, (SP)                     ; 2580
                                             05 0001A         RSB                                             ; 2582

; Routine Size:  27 bytes,    Routine Base:  _BAS$CODE + 0330
```

```
;  1111          2583  1 ROUTINE RESTART_IO : RESTART_LINK NOVALUE =
;  1112          2584  1
;  1113          2585  1 !++
;  1114          2586  1 ! FUNCTIONAL DESCRIPTION:
;  1115          2587  1 !
;  1116          2588  1 !       This is a short routine to call BAS$$RESTART_IO when unwinding
;  1117          2589  1 !       to the beginning of an I/O list.  No I/O popping is done.
;  1118          2590  1 !
;  1119          2591  1 ! FORMAL PARAMETERS:
;  1120          2592  1 !
;  1121          2593  1 !       NONE
;  1122          2594  1 !
;  1123          2595  1 ! IMPLICIT INPUTS:
;  1124          2596  1 !
;  1125          2597  1 !       Gets the PC to branch to from BAS$$RESTART_IO.
;  1126          2598  1 !
;  1127          2599  1 ! IMPLICIT OUTPUTS:
;  1128          2600  1 !
;  1129          2601  1 !       NONE
;  1130          2602  1 !
;  1131          2603  1 ! ROUTINE VALUE:
;  1132          2604  1 !
;  1133          2605  1 !       NONE
;  1134          2606  1 !
;  1135          2607  1 ! COMPLETION CODES:
;  1136          2608  1 !
;  1137          2609  1 !       NONE
;  1138          2610  1 !
;  1139          2611  1 ! SIDE EFFECTS:
;  1140          2612  1 !
;  1141          2613  1 !       Never returns to its 'caller'
;  1142          2614  1 !
;  1143          2615  1 !--
;  1144          2616  1
;  1145          2617  2     BEGIN
;  1146          2618  2
;  1147          2619  2     REGISTER
;  1148          2620  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  1149          2621  2
;  1150          2622  2     BUILTIN
;  1151          2623  2         FP,
;  1152          2624  2         SP;
;  1153          2625  2
;  1154          2626  2     FMP = .FP;
;  1155          2627  2     SP = .FMP [BSF$A_BASE_SP];                  ! Restore SP
;  1156          2628  2     SP = .SP - %UPVAL;
;  1157          2629  2     .SP = BAS$$RESTART_IO ();                   ! Get place to go
;  1158          2630  2     RETURN;                                     ! Go there, in effect.
;  1159          2631  1     END;                                        ! end of RESTART_IO


                                50            5D  D0 00000 RESTART_IO:
                                                           MOVL    FP, FMP                 ; 2626
                                5E        F8  A0  D0 00003           MOVL    -8(FMP), SP             ; 2627
```

```
                 5E                04  C2 00007        SUBL2   #4, SP                                 ; 2628
00000000G        00                00  FB 0000A        CALLS   #0, BAS$$RESTART_IO                    ; 2629
                 6E                50  D0 00011        MOVL    R0, (SP)                               ;
                                       05 00014        RSB                                            ; 2631

; Routine Size:  21 bytes,    Routine Base:  _BAS$CODE + 034B
```

```
; 1161          2632  1 ROUTINE BAS$$USER_HAND (                      ! interface to user's condition handler
; 1162          2633  1         ERR_CODE,                             ! BASIC error code
; 1163          2634  1         FMP,                                  ! user's frame
; 1164          2635  1         LEVEL                                 ! level to unwind
; 1165          2636  1     ) =
; 1166          2637  1
; 1167          2638  1 !++
; 1168          2639  1 ! FUNCTIONAL DESCRIPTION:
; 1169          2640  1 !
; 1170          2641  1 !       Try to pass a SIGNALed condition to the BASIC user's program
; 1171          2642  1 !       for processing.
; 1172          2643  1 !
; 1173          2644  1 ! FORMAL PARAMETERS:
; 1174          2645  1 !
; 1175          2646  1 !       ERR_CODE.rl.v   The BASIC error code that is being signaled.
; 1176          2647  1 !       FMP.rl.v        Pointer to the frame of the BASIC program
; 1177          2648  1 !       LEVEL.rl.v      Number of levels to unwind to get to the
; 1178          2649  1 !                       current frame of the BASIC program.
; 1179          2650  1 !
; 1180          2651  1 ! IMPLICIT INPUTS:
; 1181          2652  1 !
; 1182          2653  1 !       BAS$L_ERRFLG    0 if no error in progress, 1 if an error is
; 1183          2654  1 !                       in progress.
; 1184          2655  1 !
; 1185          2656  1 ! IMPLICIT OUTPUTS:
; 1186          2657  1 !
; 1187          2658  1 !       BAS$L_ERRFLG    Set to 1 while we are doing error processing.
; 1188          2659  1 !       BAS$L_ERL       The line number being executed when the error
; 1189          2660  1 !                       occurred.
; 1190          2661  1 !       BAS$T_ERN       The name of the module in which the error
; 1191          2662  1 !                       occurred.
; 1192          2663  1 !       BAS$L_ERR       The BASIC error number now being processed.
; 1193          2664  1 !       HIGHEST_LEVEL   If ON ERROR GO BACK, the level to UNWIND to
; 1194          2665  1 !                        if a lower level does a RESUME with no line
; 1195          2666  1 !                        number.
; 1196          2667  1 !       HIGHEST_FMP     If ON ERROR GO BACK, the frame to UNWIND to
; 1197          2668  1 !                        if a lower level does a RESUME with no line
; 1198          2669  1 !                        number.
; 1199          2670  1 !       ACCUM_LEVEL     If ON ERROR GO BACK, the number of levels above
; 1200          2671  1 !                        which must be unwound through if a lower
; 1201          2672  1 !                        level does a RESUME with a line number.
; 1202          2673  1 !       UNWIND_COUNT    If non-zero, the number of levels to UNWIND
; 1203          2674  1 !                        when we get back to the top level call of
; 1204          2675  1 !                        BAS$HANDLER.
; 1205          2676  1 !
; 1206          2677  1 ! ROUTINE VALUE:
; 1207          2678  1 !
; 1208          2679  1 !       USER_HAND_CONT (=0) => The user has processed the error condition,
; 1209          2680  1 !       continue from the point of error (or from an unwind).  If an
; 1210          2681  1 !       unwind is needed, UNWIND_COUNT is set for the highest level
; 1211          2682  1 !       handler.
; 1212          2683  1 !
; 1213          2684  1 !       USER_HAND_BACK (=1) => The user is not prepared to handle
; 1214          2685  1 !       the error at this level, but he may be able to handle it at a
; 1215          2686  1 !       deeper level.  Revert.
; 1216          2687  1 !
; 1217          2688  1 !       USER_HAND_FAIL (=2) => The user demands system processing of
```

```
;  1218          2689  1 !       this error; do not test deeper levels.  Revert and do not call
;  1219          2690  1 !       BAS$$USER_HAND again for this error.
;  1220          2691  1 !
;  1221          2692  1 ! COMPLETION CODES:
;  1222          2693  1 !
;  1223          2694  1 !       NONE
;  1224          2695  1 !
;  1225          2696  1 ! SIDE EFFECTS:
;  1226          2697  1 !
;  1227          2698  1 !       May call user code.
;  1228          2699  1 !
;  1229          2700  1 !--
;  1230          2701  1
;  1231          2702  2     BEGIN
;  1232          2703  2
;  1233          2704  2     LITERAL
;  1234          2705  2         K_MAJOR = 1,                            !A major frame
;  1235          2706  2         K_MINOR = 0;                            !A minor frame
;  1236          2707  2
;  1237          2708  2     MAP
;  1238          2709  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  1239          2710  2
;  1240          2711  2     LOCAL
;  1241          2712  2         MOD_NAME_ADDR,
;  1242          2713  2         NON_GOSUB_FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD),
;  1243          2714  2         USER_HAND_VAL,
;  1244          2715  2         ONER_RESULT,
;  1245          2716  2         BSF$A_MAJOR_STG : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME),
;  1246          2717  2         BSF$A_MINOR_STG : REF BLOCK [0, BYTE] FIELD (BSF$MINOR_FRAME),
;  1247          2718  2         NEXT_FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD),
;  1248          2719  2         THIS_FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD),
;  1249          2720  2         MAJOR_OR_MINOR,
;  1250          2721  2         SEARCH_DONE;
;  1251          2722  2
;  1252          2723  2     BUILTIN
;  1253          2724  2         FP;
;  1254          2725  2
;  1255          2726  2 !+
;  1256          2727  2 ! If the severity is "error" or "warning", let the compiled code
;  1257          2728  2 !  intercept the error, if it has requested to do so.  Note that
;  1258          2729  2 !  we must check the severity table rather than the condition value
;  1259          2730  2 !  since LIB$STOP forces the severity to "severe error".
;  1260          2731  2 !-
;  1261          2732  2
;  1262          2733  4     IF ((.ERR_SEVERITY [.ERR_CODE] NEQ STS$K_ERROR)     !
;  1263          2734  3         AND (.ERR_SEVERITY [.ERR_CODE] NEQ STS$K_WARNING))
;  1264          2735  2     THEN
;  1265          2736  2         RETURN (USER_HAND_FAIL);
;  1266          2737  2
;  1267          2738  3     IF (.BAS$L_ERRFLG NEQ 0)
;  1268          2739  2     THEN
;  1269          2740  2 !+
;  1270          2741  2 ! The user has committed an error or said ON ERROR GOTO 0 durring
;  1271          2742  2 !  error processing.  Demand system processing.
;  1272          2743  2 !-
;  1273          2744  2         RETURN (USER_HAND_FAIL);
;  1274          2745  2
```

```
;  1275          2746  3     IF (.BAS$L_GOING_BACK NEQ 0)
;  1276          2747  3     THEN
;  1277          2748  !+
;  1278          2749  ! We are one BASIC level deeper in ON ERROR GO BACK processing
;  1279          2750  !-
;  1280          2751          BEGIN
;  1281          2752          BAS$L_ERRFLG = 1;
;  1282          2753          ACCUM_LEVEL = .ACCUM_LEVEL + .LEVEL;
;  1283          2754          END
;  1284          2755     ELSE
;  1285          2756  !+
;  1286          2757  ! This is the first time we have seen this error.  Set things up.
;  1287          2758  !-
;  1288          2759          BEGIN
;  1289          2760
;  1290          2761          !+
;  1291          2762          ! only set 'going_back' if this is not a restartable error, i.e.,
;  1292          2763          ! not err=50 (data format error) and not err=52 (illegal number).
;  1293          2764          ! This statement may need further conditionalization if more
;  1294          2765          ! restartable errors are added.
;  1295          2766          !-
;  1296          2767          BAS$L_GOING_BACK = ( IF (( .ERR_CODE EQL 50 ) OR ( .ERR_CODE EQL 52 ))
;  1297          2768                               THEN 0
;  1298          2769                               ELSE 1 );
;  1299          2770
;  1300          2771          BAS$L_ERRFLG = 1;                       ! error in progress
;  1301          2772          BAS$A_CH_CUR_LN = .FMP [BSF$A_MARK];    ! remember default restart PC
;  1302          2773          BAS$L_ERL = BAS$$LINE (.FMP);           ! compute default restart line number
;  1303          2774          MOD_NAME_ADDR = BAS$$MODULE (.FMP);
;  1304          2775          BAS$T_ERN [DSC$A_POINTER] = .MOD_NAME_ADDR + 1;
;  1305          2776          BAS$T_ERN [DSC$W_LENGTH] = .BLOCK [.MOD_NAME_ADDR, 0, 0, 8, 0; 1, BYTE];
;  1306          2777          BAS$T_ERN [DSC$B_CLASS] = DSC$K_CLASS_S;
;  1307          2778          BAS$T_ERN [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  1308          2779          BAS$L_ERR = .ERR_CODE;
;  1309          2780          HIGHEST_LEVEL = .LEVEL;
;  1310          2781          HIGHEST_FMP = .FMP;
;  1311          2782          ACCUM_LEVEL = .LEVEL;
;  1312          2783          END;
;  1313          2784
;  1314          2785  !+
;  1315          2786  ! Fetch the current value of BSF$A_USER_HAND.
;  1316          2787  ! We must first dig back to the first non-GOSUB frame.
;  1317          2788  !-
;  1318          2789      NON_GOSUB_FMP = .FMP;
;  1319          2790      SEARCH_DONE = 0;
;  1320          2791
;  1321          2792      DO
;  1322          2793          BEGIN
;  1323          2794
;  1324          2795  4       IF (.NON_GOSUB_FMP [BSF$A_HANDLER] EQLA BAS$HANDLER)
;  1325          2796          THEN
;  1326          2797
;  1327          2798              IF (.NON_GOSUB_FMP [BSF$B_PROC_CODE] NEQ BSF$K_PROC_GOSB) THEN SEARCH_DONE = 1;
;  1328          2799
;  1329          2800  4       IF ( NOT .SEARCH_DONE)
;  1330          2801  3       THEN
;  1331          2802  4           BEGIN
```

```
;  1332          2803  4              NON_GOSUB_FMP = .NON_GOSUB_FMP [BSF$A_SAVED_FP];
;  1333          2804  4
;  1334          2805  5              IF (.NON_GOSUB_FMP EQLA 0)
;  1335          2806  4              THEN
;  1336          2807  4          !+
;  1337          2808  4          ! We have been unable to find a non-GOSUB frame.  This is quite
;  1338          2809  4          ! unreasonable.  Force system handling on this error.
;  1339          2810  4          !-
;  1340          2811  4                  RETURN (USER_HAND_FAIL);
;  1341          2812  4
;  1342          2813  4              END;
;  1343          2814  3
;  1344          2815  3          END
;  1345          2816  2      UNTIL (.SEARCH_DONE);
;  1346          2817  2
;  1347          2818  2      CASE .NON_GOSUB_FMP [BSF$B_PROC_CODE] FROM BSF$K_PROC_MAIN TO BSF$K_PROC_IOL OF
;  1348          2819  2          SET
;  1349          2820  2
;  1350          2821  2          [BSF$K_PROC_MAIN, BSF$K_PROC_SUB, BSF$K_PROC_EXTF, BSF$K_PROC_DEFS] :
;  1351          2822  3              BEGIN
;  1352          2823  3              BSF$A_MAJOR_STG = .NON_GOSUB_FMP [BSF$A_BASE_R11];
;  1353          2824  3              USER_HAND_VAL = .BSF$A_MAJOR_STG [BSF$A_USER_HAND];
;  1354          2825  3              MAJOR_OR_MINOR = K_MAJOR;
;  1355          2826  3              END;
;  1356          2827  2
;  1357          2828  2          [BSF$K_PROC_DEF] :
;  1358          2829  3              BEGIN
;  1359          2830  3              BSF$A_MINOR_STG = .NON_GOSUB_FMP [BSF$A_BASE_R10];
;  1360          2831  3              USER_HAND_VAL = .BSF$A_MINOR_STG [BSF$A_USER_HAND];
;  1361          2832  3              MAJOR_OR_MINOR = K_MINOR;
;  1362          2833  3              END;
;  1363          2834  2
;  1364          2835  2          [BSF$K_PROC_IOL] :
;  1365          2836  3              BEGIN                               ! Don't let error handling go beyond the immediate code
;  1366          2837  3              USER_HAND_VAL = 0;
;  1367          2838  3              END;
;  1368          2839  2
;  1369          2840  2          [BSF$K_PROC_GOSB, BSF$K_PROC_ONER, OUTRANGE] :
;  1370          2841  2              USER_HAND_VAL = 0;                  ! this should never happen
;  1371          2842  2          TES;
;  1372          2843  2
;  1373          2844  3      IF (.USER_HAND_VAL EQL 0)
;  1374          2845  2      THEN
;  1375          2846  2      !+
;  1376          2847  2      ! The user has specified (or defaulted to) system error handling
;  1377          2848  2      !  for any errors.  Revert and don't call BAS$$USER_HAND again.
;  1378          2849  2      !-
;  1379          2850  2          RETURN (USER_HAND_FAIL);
;  1380          2851  2
;  1381          2852  3      IF (.USER_HAND_VAL EQL 1)
;  1382          2853  2      THEN
;  1383          2854  3          BEGIN
;  1384          2855  3      !+
;  1385          2856  3      ! The user has specified ON ERROR GO BACK.  Revert but do call
;  1386          2857  3      !  BAS$$USER_HAND again.  Note that GOSUBs get unwound one at a
;  1387          2858  3      !  time by this mechanism, but we will make the same decision
;  1388          2859  3      !  each time because the frame is marked for an immediate
```

```
;  1389          2860  3 !  ON ERROR GO BACK.
;  1390          2861  3 !-
;  1391          2862  3         BAS$L_ERRFLG = 0;
;  1392          2863  3         RETURN (USER_HAND_BACK);
;  1393          2864  3         END;
;  1394          2865  3
;  1395          2866  3 !+
;  1396          2867  3 ! The user has specified an entry point in this frame for error
;  1397          2868  3 ! processing.  We call his code as a modified GOSUB.
;  1398          2869  3 ! Further system processing depends on how the user's code terminates.
;  1399          2870  3 !-
;  1400          2871  3     ONER_RESULT = BAS$INIT_ONERR (.NON_GOSUB_FMP, .USER_HAND_VAL);
;  1401          2872  3
;  1402          2873  3     SELECTONEU (.ONER_RESULT) OF
;  1403          2874  3         SET
;  1404          2875  3
;  1405          2876  3         [USER_ERR_RSUMZ] :
;  1406          2877  3 !+
;  1407          2878  3 ! The condition handler ended with a RESUME with no line number.
;  1408          2879  3 ! Unwind to the frame in which the signal happened and restart
;  1409          2880  3 ! the statement.
;  1410          2881  3 !-
;  1411          2882  3             BEGIN
;  1412          2883  3             BAS$L_ERRFLG = 0;
;  1413          2884  3             BAS$A_RESTART = .BAS$A_CH_CUR_LN;
;  1414          2885  3             BAS$A_CH_CUR_LN = 0;
;  1415          2886  3             BAS$L_GOING_BACK = 0;
;  1416          2887  3             UNWIND_COUNT = .HIGHEST_LEVEL;
;  1417          2888  3 !+
;  1418          2889  3 ! The compiler should not permit a module to exist with RESUME with no
;  1419          2890  3 ! line number and /NOLINE, but check for that case here and give an
;  1420          2891  3 ! error message.
;  1421          2892  3 !-
;  1422          2893  3
;  1423          2894  3             IF (.BAS$A_RESTART EQLA 0) THEN BAS$$STOP (BAS$K_PROLOSSOR);
;  1424          2895  3
;  1425          2896  4             IF (.HIGHEST_LEVEL EQL 0)
;  1426          2897  3             THEN
;  1427          2898  4                 BEGIN
;  1428          2899  4 !+
;  1429          2900  4 ! Rather than doing an unwind to 0, search through the frames and patch
;  1430          2901  4 ! the return PC.
;  1431          2902  4 !-
;  1432          2903  4                 THIS_FMP = .FP;
;  1433          2904  4
;  1434          2905  4                 DO
;  1435          2906  5                     BEGIN
;  1436          2907  5                     NEXT_FMP = .THIS_FMP;
;  1437          2908  5                     THIS_FMP = .THIS_FMP [BSF$A_SAVED_FP];
;  1438          2909  5                     END
;  1439          2910  4                 UNTIL (.THIS_FMP EQLA .HIGHEST_FMP);
;  1440          2911  4
;  1441          2912  4                 NEXT_FMP [BSF$A_SAVED_PC] = RESTART;
;  1442          2913  3                 END;
;  1443          2914  3
;  1444          2915  3             RETURN (USER_HAND_CONT);
;  1445          2916  2             END;
```

```
;  1446          2917  2
;  1447          2918  2          [USER_ERR_GOBK] :
;  1448          2919  2      !+
;  1449          2920  2      ! The condition handler ended with ON ERROR GO BACK.  Revert but
;  1450          2921  2      ! continue to call BAS$$USER_HAND.  However, this frame is marked for
;  1451          2922  2      ! an immediate ON ERROR GO BACK in case we are in a GOSUB: we don't
;  1452          2923  2      ! want to call the user's error handler again.
;  1453          2924  2      !-
;  1454          2925  2                  BEGIN
;  1455          2926  2                  BAS$L_ERRFLG = 0;
;  1456          2927  2
;  1457          2928  2                  CASE .MAJOR_OR_MINOR FROM K_MINOR TO K_MAJOR OF
;  1458          2929  2                      SET
;  1459          2930  2
;  1460          2931  2                      [K_MINOR] :
;  1461          2932  2                          BSF$A_MINOR_STG [BSF$A_USER_HAND] = 1;
;  1462          2933  2
;  1463          2934  2                      [K_MAJOR] :
;  1464          2935  2                          BSF$A_MAJOR_STG [BSF$A_USER_HAND] = 1;
;  1465          2936  2                      TES;
;  1466          2937  2
;  1467          2938  2                  RETURN (USER_HAND_BACK);
;  1468          2939  2                  END;
;  1469          2940  2
;  1470          2941  2          [USER_ERR_OEGZ] :
;  1471          2942  2      !+
;  1472          2943  2      ! The condition handler ended with ON ERROR GOTO 0.  Revert but
;  1473          2944  2      ! force system handling for this error.
;  1474          2945  2      !-
;  1475          2946  3                  BEGIN
;  1476          2947  3
;  1477          2948  3                  CASE .MAJOR_OR_MINOR FROM K_MINOR TO K_MAJOR OF
;  1478          2949  3                      SET
;  1479          2950  3
;  1480          2951  3                      [K_MINOR] :
;  1481          2952  3                          BSF$A_MINOR_STG [BSF$A_USER_HAND] = 0;
;  1482          2953  3
;  1483          2954  3                      [K_MAJOR] :
;  1484          2955  3                          BSF$A_MAJOR_STG [BSF$A_USER_HAND] = 0;
;  1485          2956  3                      TES;
;  1486          2957  3
;  1487          2958  3                  RETURN (USER_HAND_FAIL);
;  1488          2959  3                  END;
;  1489          2960  2
;  1490          2961  2              [OTHERWISE] :
;  1491          2962  2      !+
;  1492          2963  2      ! The condition handler ended with a RESUME with a line number.
;  1493          2964  2      ! Unwind to the current frame and restart at the indicated PC.
;  1494          2965  2      !-
;  1495          2966  3                  BEGIN
;  1496          2967  3                  BAS$L_ERRFLG = 0;
;  1497          2968  3                  BAS$A_RESTART = .ONER_RESULT;
;  1498          2969  3                  BAS$A_CH_CUR_LN = 0;
;  1499          2970  3                  BAS$L_GOING_BACK = 0;
;  1500          2971  3                  UNWIND_COUNT = .ACCUM_LEVEL;
;  1501          2972  3
;  1502          2973  4                  IF (.LEVEL EQL 0)
```

```
                THEN
                    BEGIN
        !+
        ! Rather than doing an unwind to 0, search through the frames and
        ! patch the return PC.
        !-
                    THIS_FMP = .FP;

                    DO
                        BEGIN
                        NEXT_FMP = .THIS_FMP;
                        THIS_FMP = .THIS_FMP [BSF$A_SAVED_FP];
                        END
                    UNTIL (.THIS_FMP EQLA .FMP);

                    NEXT_FMP [BSF$A_SAVED_PC] = RESTART;

                    END;

                RETURN (USER_HAND_CONT);
                END
            TES;

        END;                                          ! of BAS$$USER_HAND
```

```
                         007C 00000 BAS$$USER_HAND:
                                              .WORD   Save R2,R3,R4,R5,R6                   ; 2632
      56       FC9A   CF   9E 00002           MOVAB   ERR_SEVERITY, R6                      ;
      55  00000000'   EF   9E 00007           MOVAB   BAS$L_ERRFLG, R5                      ;
      50                66   9E 0000E           MOVAB   ERR_SEVERITY, R0                      ;
      02          04 BC40   91 00011           CMPB    @ERR_CODE[R0], #2                     ; 2733
                        06   13 00016           BEQL    1$                                    ;
                  04 BC40   95 00018           TSTB    @ERR_CODE[R0]                         ; 2734
                        02   12 0001C           BNEQ    2$                                    ;
                        65   D5 0001E 1$:       TSTL    BAS$L_ERRFLG                          ; 2738
                        03   13 00020 2$:       BEQL    3$                                    ;
                      013C   31 00022           BRW     30$                                   ;
      65                01   D0 00025 3$:       MOVL    #1, BAS$L_ERRFLG                      ; 2752
                2C      A5   D5 00028           TSTL    BAS$L_GOING_BACK                      ; 2746
                        07   13 0002B           BEQL    4$                                    ;
0C    A5        0C      AC   C0 0002D           ADDL2   LEVEL, ACCUM_LEVEL                    ; 2753
                        54   11 00032           BRB     8$                                    ; 2746
      32        04      AC   D1 00034 4$:       CMPL    ERR_CODE, #50                         ; 2767
                        06   13 00038           BEQL    5$                                    ;
      34        04      AC   D1 0003A           CMPL    ERR_CODE, #52                         ;
                        04   12 0003E           BNEQ    6$                                    ;
                        50   D4 00040 5$:       CLRL    R0                                    ;
                        03   11 00042           BRB     7$                                    ;
      50                01   D0 00044 6$:       MOVL    #1, R0                                ;
2C    A5                50   D0 00047 7$:       MOVL    R0, BAS$L_GOING_BACK                  ;
      52        08      AC   D0 0004B           MOVL    FMP, R2                               ; 2772
28    A5        FC      A2   D0 0004F           MOVL    -4(R2), BAS$A_CH_CUR_LN               ;
                        52   DD 00054           PUSHL   R2                                    ; 2773
```

```
                FEED   CF              01  FB 00056          CALLS   #1, BAS$$LINE                                   ;
                  FC   A5              50  D0 0005B          MOVL    R0, BAS$L_ERL                                   ;
                                       52  DD 0005F          PUSHL   R2                                              ; 2774
                FF20   CF              01  FB 00061          CALLS   #1, BAS$$MODULE                                 ;
                  F4   A5          01  A0  9E 00066          MOVAB   1(R0), BAS$T_ERN+4                              ; 2775
                  F0   A5              60  9B 0006B          MOVZBW  (MOD_NAME_ADDR), BAS$T_ERN                      ; 2776
                  F2   A5        010E  8F  B0 0006F          MOVW    #270, BAS$T_ERN+2                               ; 2778
                  F8   A5          04  AC  D0 00075          MOVL    ERR_CODE, BAS$L_ERR                             ; 2779
                  04   A5          0C  AC  D0 0007A          MOVL    LEVEL, HIGHEST_LEVEL                            ; 2780
                  08   A5              52  D0 0007F          MOVL    R2, HIGHEST_FMP                                 ; 2781
                  0C   A5          0C  AC  D0 00083          MOVL    LEVEL, ACCUM_LEVEL                              ; 2782
                       50          08  AC  D0 00088 8$:      MOVL    FMP, NON_GOSUB_FMP                              ; 2789
                                       52  D4 0008C          CLRL    SEARCH_DONE                                     ; 2790
                       51  00000000G  00  9E 0008E 9$:      MOVAB   BAS$HANDLER, R1                                 ; 2795
                       51              60  D1 00095          CMPL    (NON_GOSUB_FMP), R1
                                       09  12 00098          BNEQ    10$
                       06          E5  A0  91 0009A          CMPB    -27(NON_GOSUB_FMP), #6                          ; 2798
                                       03  13 0009E          BEQL    10$
                       52              01  D0 000A0          MOVL    #1, SEARCH_DONE
                       09              52  E8 000A3 10$:      BLBS    SEARCH_DONE, 11$                                ; 2800
                       50          0C  A0  D0 000A6          MOVL    12(NON_GOSUB_FMP), NON_GOSUB_FMP                ; 2803
                                       37  13 000AA          BEQL    17$                                             ; 2805
                       DF              52  E9 000AC          BLBC    SEARCH_DONE, 9$                                 ; 2816
                   07  01          E5  A0  8F 000AF 11$:      CASEB   -27(NON_GOSUB_FMP), #1, #7                      ; 2818
             001F  0012        0012  0012  000B4 12$:      .WORD   13$-12$,-
             002B  002B        002B  0012  000BC                  13$-12$,-
                                                                  13$-12$,-
                                                                  14$-12$,-
                                                                  13$-12$,-
                                                                  15$-12$,-
                                                                  15$-12$,-
                                                                  15$-12$,-
                                                                  15$-12$
                                       19  11 000C4          BRB     15$                                             ; 2841
                       53          F4  A0  D0 000C6 13$:      MOVL    -12(NON_GOSUB_FMP), BSF$A_MAJOR_STG             ; 2823
                       51          7F  A3  D0 000CA          MOVL    127(BSF$A_MAJOR_STG), USER_HAND_VAL             ; 2824
                       54              01  D0 000CE          MOVL    #1, MAJOR_OR_MINOR                              ; 2825
                                       0E  11 000D1          BRB     16$                                             ; 2818
                       52          F0  A0  D0 000D3 14$:      MOVL    -16(NON_GOSUB_FMP), BSF$A_MINOR_STG             ; 2830
                       51          7F  A2  D0 000D7          MOVL    127(BSF$A_MINOR_STG), USER_HAND_VAL             ; 2831
                                       54  D4 000DB          CLRL    MAJOR_OR_MINOR                                  ; 2832
                                       02  11 000DD          BRB     16$                                             ; 2818
                                       51  D4 000DF 15$:      CLRL    USER_HAND_VAL                                   ; 2837
                                       51  D5 000E1 16$:      TSTL    USER_HAND_VAL                                   ; 2844
                                       7C  13 000E3 17$:      BEQL    30$
                       01              51  D1 000E5          CMPL    USER_HAND_VAL, #1                               ; 2852
                                       04  12 000E8          BNEQ    18$
                                       65  D4 000EA          CLRL    BAS$L_ERRFLG                                    ; 2862
                                       5A  11 000EC          BRB     25$                                             ; 2863
                                       03  BB 000EE 18$:      PUSHR   #^M<R0,R1>                                      ; 2871
         00000000G     00              02  FB 000F0          CALLS   #2, BAS$INIT_ONERR
                                       50  D5 000F7          TSTL    ONER_RESULT                                     ; 2876
                                       34  12 000F9          BNEQ    21$
                                       65  D4 000FB          CLRL    BAS$L_ERRFLG                                    ; 2883
                   30  A5          28  A5  D0 000FD          MOVL    BAS$A_CH_CUR_LN, BAS$A_RESTART                  ; 2884
                                   28  A5  7C 00102          CLRQ    BAS$A_CH_CUR_LN                                 ; 2885
                   10  A5          04  A5  D0 00105          MOVL    HIGHEST_LEVEL, UNWIND_COUNT                     ; 2887
                                   30  A5  D5 0010A          TSTL    BAS$A_RESTART                                   ; 2894
```

```
                             09  12 0010D          BNEQ    19$                                          ;
                 7E     00G  8F  9A 0010F          MOVZBL  #BAS$K_PROLOSSOR, -(SP)                      ;
         FD9C    CF          01  FB 00113          CALLS   #1, BAS$$STOP                                ;
                        04   A5  D5 00118  19$:    TSTL    HIGHEST_LEVEL                                ; 2896
                             71  12 0011B          BNEQ    34$                                          ;
                 51          5D  D0 0011D          MOVL    FP, THIS_FMP                                 ; 2903
                 50          51  D0 00120  20$:    MOVL    THIS_FMP, NEXT_FMP                           ; 2907
                 51     0C   A1  D0 00123          MOVL    12(THIS_FMP), THIS_FMP                       ; 2908
         08      A5          51  D1 00127          CMPL    THIS_FMP, HIGHEST_FMP                        ; 2910
                             F3  12 0012B          BNEQ    20$                                          ;
                             59  11 0012D          BRB     33$                                          ; 2912
         01                  50  D1 0012F  21$:    CMPL    ONER_RESULT, #1                              ; 2918
                             18  12 00132          BNEQ    26$                                          ;
                             65  D4 00134          CLRL    BAS$L_ERRFLG                                 ; 2926
  01             00          54  CF 00136          CASEL   MAJOR_OR_MINOR, #0, #1                       ; 2928
               000A        0004     0013A  22$:    .WORD   23$-22$,-                                    ;
                                                           24$-22$                                      ;
         7F      A2          01  D0 0013E  23$:    MOVL    #1, 127(BSF$A_MINOR_STG)                     ; 2932
                             04  11 00142          BRB     25$                                          ;
         7F      A3          01  D0 00144  24$:    MOVL    #1, 127(BSF$A_MAJOR_STG)                     ; 2935
                 50          01  D0 00148  25$:    MOVL    #1, R0                                       ; 2938
                                 04 0014B          RET                                                  ;
         02                  50  D1 0014C  26$:    CMPL    ONER_RESULT, #2                              ; 2941
                             14  12 0014F          BNEQ    31$                                          ;
  01             00          54  CF 00151          CASEL   MAJOR_OR_MINOR, #0, #1                       ; 2948
               0009        0004     00155  27$:    .WORD   28$-27$,-                                    ;
                                                           29$-27$                                      ;
                        7F   A2  D4 00159  28$:    CLRL    127(BSF$A_MINOR_STG)                         ; 2952
                             03  11 0015C          BRB     30$                                          ;
                        7F   A3  D4 0015E  29$:    CLRL    127(BSF$A_MAJOR_STG)                         ; 2955
                 50          02  D0 00161  30$:    MOVL    #2, R0                                       ; 2958
                                 04 00164          RET                                                  ;
                             65  D4 00165  31$:    CLRL    BAS$L_ERRFLG                                 ; 2967
         30      A5          50  D0 00167          MOVL    ONER_RESULT, BAS$A_RESTART                   ; 2968
                        28   A5  7C 0016B          CLRQ    BAS$A_CH_CUR_LN                              ; 2969
         10      A5     0C   A5  D0 0016E          MOVL    ACCUM_LEVEL, UNWIND_COUNT                    ; 2971
                        0C   AC  D5 00173          TSTL    LEVEL                                        ; 2973
                             16  12 00176          BNEQ    34$                                          ;
                 51          5D  D0 00178          MOVL    FP, THIS_FMP                                 ; 2980
                 50          51  D0 0017B  32$:    MOVL    THIS_FMP, NEXT_FMP                           ; 2984
                 51     0C   A1  D0 0017E          MOVL    12(THIS_FMP), THIS_FMP                       ; 2985
         08      AC          51  D1 00182          CMPL    THIS_FMP, FMP                                ; 2987
                             F3  12 00186          BNEQ    32$                                          ;
         10      A0   FE44   CF  9E 00188  33$:    MOVAB   RESTART, 16(NEXT_FMP)                        ; 2989
                             50  D4 0018E  34$:    CLRL    R0                                           ; 2993
                                 04 00190          RET                                                  ; 2997

; Routine Size:  401 bytes,    Routine Base:  _BAS$CODE + 0360


; 1527         2998  1
```

```
1 GLOBAL ROUTINE BAS$RESUME (                            ! end of error handler
1         NEW_PC                                          ! where to restart
1     ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Resume execution from an error handler.  The compiled code
1 !       calls RESUME passing the address of the location at which
1 !       to continue execution.  We must be in an error handler.
1 !       The stack is cut back to the call to BAS$INIT_ONER which
1 !       is in BAS$$USER_HAND and the RET at the end of this routine
1 !       actually returns from BAS$INIT_ONER.  This is similar to
1 !       GOSUB processing.  To simplify the restoring of registers
1 !       BAS$INIT_ONER saves them all, so the return to BAS$$USER_HAND
1 !       restores them.
1 !
1 !       If necessary, GOSUB frames are removed looking for the
1 !       condition handling frame, but if another type of frame
1 !       is encountered we have an error.  If there is no error
1 !       pending then the RESUME is turned into a GOTO, for
1 !       compatability with BASIC-PLUS.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NEW_PC.ra.v     The location at which to continue execution.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       The resume PC.  This is returned to the caller of BAS$INIT_ONER,
1 !       which is presumed to be BAS$$USER_HAND.
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May cut back the stack, thus not returning to the caller.
1 !       If it does return, it is not to the call site but to the
1 !       location specified in the parameter.
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         FP;
2
```

```
;  1586          3056  2     LOCAL
;  1587          3057  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  1588          3058  2
;  1589          3059  2     !+
;  1590          3060  2     ! If there is no error being processed, stuff the parameter into the
;  1591          3061  2     ! return address and return, thus turning the RESUME statement into
;  1592          3062  2     ! a GOTO.
;  1593          3063  2     !-
;  1594          3064  2
;  1595          3065  2         IF (.BAS$L_ERRFLG EQL 0)
;  1596          3066  2         THEN
;  1597          3067  2             BEGIN
;  1598          3068  2             FMP = .FP;
;  1599          3069  2             FMP [BSF$A_SAVED_PC] = .NEW_PC;
;  1600          3070  2             RETURN (0);                         ! the value will be ignored
;  1601          3071  2             END;
;  1602          3072  2
;  1603          3073  2     !+
;  1604          3074  2     ! Dig back through GOSUB frames to find the condition handling frame.
;  1605          3075  2     !-
;  1606          3076  2         FMP = .FP;
;  1607          3077  2         FMP = .FMP [BSF$A_SAVED_FP];
;  1608          3078  2
;  1609          3079  2         WHILE (.FMP [BSF$B_PROC_CODE] EQL BSF$K_PROC_GOSB) DO
;  1610          3080  2             BEGIN
;  1611          3081  2             FMP = .FMP [BSF$A_SAVED_FP];
;  1612          3082  2
;  1613          3083  3             IF (.FMP [BSF$A_HANDLER] NEQA BAS$HANDLER)
;  1614          3084  3             THEN
;  1615          3085  3     !+
;  1616          3086  3     ! The previous frame is not a BASIC frame.  This means that the user
;  1617          3087  3     ! began processing an error, called a non-BASIC routine which called
;  1618          3088  3     ! a BASIC routine which tried to dismiss the error.  Disallow this
;  1619          3089  3     ! kind of poorly-structured code.
;  1620          3090  3     !-
;  1621          3091  3                 BAS$$SIGNAL (BAS$K_RESNO_ERR);
;  1622          3092  3
;  1623          3093  3     !+
;  1624          3094  3     ! Deallocate any heap storage that may be held by this frame.
;  1625          3095  3     !-
;  1626          3096  3             BAS$$UNWIND (.FMP);
;  1627          3097  3             END;
;  1628          3098  2
;  1629          3099  2     !+
;  1630          3100  2     ! We have finished cutting back the GOSUB frames.  Now be sure we are
;  1631          3101  2     ! in the condition handler.
;  1632          3102  2     !-
;  1633          3103  2
;  1634          3104  3         IF (.FMP [BSF$B_PROC_CODE] NEQ BSF$K_PROC_ONER)
;  1635          3105  3         THEN
;  1636          3106  3     !+
;  1637          3107  3     ! We are not.  This can happen if the user begins processing an error,
;  1638          3108  3     ! then calls another routine which tries to dismiss the error.
;  1639          3109  3     ! Disallow this, also.
;  1640          3110  3     !-
;  1641          3111  3             BAS$$SIGNAL (BAS$K_RESNO_ERR);
;  1642          3112  2
```

```
;  1643          3113  2 !+
;  1644          3114  2 ! We have reached the condition handling frame.  By stuffing the
;  1645          3115  2 ! pointer to this frame into FP we effectively cut back the stack,
;  1646          3116  2 ! since this routine's RET will then return from the condition
;  1647          3117  2 ! handler.  Note that we are not restoring any registers; we depend
;  1648          3118  2 ! on the fact that BAS$INIT_ONER saves them all, so the RET we are
;  1649          3119  2 ! about to do will restore registers for BAS$$USER_HAND.
;  1650          3120  2 !-
;  1651          3121  2     BAS$$UNWIND (.FMP);
;  1652          3122  2     FP = .FMP;
;  1653          3123  2 !+
;  1654          3124  2 ! By returning a number whose unsigned value is greater than 2
;  1655          3125  2 ! we indicate to BAS$$USER_HAND that the user has written RESUME
;  1656          3126  2 ! with a line number.
;  1657          3127  2 !-
;  1658          3128  2     RETURN (.NEW_PC);
;  1659          3129  1     END;                                    ! of BAS$RESUME

                                    000C 00000          .ENTRY  BAS$RESUME, Save R2,R3                  ; 2999
                    53 00000000G    00   9E 00002       MOVAB   BAS$$UNWIND, R3                         ;
                       00000000'    EF   D5 00009       TSTL    BAS$L_ERRFLG                            ; 3065
                                    0A   12 0000F       BNEQ    1$                                      ;
                    52              5D   D0 00011       MOVL    FP, FMP                                 ; 3068
                10  A2        04   AC   D0 00014       MOVL    NEW_PC, 16(FMP)                         ; 3069
                                    49   11 00019       BRB     6$                                      ; 3070
                    52              5D   D0 0001B 1$:   MOVL    FP, FMP                                 ; 3076
                    52         0C   A2   D0 0001E       MOVL    12(FMP), FMP                            ; 3077
                    06         E5   A2   91 00022 2$:   CMPB    -27(FMP), #6                            ; 3079
                                    20   12 00026       BNEQ    4$                                      ;
                    52         0C   A2   D0 00028       MOVL    12(FMP), FMP                            ; 3081
                    50 00000000G    00   9E 0002C       MOVAB   BAS$HANDLER, R0                         ; 3083
                    50              62   D1 00033       CMPL    (FMP), R0                               ;
                                    09   13 00036       BEQL    3$                                      ;
                    7E         00G  8F   9A 00038       MOVZBL  #BAS$K_RESNO_ERR, -(SP)                 ; 3091
                FCCE CF             01   FB 0003C       CALLS   #1, BAS$$SIGNAL                         ;
                                    52   DD 00041 3$:   PUSHL   FMP                                    ; 3096
                    63              01   FB 00043       CALLS   #1, BAS$$UNWIND                         ;
                                    DA   11 00046       BRB     2$                                      ; 3079
                    07         E5   A2   91 00048 4$:   CMPB    -27(FMP), #7                            ; 3104
                                    09   13 0004C       BEQL    5$                                      ;
                    7E         00G  8F   9A 0004E       MOVZBL  #BAS$K_RESNO_ERR, -(SP)                 ; 3111
                FCB8 CF             01   FB 00052       CALLS   #1, BAS$$SIGNAL                         ;
                                    52   DD 00057 5$:   PUSHL   FMP                                    ; 3121
                    63              01   FB 00059       CALLS   #1, BAS$$UNWIND                         ;
                    5D              52   D0 0005C       MOVL    FMP, FP                                 ; 3122
                    50         04   AC   D0 0005F       MOVL    NEW_PC, R0                              ; 3128
                                    04   00063       RET                                             ;
                                    50   D4 00064 6$:   CLRL    R0                                      ; 3129
                                    04   00066       RET                                             ;

; Routine Size:  103 bytes,    Routine Base:  _BAS$CODE + 04F1
```

```
; 1660          3130  1
```

```
;  1662          3131  1 GLOBAL ROUTINE BAS$RESUME_Z =                    ! Resume with no line number
;  1663          3132  1
;  1664          3133  1 !++
;  1665          3134  1 ! FUNCTIONAL DESCRIPTION:
;  1666          3135  1 !
;  1667          3136  1 !       Resume execution from an error handler.  The compiled code
;  1668          3137  1 !       calls RESUME_Z to indicate that the statement in which the
;  1669          3138  1 !       error occurred is to be restarted or continued.  (Which
;  1670          3139  1 !       depends on which error is in progress.)
;  1671          3140  1 !       The stack is cut back to the call to BAS$INIT_ONER which
;  1672          3141  1 !       is in BAS$$USER_HAND and the RET at the end of this routine
;  1673          3142  1 !       actually returns from BAS$INIT_ONER.  This is similar to
;  1674          3143  1 !       GOSUB processing.  To simplify the restoring of registers
;  1675          3144  1 !       BAS$INIT_ONER saves them all, so the return to BAS$$USER_HAND
;  1676          3145  1 !       restores them.
;  1677          3146  1 !
;  1678          3147  1 !       If necessary, GOSUB frames are removed looking for the
;  1679          3148  1 !       condition handling frame, but if another type of frame
;  1680          3149  1 !       is encountered we have an error.
;  1681          3150  1 !
;  1682          3151  1 ! FORMAL PARAMETERS:
;  1683          3152  1 !
;  1684          3153  1 !       NONE
;  1685          3154  1 !
;  1686          3155  1 ! IMPLICIT INPUTS:
;  1687          3156  1 !
;  1688          3157  1 !       NONE
;  1689          3158  1 !
;  1690          3159  1 ! IMPLICIT OUTPUTS:
;  1691          3160  1 !
;  1692          3161  1 !       NONE
;  1693          3162  1 !
;  1694          3163  1 ! ROUTINE VALUE:
;  1695          3164  1 !
;  1696          3165  1 !       USER_ERR_RSUMZ, to indicate to BAS$$USER_HAND that the user did
;  1697          3166  1 !       a RESUME with no line number.
;  1698          3167  1 !
;  1699          3168  1 ! COMPLETION CODES:
;  1700          3169  1 !
;  1701          3170  1 !       NONE
;  1702          3171  1 !
;  1703          3172  1 ! SIDE EFFECTS:
;  1704          3173  1 !
;  1705          3174  1 !       Cuts back the stack, thus not returning to the caller.
;  1706          3175  1 !
;  1707          3176  1 !--
;  1708          3177  1
;  1709          3178  2     BEGIN
;  1710          3179  2
;  1711          3180  2     BUILTIN
;  1712          3181  2         FP;
;  1713          3182  2
;  1714          3183  2     LOCAL
;  1715          3184  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  1716          3185  2
;  1717          3186  2 !+
;  1718          3187  2 ! If there is no error being processed, the RESUME statement without
```

```
;  1719     3188  2 ! a line number is invalid.
;  1720     3189  2 !-
;  1721     3190  2
;  1722     3191  2     IF (.BAS$L_ERRFLG EQL 0) THEN BAS$$SIGNAL (BAS$K_RESNO_ERR);
;  1723     3192  2
;  1724     3193  2 !+
;  1725     3194  2 ! If we are not in the same program unit as the source of the error,
;  1726     3195  2 ! we have an error.  This is done for compatability with the PDP-11.
;  1727     3196  2 !-
;  1728     3197  2     FMP = .FP;
;  1729     3198  2     FMP = .FMP [BSF$A_SAVED_FP];
;  1730     3199  2
;  1731     3200  2     IF ((BAS$$MODULE (.FMP) + 1) NEQA .BAS$T_ERN [DSC$A_POINTER]) THEN BAS$$STOP (BAS$K_ILLRESSUB);
;  1732     3201  2
;  1733     3202  2 !+
;  1734     3203  2 ! Dig back through GOSUB frames to find the condition handling frame.
;  1735     3204  2 !-
;  1736     3205  2
;  1737     3206  2     WHILE (.FMP [BSF$B_PROC_CODE] EQL BSF$K_PROC_GOSB) DO
;  1738     3207  3         BEGIN
;  1739     3208  3         FMP = .FMP [BSF$A_SAVED_FP];
;  1740     3209  3
;  1741     3210  3         IF (.FMP [BSF$A_HANDLER] NEQA BAS$HANDLER)
;  1742     3211  3         THEN
;  1743     3212  3 !+
;  1744     3213  3 ! The previous frame is not a BASIC frame.  This means that the user
;  1745     3214  3 ! began processing an error, called a non-BASIC routine which called
;  1746     3215  3 ! a BASIC routine which tried to dismiss the error.  Disallow this
;  1747     3216  3 ! kind of poorly-structured code.
;  1748     3217  3 !-
;  1749     3218  3             BAS$$SIGNAL (BAS$K_RESNO_ERR);
;  1750     3219  3
;  1751     3220  3 !+
;  1752     3221  3 ! Deallocate any heap storage that may be held by this frame.
;  1753     3222  3 !-
;  1754     3223  3         BAS$$UNWIND (.FMP);
;  1755     3224  2         END;
;  1756     3225  2
;  1757     3226  2 !+
;  1758     3227  2 ! We have finished cutting back the GOSUB frames.  Now be sure we are
;  1759     3228  2 ! in the condition handler.
;  1760     3229  2 !-
;  1761     3230  2
;  1762     3231  2     IF (.FMP [BSF$B_PROC_CODE] NEQ BSF$K_PROC_ONER)
;  1763     3232  2     THEN
;  1764     3233  2 !+
;  1765     3234  2 ! We are not.  This can happen if the user begins processing an error,
;  1766     3235  2 ! then calls another routine which tries to dismiss the error.
;  1767     3236  2 ! Disallow this, also.
;  1768     3237  2 !-
;  1769     3238  2         BAS$$SIGNAL (BAS$K_ILLRESSUB);
;  1770     3239  2
;  1771     3240  2 !+
;  1772     3241  2 ! We have reached the condition handling frame.  By stuffing the
;  1773     3242  2 ! pointer to this frame into FP we effectively cut back the stack,
;  1774     3243  2 ! since this routine's RET will then return from the condition
;  1775     3244  2 ! handler.  Note that we are not restoring any registers; we depend
```

```
;  1776          3245  2 ! on the fact that BAS$INIT_ONER saves them all, so the RET we are
;  1777          3246  2 ! about to do will restore registers for BAS$$USER_HAND.
;  1778          3247  2 !-
;  1779          3248  2     BAS$$UNWIND (.FMP);
;  1780          3249  2     FP = .FMP;
;  1781          3250  2 !+
;  1782          3251  2 ! Indicate to BAS$$USER_HAND that the user has written a RESUME
;  1783          3252  2 ! with no line number.
;  1784          3253  2 !-
;  1785          3254  2     RETURN (USER_ERR_RSUMZ);
;  1786          3255  1     END;                                    ! of BAS$RESUME_Z



                                    001C 00000     .ENTRY  BAS$RESUME_Z, Save R2,R3,R4                     ; 3131
                54 00000000G   00   9E 00002     MOVAB   BAS$$UNWIND, R4                                 ;
                53     FC9B    CF   9E 00009     MOVAB   BAS$$SIGNAL, R3                                 ;
                   00000000'   EF   D5 0000E     TSTL    BAS$L_ERRFLG                                    ; 3191
                               07   12 00014     BNEQ    1$                                              ;
                7E        00G  8F   9A 00016     MOVZBL  #BAS$K_RESNO_ERR, -(SP)                         ;
                63             01   FB 0001A     CALLS   #1, BAS$$SIGNAL                                 ;
                52             5D   D0 0001D 1$:  MOVL    FP, FMP                                         ; 3197
                52        0C   A2   D0 00020     MOVL    12(FMP), FMP                                    ; 3198
                               52   DD 00024     PUSHL   FMP                                             ; 3200
            00E6   C3          01   FB 00026     CALLS   #1, BAS$$MODULE                                 ;
                               50   D6 0002B     INCL    R0                                              ;
    00000000'   EF             50   D1 0002D     CMPL    R0, BAS$T_ERN+4                                 ;
                               08   13 00034     BEQL    2$                                              ;
                7E        00G  8F   9A 00036     MOVZBL  #BAS$K_ILLRESSUB, -(SP)                         ;
              14   A3          01   FB 0003A     CALLS   #1, BAS$$STOP                                   ;
                06        E5   A2   91 0003E 2$:  CMPB    -27(FMP), #6                                    ; 3206
                               1E   12 00042     BNEQ    4$                                              ;
                52        0C   A2   D0 00044     MOVL    12(FMP), FMP                                    ; 3208
                50 00000000G   00   9E 00048     MOVAB   BAS$HANDLER, R0                                 ; 3210
                50             62   D1 0004F     CMPL    (FMP), R0                                       ;
                               07   13 00052     BEQL    3$                                              ;
                7E        00G  8F   9A 00054     MOVZBL  #BAS$K_RESNO_ERR, -(SP)                         ; 3218
                63             01   FB 00058     CALLS   #1, BAS$$SIGNAL                                 ;
                               52   DD 0005B 3$:  PUSHL   FMP                                             ; 3223
                64             01   FB 0005D     CALLS   #1, BAS$$UNWIND                                 ;
                               DC   11 00060     BRB     2$                                              ; 3206
                07        E5   A2   91 00062 4$:  CMPB    -27(FMP), #7                                    ; 3231
                               07   13 00066     BEQL    5$                                              ;
                7E        00G  8F   9A 00068     MOVZBL  #BAS$K_ILLRESSUB, -(SP)                         ; 3238
                63             01   FB 0006C     CALLS   #1, BAS$$SIGNAL                                 ;
                               52   DD 0006F 5$:  PUSHL   FMP                                             ; 3248
                64             01   FB 00071     CALLS   #1, BAS$$UNWIND                                 ;
                5D             52   D0 00074     MOVL    FMP, FP                                         ; 3249
                               50   D4 00077     CLRL    R0                                              ; 3254
                                    04 00079     RET                                                     ; 3255

; Routine Size:  122 bytes,    Routine Base:  _BAS$CODE + 0558

;  1787          3256  1
```

```
;  1789          3257  1 GLOBAL ROUTINE BAS$ON_ERR_Z =                    ! ON ERROR GOTO 0
;  1790          3258  1
;  1791          3259  1 !++
;  1792          3260  1 ! FUNCTIONAL DESCRIPTION:
;  1793          3261  1 !
;  1794          3262  1 !       The BASIC statement ON ERROR GOTO 0 is compiled as
;  1795          3263  1 !
;  1796          3264  1 !               CALLS   #0,BAS$ON_ERR_Z
;  1797          3265  1 !               CLRL    BAS$A_USER_HAND(Rn)
;  1798          3266  1 !
;  1799          3267  1 !       Thus, the job of BAS$ON_ERR_Z is to return if there is no
;  1800          3268  1 !       error in progress, and to provide system error processing
;  1801          3269  1 !       if there is an error in progress.  This latter function is
;  1802          3270  1 !       done by cutting back the stack (like RESUME) and returning
;  1803          3271  1 !       to BAS$$USER_HAND indicating that system error handling
;  1804          3272  1 !       is required.
;  1805          3273  1 !
;  1806          3274  1 ! FORMAL PARAMETERS:
;  1807          3275  1 !
;  1808          3276  1 !       NONE
;  1809          3277  1 !
;  1810          3278  1 ! IMPLICIT INPUTS:
;  1811          3279  1 !
;  1812          3280  1 !       BAS$L_ERRFLG
;  1813          3281  1 !
;  1814          3282  1 ! IMPLICIT OUTPUTS:
;  1815          3283  1 !
;  1816          3284  1 !       NONE
;  1817          3285  1 !
;  1818          3286  1 ! ROUTINE VALUE:
;  1819          3287  1 !
;  1820          3288  1 !       Either no value, or USER_ERR_OEGZ, to indicate to BAS$$USER_HAND
;  1821          3289  1 !       that the user did an ON ERROR GOTO 0 in his error handler.
;  1822          3290  1 !
;  1823          3291  1 ! COMPLETION CODES:
;  1824          3292  1 !
;  1825          3293  1 !       NONE
;  1826          3294  1 !
;  1827          3295  1 ! SIDE EFFECTS:
;  1828          3296  1 !
;  1829          3297  1 !       May cut back the stack, thus not returning to the caller.
;  1830          3298  1 !
;  1831          3299  1 !--
;  1832          3300  1
;  1833          3301  2     BEGIN
;  1834          3302  2
;  1835          3303  2     BUILTIN
;  1836          3304  2         FP;
;  1837          3305  2
;  1838          3306  2     LOCAL
;  1839          3307  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  1840          3308  2
;  1841          3309  2 !+
;  1842          3310  2 ! If there is no error being processed, just return.  The value will
;  1843          3311  2 ! be ignored.
;  1844          3312  2 !-
;  1845          3313  2
```

```
;  1846          3314  2     IF (.BAS$L_ERRFLG EQL 0) THEN RETURN (0);
;  1847          3315  2
;  1848          3316  2 !+
;  1849          3317  2 ! Dig back through GOSUB frames to find the condition handling frame.
;  1850          3318  2 !-
;  1851          3319  2     FMP = .FP;
;  1852          3320  2     FMP = .FMP [BSF$A_SAVED_FP];
;  1853          3321  2
;  1854          3322  2     WHILE (.FMP [BSF$B_PROC_CODE] EQL BSF$K_PROC_GOSB) DO
;  1855          3323  3         BEGIN
;  1856          3324  3         FMP = .FMP [BSF$A_SAVED_FP];
;  1857          3325  3
;  1858          3326  3         IF (.FMP [BSF$A_HANDLER] NEQA BAS$HANDLER)
;  1859          3327  3         THEN
;  1860          3328  4 !+
;  1861          3329  4 ! The previous frame is not a BASIC frame.  This means that the user
;  1862          3330  4 ! began processing an error, called a non-BASIC routine which called
;  1863          3331  4 ! a BASIC routine which tried to specify system error handling.
;  1864          3332  4 ! Disallow this kind of poorly-structured code.
;  1865          3333  4 !-
;  1866          3334  4             BAS$$SIGNAL (BAS$K_RESNO_ERR);
;  1867          3335  4
;  1868          3336  3 !+
;  1869          3337  3 ! Deallocate any heap storage that may be held by this frame.
;  1870          3338  3 !-
;  1871          3339  3         BAS$$UNWIND (.FMP);
;  1872          3340  3         END;
;  1873          3341  2
;  1874          3342  2 !+
;  1875          3343  2 ! We have finished cutting back the GOSUB frames.  Now be sure we are
;  1876          3344  2 ! in the condition handler.
;  1877          3345  2 !-
;  1878          3346  2
;  1879          3347  3     IF (.FMP [BSF$B_PROC_CODE] NEQ BSF$K_PROC_ONER)
;  1880          3348  3     THEN
;  1881          3349  4 !+
;  1882          3350  4 ! We are not. This can happen if the user begins processing an error,
;  1883          3351  4 ! then calls another routine which tries to specify system error
;  1884          3352  4 ! handling.  Disallow this, also.
;  1885          3353  4 !-
;  1886          3354  4         BAS$$SIGNAL (BAS$K_RESNO_ERR);
;  1887          3355  4
;  1888          3356  3 !+
;  1889          3357  3 ! We have reached the condition handling frame.  By stuffing the
;  1890          3358  3 ! pointer to this frame into FP we effectively cut back the stack,
;  1891          3359  3 ! since this routine's RET will then return from the condition
;  1892          3360  3 ! handler.  Note that we are not restoring any registers; we depend
;  1893          3361  3 ! on the fact that BAS$INIT_ONER saves them all, so the RET we are
;  1894          3362  3 ! about to do will restore registers for BAS$$USER_HAND.
;  1895          3363  3 !-
;  1896          3364  3     BAS$$UNWIND (.FMP);
;  1897          3365  3     FP = .FMP;
;  1898          3366  3 !+
;  1899          3367  3 ! Indicate to BAS$$USER_HAND that the user has written ON ERROR GOTO 0.
;  1900          3368  3 !-
;  1901          3369  3     RETURN (USER_ERR_OEGZ);
;  1902          3370  1     END;                                    ! of BAS$ON_ERR_Z
```

```
                                 000C 00000          .ENTRY  BAS$ON_ERR_Z, Save R2,R3                        ; 3257
                53 00000000G     00   9E 00002      MOVAB   BAS$$UNWIND, R3                                 ;
                   00000000'     EF   D5 00009      TSTL    BAS$L_ERRFLG                                    ; 3314
                                 48   13 0000F      BEQL    5$                                              ;
                52               5D   D0 00011      MOVL    FP, FMP                                         ; 3319
                52          0C   A2   D0 00014      MOVL    12(FMP), FMP                                    ; 3320
                06          E5   A2   91 00018 1$:   CMPB    -27(FMP), #6                                    ; 3322
                                 20   12 0001C      BNEQ    3$                                              ;
                52          0C   A2   D0 0001E      MOVL    12(FMP), FMP                                    ; 3324
                50 00000000G     00   9E 00022      MOVAB   BAS$HANDLER, R0                                 ; 3326
                50               62   D1 00029      CMPL    (FMP), R0                                       ;
                                 09   13 0002C      BEQL    2$                                              ;
                7E          00G  8F   9A 0002E      MOVZBL  #BAS$K_RESNO_ERR, -(SP)                         ; 3334
            FBF7 CF              01   FB 00032      CALLS   #1, BAS$$SIGNAL                                 ;
                                 52   DD 00037 2$:   PUSHL   FMP                                           ; 3339
                63               01   FB 00039      CALLS   #1, BAS$$UNWIND                                 ;
                                 DA   11 0003C      BRB     1$                                              ; 3322
                07          E5   A2   91 0003E 3$:   CMPB    -27(FMP), #7                                    ; 3347
                                 09   13 00042      BEQL    4$                                              ;
                7E          00G  8F   9A 00044      MOVZBL  #BAS$K_RESNO_ERR, -(SP)                         ; 3354
            FBE1 CF              01   FB 00048      CALLS   #1, BAS$$SIGNAL                                 ;
                                 52   DD 0004D 4$:   PUSHL   FMP                                           ; 3364
                63               01   FB 0004F      CALLS   #1, BAS$$UNWIND                                 ;
                5D               52   D0 00052      MOVL    FMP, FP                                         ; 3365
                50               02   D0 00055      MOVL    #2, R0                                          ; 3369
                                 04   00058          RET                                                     ;
                                 50   D4 00059 5$:   CLRL    R0                                              ; 3370
                                 04   0005B          RET                                                     ;

; Routine Size:  92 bytes,    Routine Base:  _BAS$CODE + 05D2

; 1903          3371  1
```

```
;  1905     3372  1 GLOBAL ROUTINE BAS$ON_ERR_BK =                    ! ON ERROR GO BACK
;  1906     3373  1
;  1907     3374  1 !++
;  1908     3375  1 ! FUNCTIONAL DESCRIPTION:
;  1909     3376  1 !
;  1910     3377  1 !       The BASIC statement ON ERROR GO BACK is compiled as
;  1911     3378  1 !
;  1912     3379  1 !               CALLS   #0,BAS$ON_ERR_BK
;  1913     3380  1 !               MOVZBL  #1,BAS$A_USER_HAND(Rn)
;  1914     3381  1 !
;  1915     3382  1 !       Thus, the job of BAS$ON_ERR_BK is to return if there is no
;  1916     3383  1 !       error in progress, and to permit the caller of the BASIC
;  1917     3384  1 !       procedure to try to process the error if there is an error
;  1918     3385  1 !       in progress.  This latter function is done by cutting back
;  1919     3386  1 !       the stack (like RESUME) and returning to BAS$$USER_HAND with
;  1920     3387  1 !       a USER_ERR_GOBK, indicating that it is to revert.
;  1921     3388  1 !
;  1922     3389  1 ! FORMAL PARAMETERS:
;  1923     3390  1 !
;  1924     3391  1 !       NONE
;  1925     3392  1 !
;  1926     3393  1 ! IMPLICIT INPUTS:
;  1927     3394  1 !
;  1928     3395  1 !       BAS$L_ERRFLG
;  1929     3396  1 !
;  1930     3397  1 ! IMPLICIT OUTPUTS:
;  1931     3398  1 !
;  1932     3399  1 !       NONE
;  1933     3400  1 !
;  1934     3401  1 ! ROUTINE VALUE:
;  1935     3402  1 !
;  1936     3403  1 !       Either no value, or USER_ERR_GOBK.
;  1937     3404  1 !
;  1938     3405  1 ! COMPLETION CODES:
;  1939     3406  1 !
;  1940     3407  1 !       NONE
;  1941     3408  1 !
;  1942     3409  1 ! SIDE EFFECTS:
;  1943     3410  1 !
;  1944     3411  1 !       May cut back the stack, thus not returning to the caller.
;  1945     3412  1 !
;  1946     3413  1 !--
;  1947     3414  1
;  1948     3415  2     BEGIN
;  1949     3416  2
;  1950     3417  2     BUILTIN
;  1951     3418  2         FP;
;  1952     3419  2
;  1953     3420  2     LOCAL
;  1954     3421  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD);
;  1955     3422  2
;  1956     3423  2 !+
;  1957     3424  2 ! If there is no error being processed, just return.  The value will
;  1958     3425  2 ! be ignored.
;  1959     3426  2 !-
;  1960     3427  2
;  1961     3428  2     IF (.BAS$L_ERRFLG EQL 0) THEN RETURN (0);
```

```
;  1962          3429  2
;  1963          3430  2 !+
;  1964          3431  2 ! Dig back through GOSUB frames to find the condition handling frame.
;  1965          3432  2 !-
;  1966          3433  2     FMP = .FP;
;  1967          3434  2     FMP = .FMP [BSF$A_SAVED_FP];
;  1968          3435  2
;  1969          3436  2     WHILE (.FMP [BSF$B_PROC_CODE] EQL BSF$K_PROC_GOSB) DO
;  1970          3437  3         BEGIN
;  1971          3438  3         FMP = .FMP [BSF$A_SAVED_FP];
;  1972          3439  3
;  1973          3440  4         IF (.FMP [BSF$A_HANDLER] NEQA BAS$HANDLER)
;  1974          3441  4         THEN
;  1975          3442  3 !+
;  1976          3443  3 ! The previous frame is not a BASIC frame.  This means that the user
;  1977          3444  3 ! began processing an error, called a non-BASIC routine which called
;  1978          3445  3 ! a BASIC routine which tried to allow the caller of the routine that
;  1979          3446  3 ! got the error to handle the error.  Disallow this kind of
;  1980          3447  3 ! poorly-structured code.
;  1981          3448  3 !-
;  1982          3449  3             BAS$$SIGNAL (BAS$K_RESNO_ERR);
;  1983          3450  3
;  1984          3451  3 !+
;  1985          3452  3 ! Deallocate any heap storage that may be held by this frame.
;  1986          3453  3 !-
;  1987          3454  3         BAS$$UNWIND (.FMP);
;  1988          3455  3         END;
;  1989          3456  2
;  1990          3457  2 !+
;  1991          3458  2 ! We have finished cutting back the GOSUB frames.  Now be sure we are
;  1992          3459  2 ! in the condition handler.
;  1993          3460  2 !-
;  1994          3461  2
;  1995          3462  3     IF (.FMP [BSF$B_PROC_CODE] NEQ BSF$K_PROC_ONER)
;  1996          3463  2     THEN
;  1997          3464  2 !+
;  1998          3465  2 ! We are not.  This can happen if the user begins processing an error,
;  1999          3466  2 ! then calls another routine which tries to specify system error
;  2000          3467  2 ! handling.  Disallow this, also.
;  2001          3468  2 !-
;  2002          3469  2         BAS$$SIGNAL (BAS$K_IMPERRHAN);
;  2003          3470  2
;  2004          3471  2 !+
;  2005          3472  2 ! We have reached the condition handling frame.  By stuffing the
;  2006          3473  2 ! pointer to this frame into FP we effectively cut back the stack,
;  2007          3474  2 ! since this routine's RET will then return from the condition
;  2008          3475  2 ! handler.  Note that we are not restoring any registers; we depend
;  2009          3476  2 ! on the fact that BAS$INIT_ONER saves them all, so the RET we are
;  2010          3477  2 ! about to do will restore registers for BAS$$USER_HAND.
;  2011          3478  2 !-
;  2012          3479  2     BAS$$UNWIND (.FMP);
;  2013          3480  2     FP = .FMP;
;  2014          3481  2 !+
;  2015          3482  2 ! Indicate to BAS$$USER_HAND that the user has written ON ERROR GO BACK.
;  2016          3483  2 !-
;  2017          3484  2     RETURN (USER_ERR_GOBK);
;  2018          3485  1     END;                                    ! of BAS$ON_ERR_BK
```

```
                          000C 00000          .ENTRY  BAS$ON_ERR_BK, Save R2,R3                       ; 3372
          53 00000000G    00   9E 00002      MOVAB   BAS$$UNWIND, R3                                 ;
             00000000'    EF   D5 00009      TSTL    BAS$L_ERRFLG                                    ; 3428
                          48   13 0000F      BEQL    5$                                              ;
          52              5D   D0 00011      MOVL    FP, FMP                                         ; 3433
          52        0C    A2   D0 00014      MOVL    12(FMP), FMP                                    ; 3434
          06        E5    A2   91 00018 1$:  CMPB    -27(FMP), #6                                    ; 3436
                          20   12 0001C      BNEQ    3$                                              ;
          52        0C    A2   D0 0001E      MOVL    12(FMP), FMP                                    ; 3438
          50 00000000G    00   9E 00022      MOVAB   BAS$HANDLER, R0                                 ; 3440
          50              62   D1 00029      CMPL    (FMP), R0                                       ;
                          09   13 0002C      BEQL    2$                                              ;
          7E        00G   8F   9A 0002E      MOVZBL  #BAS$K_RESNO_ERR, -(SP)                         ; 3449
      FB9B CF             01   FB 00032      CALLS   #1, BAS$$SIGNAL                                 ;
                          52   DD 00037 2$:  PUSHL   FMP                                            ; 3454
          63              01   FB 00039      CALLS   #1, BAS$$UNWIND                                 ;
                          DA   11 0003C      BRB     1$                                              ; 3436
          07        E5    A2   91 0003E 3$:  CMPB    -27(FMP), #7                                    ; 3462
                          09   13 00042      BEQL    4$                                              ;
          7E        00G   8F   9A 00044      MOVZBL  #BAS$K_IMPERRHAN, -(SP)                         ; 3469
      FB85 CF             01   FB 00048      CALLS   #1, BAS$$SIGNAL                                 ;
                          52   DD 0004D 4$:  PUSHL   FMP                                            ; 3479
          63              01   FB 0004F      CALLS   #1, BAS$$UNWIND                                 ;
          5D              52   D0 00052      MOVL    FMP, FP                                         ; 3480
          50              01   D0 00055      MOVL    #1, R0                                          ; 3484
                               04 00058      RET                                                     ;
                          50   D4 00059 5$:  CLRL    R0                                              ; 3485
                               04 0005B      RET                                                     ;

; Routine Size:  92 bytes,    Routine Base:  _BAS$CODE + 062E

; 2019          3486  1
```

```
;  2021          3487  1 GLOBAL ROUTINE BAS$$HANDLER (                         ! handler for BASIC compiled code
;  2022          3488  1         SIGNAL_ARGS,                                  ! VAX/VMS signal arguments
;  2023          3489  1         MECHANISM_ARGS                                ! VAX/VMS mechanism arguments
;  2024          3490  1     ) =
;  2025          3491  1
;  2026          3492  1 !++
;  2027          3493  1 ! FUNCTIONAL DESCRIPTION:
;  2028          3494  1 !
;  2029          3495  1 !       Handle an exception from within a BASIC-PLUS-2 routine.
;  2030          3496  1 !       Note that the real entry point, BAS$HANDLER, is a location in
;  2031          3497  1 !       the sharable library's vector (or in a small module if this code
;  2032          3498  1 !       is not shared) so that a frame can be tested for being a BASIC
;  2033          3499  1 !       frame by testing for BAS$HANDLER in 0(FP).
;  2034          3500  1 !
;  2035          3501  1 ! FORMAL PARAMETERS:
;  2036          3502  1 !
;  2037          3503  1 !       SIGNAL_ARGS     A vector of longwords which indicate the nature
;  2038          3504  1 !                       of the condition.
;  2039          3505  1 !       MECHANISM_ARGS  A vector of longwords that indicate the state
;  2040          3506  1 !                       of the process at the time of the signal.
;  2041          3507  1 !
;  2042          3508  1 ! IMPLICIT INPUTS:
;  2043          3509  1 !
;  2044          3510  1 !       The information in the frames of the BASIC-PLUS-2 routines
;  2045          3511  1 !       in and before the one which encountered the error.
;  2046          3512  1 !
;  2047          3513  1 ! IMPLICIT OUTPUTS:
;  2048          3514  1 !
;  2049          3515  1 !       NONE
;  2050          3516  1 !
;  2051          3517  1 ! ROUTINE VALUE:
;  2052          3518  1 !
;  2053          3519  1 !       An indication to the VAX/VMS CHF of whether or not to revert.
;  2054          3520  1 !
;  2055          3521  1 ! COMPLETION CODES:
;  2056          3522  1 !
;  2057          3523  1 !       SS$_RESIGNAL
;  2058          3524  1 !       SS$_CONTINUE
;  2059          3525  1 !
;  2060          3526  1 ! SIDE EFFECTS:
;  2061          3527  1 !
;  2062          3528  1 !       May do an UNWIND to let the BASIC-PLUS-2 code process the error.
;  2063          3529  1 !       On an UNWIND, will deallocate any heap storage held by its frame.
;  2064          3530  1 !
;  2065          3531  1 !--
;  2066          3532  1
;  2067          3533  2     BEGIN
;  2068          3534  2
;  2069          3535  2     MAP
;  2070          3536  2         SIGNAL_ARGS : REF BLOCK [0, BYTE],
;  2071          3537  2         MECHANISM_ARGS : REF BLOCK [0, BYTE];
;  2072          3538  2
;  2073          3539  2     BUILTIN
;  2074          3540  2         CALLG,                                        ! call with hand-built argument list
;  2075          3541  2         FP;
;  2076          3542  2
;  2077          3543  2     LOCAL
```

```
;  2078          3544  2          COND_VAL : BLOCK [4, BYTE],                 ! condition value being signaled
;  2079          3545  2          TEMP_COND_VAL : BLOCK [4, BYTE],            ! temp, for copying
;  2080          3546  2          NUM_FAO_ARGS,                               ! number of FAO arguments while copying
;  2081          3547  2          SCAN_DONE,                                  ! used to control copying loop
;  2082          3548  2          COPY_LIMIT,                                 ! likewise
;  2083          3549  2          FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD),      ! frame pointer
;  2084          3550  2          LEN_VECTOR : VOLATILE,                      ! length of new vector
;  2085          3551  2          NEW_VECTOR : REF BLOCK [0, BYTE] VOLATILE,      ! the new vector to be signaled
;  2086          3552  2          PUTTER,                                     ! used to store in new vector
;  2087          3553  2          GETTER,                                     ! used to fetch from old vector
;  2088          3554  2          GET_VM_RESULT,                              ! condition value from calling LIB$GET_VM
;  2089          3555  2          USER_RESULT,                                ! condition value from calling BAS$$USER_HAND
;  2090          3556  2          TOP_LEVEL,                                  ! true if we have original jurisdiction
;  2091          3557  2          COND_VAL_CHANGE,                            ! true if we translated cond
;  2092          3558  2          RESTART_IO_FLAG;                            ! This is a restartable I/O statement
;  2093          3559  2
;  2094          3560  2      !+
;  2095          3561  2      ! The SYSTEM_ERROR cell is 1 if the user specified that system
;  2096          3562  2      ! error handling is to take place.  It is set based on a call to
;  2097          3563  2      ! BAS$$USER_HAND, and, if set, BAS$$USER_HAND will not be called
;  2098          3564  2      ! again for this error.
;  2099          3565  2      ! The GONE_BACK cell is 0 if this is the first instance of the
;  2100          3566  2      ! BASIC handler to see this error, and 1 if we have gone through
;  2101          3567  2      ! a reversion (if only to get traceback).
;  2102          3568  2      !-
;  2103          3569  2      !+
;  2104          3570  2      ! Declare a handler to clean up heap storage.
;  2105          3571  2      !-
;  2106          3572  2
;  2107          3573  2          ENABLE
;  2108          3574  2              HANDLER_HANDLER (LEN_VECTOR, NEW_VECTOR);
;  2109          3575  2
;  2110          3576  2      !+
;  2111          3577  2      ! In case we cut back frames, put the signal argument PSL into our frame
;  2112          3578  2      ! so the PSL will be correctly restored.  If the emulator is invoked,
;  2113          3579  2      ! it will put an extra frame on the stack and the return PSL will be
;  2114          3580  2      ! incorrect if we fail to do this.
;  2115          3581  2      !-
;  2116          3582  2
;  2117          3583  3          BEGIN
;  2118          3584  3          LOCAL
;  2119          3585  3              FRAME : REF VECTOR [4,WORD];
;  2120          3586  3
;  2121          3587  3          FRAME = .FP;
;  2122          3588  3          IF .SIGNAL_ARGS [CHF$L_SIG_ARGS] GTR 1
;  2123          3589  3          THEN
;  2124          3590  3              FRAME [2] = .SIGNAL_ARGS [(.SIGNAL_ARGS [CHF$L_SIG_ARGS] * %UPVAL),0,16,0];
;  2125          3591  2          END;
;  2126          3592  2
;  2127          3593  2      !+
;  2128          3594  2      ! Set up a pointer to our frame.
;  2129          3595  2      !-
;  2130          3596  2          FMP = .MECHANISM_ARGS [CHF$L_MCH_FRAME];
;  2131          3597  2
;  2132          3598  2      !+
;  2133          3599  2      ! Remember whether we are the first handler to process this error,
;  2134          3600  2      ! and tell deeper handlers that they are not.
```

```
;  2135          3601  2 !-
;  2136          3602  2     TOP_LEVEL = (IF (.GONE_BACK) THEN 0 ELSE 1);
;  2137          3603  2     GONE_BACK = 1;
;  2138          3604  2 !+
;  2139          3605  2 ! Check for certain non-BASIC errors.  Many of these are
;  2140          3606  2 ! converted to their equivalent BASIC error.
;  2141          3607  2 !-
;  2142          3608  2     COND_VAL = .SIGNAL_ARGS [CHF$L_SIG_NAME];
;  2143          3609  2
;  2144          3610  2     IF (.COND_VAL [STS$V_FAC_NO] EQL BAS$K_FAC_NO)
;  2145          3611  2     THEN
;  2146          3612  2         COND_VAL_CHANGE = 0
;  2147          3613  2     ELSE
;  2148          3614  3         BEGIN
;  2149          3615  3
;  2150          3616  3         CASE LIB$MATCH_COND (COND_VAL,                  !
;  2151          3617  3                 %REF (MTH$_SQUROONEG),                  ! 1 = negative square root
;  2152          3618  3                 %REF (MTH$_LOGZERNEG),                  ! 2 = negative or zero log
;  2153          3619  3                 %REF (MTH$_FLOOVEMAT),                  ! 3 = floating overflow (EXP or TAN)
;  2154          3620  3                 %REF (SS$_FLTDIV),                      ! 4 = floating divide by zero
;  2155          3621  3                 %REF (SS$_INTDIV),                      ! 5 = integer divide by zero
;  2156          3622  3                 %REF (SS$_FLTOVF),                      ! 6 = floating overflow
;  2157          3623  3                 %REF (SS$_INTOVF),                      ! 7 = integer overflow
;  2158          3624  3                 %REF (MTH$_SIGLOSMAT),                  ! 8 = significance lost in math library
;  2159          3625  3                 %REF (MTH$_UNDEXP),                     ! 9 = undefined ** operation
;  2160          3626  3                 %REF (SS$_ACCVIO),                      ! 10 = access violation
;  2161          3627  3                 %REF (SS$_ROPRAND),                     ! 11 = reserved (floating) operand
;  2162          3628  3                 %REF (SS$_SUBRNG),                      ! 12 = subscript out of range
;  2163          3629  3                 %REF (STR$_INSVIRMEM),                  ! 13 = insufficient virtual memory (strings)
;  2164          3630  3                 %REF (STR$_DIVBY_ZER),                  ! 14 = String divide by zero
;  2165          3631  3                 %REF (OTS$_IO_CONCLO),                  ! 15 = I/O continued to closed file
;  2166          3632  3                 %REF (SS$_UNWIND),                      ! 16 = unwinding through this frame
;  2167          3633  3                 %REF (STR$_STRTOOLON),                  ! 17 = String too long (greater than 65535)
;  2168          3634  3                 %REF (SS$_FLTDIV_F),                    ! 18 = floating divide by zero fault
;  2169          3635  3                 %REF (SS$_FLTOVF_F),                    ! 19 = floating overflow fault
;  2170          3636  3                 %REF (SS$_DECOVF)                       ! 20 = decimal overflow
;  2171          3637  3             FROM 0 TO 20 OF
;  2172          3638  3                 SET
;  2173          3639  3
;  2174          3640  3                 [0] :
;  2175          3641  3                     COND_VAL_CHANGE = 0;                ! none of the above, don't translate
;  2176          3642  3
;  2177          3643  3                 [1] :
;  2178          3644  4                     BEGIN                               ! SQRT(N), N LSS 0
;  2179          3645  4                     COND_VAL = BAS$$COND_VAL (BAS$K_IMASQUROO);
;  2180          3646  4                     COND_VAL_CHANGE = 1;
;  2181          3647  3                     END;
;  2182          3648  3
;  2183          3649  3                 [2] :
;  2184          3650  4                     BEGIN                               ! LOG(N), N LEQ 0
;  2185          3651  4                     COND_VAL = BAS$$COND_VAL (BAS$K_ILLARGLOG);
;  2186          3652  4                     COND_VAL_CHANGE = 1;
;  2187          3653  3                     END;
;  2188          3654  3
;  2189          3655  3                 [4, 5, 14, 18] :
;  2190          3656  4                     BEGIN                               ! N/0, N%/0 and $QUO(N$,'0',P%)
;  2191          3657  4                     COND_VAL = BAS$$COND_VAL (BAS$K_DIVBY_ZER);
```

```
;  2192          3658  4                COND_VAL_CHANGE = 1;
;  2193          3659  3                END;
;  2194          3660  3
;  2195          3661  3            [7] :
;  2196          3662  4                BEGIN                                   ! integer overflow
;  2197          3663  4                COND_VAL = BAS$$COND_VAL (BAS$K_INTERR);
;  2198          3664  4                COND_VAL_CHANGE = 1;
;  2199          3665  3                END;
;  2200          3666  3
;  2201          3667  3            [10] :
;  2202          3668  4                BEGIN                                   ! invalid address
;  2203          3669  4                COND_VAL = BAS$$COND_VAL (BAS$K_MEMMANVIO);
;  2204          3670  4                COND_VAL_CHANGE = 1;
;  2205          3671  3                END;
;  2206          3672  3
;  2207          3673  3            [11] :
;  2208          3674  4                BEGIN                                   ! reserved operand--fixup to zero
;  2209          3675  4
;  2210          3676  4                LOCAL
;  2211          3677  4                    FIXUP_RESULT;
;  2212          3678  4
;  2213          3679  4                FIXUP_RESULT = LIB$FIXUP_FLT (.SIGNAL_ARGS, .MECHANISM_ARGS);
;  2214          3680  4 !+
;  2215          3681  4 ! If the fixup attempt fails, then this handler fails.
;  2216          3682  4 !-
;  2217          3683  4
;  2218          3684  5                IF ( NOT .FIXUP_RESULT)
;  2219          3685  4                THEN
;  2220          3686  5                    BEGIN
;  2221          3687  5                    GONE_BACK = (IF (.TOP_LEVEL) THEN 0 ELSE 1);
;  2222          3688  5                    RETURN (.FIXUP_RESULT);
;  2223          3689  4                    END;
;  2224          3690  4
;  2225          3691  4 !+
;  2226          3692  4 ! Convert the signal to the catch-all floating point warning for BASIC.
;  2227          3693  4 !-
;  2228          3694  4                COND_VAL = BAS$$COND_VAL (BAS$K_FLOPOIERR);
;  2229          3695  4                COND_VAL_CHANGE = 1;
;  2230          3696  3                END;
;  2231          3697  3
;  2232          3698  3            [3, 6, 8, 9, 19] :
;  2233          3699  4                BEGIN                                   ! all other math errors
;  2234          3700  4                COND_VAL = BAS$$COND_VAL (BAS$K_FLOPOIERR);
;  2235          3701  4                COND_VAL_CHANGE = 1;
;  2236          3702  3                END;
;  2237          3703  3
;  2238          3704  3            [12] :
;  2239          3705  4                BEGIN                                   ! subscript out of range
;  2240          3706  4                COND_VAL = BAS$$COND_VAL (BAS$K_SUBOUTRAN);
;  2241          3707  4                COND_VAL_CHANGE = 1;
;  2242          3708  3                END;
;  2243          3709  3
;  2244          3710  3            [13] :
;  2245          3711  4                BEGIN                                   ! String package ran out of memory
;  2246          3712  4                COND_VAL = BAS$$COND_VAL (BAS$K_MAXMEMEXC);
;  2247          3713  4                COND_VAL_CHANGE = 1;
;  2248          3714  3                END;
```

```
;  2249          3715  3
;  2250          3716  3                [15] :
;  2251          3717  4                    BEGIN
;  2252          3718  4            !+
;  2253          3719  4            ! I/O continued to closed file.  This happens when a function is called in
;  2254          3720  4            ! an I/O list (for example, to evaluate a subscript) and the function
;  2255          3721  4            ! closes the I/O channel that the I/O list is operating on.  For compatability
;  2256          3722  4            ! with BASIC-PLUS, give the error message ILLEGAL BYTE COUNT FOR I/O.
;  2257          3723  4            !-
;  2258          3724  4                    COND_VAL = BAS$$COND_VAL (BAS$K_ILLBYTCOU);
;  2259          3725  4                    COND_VAL_CHANGE = 1;
;  2260          3726  3                    END;
;  2261          3727  3
;  2262          3728  3                [16] :
;  2263          3729  4                    BEGIN                               ! Unwinding through this frame
;  2264          3730  4            !+
;  2265          3731  4            ! We are unwinding through this frame.  This may be due to a RESUME
;  2266          3732  4            ! statement cutting back this frame, to the RUN command recovering
;  2267          3733  4            ! from an error, or to a non-BASIC part of the user's program doing
;  2268          3734  4            ! error recovery.  Deallocate any heap storage held by this frame.
;  2269          3735  4            !-
;  2270          3736  4                    BAS$$UNWIND (.FMP);
;  2271          3737  3                    END;
;  2272          3738  3
;  2273          3739  3                [17] :
;  2274          3740  4                    BEGIN                               ! String created longer than 65535 characters
;  2275          3741  4                    COND_VAL = BAS$$COND_VAL (BAS$K_STRTOOLON);
;  2276          3742  4                    COND_VAL_CHANGE = 1;
;  2277          3743  3                    END;
;  2278          3744  3
;  2279          3745  3                [20] :                                  ! Decimal overflow
;  2280          3746  4                    BEGIN
;  2281          3747  4                    COND_VAL = BAS$$COND_VAL (BAS$K_DECERR);
;  2282          3748  4                    COND_VAL_CHANGE = 1;
;  2283          3749  3                    END;
;  2284          3750  3
;  2285          3751  3                TES;
;  2286          3752  3
;  2287          3753  3            !+
;  2288          3754  3            ! If the translated signal condition is not a BASIC condition,
;  2289          3755  3            ! we can't process it.  Return to CHF and indicate that the next
;  2290          3756  3            ! higher frame should be given a chance at it.
;  2291          3757  3            !-
;  2292          3758  3
;  2293          3759  4            IF (.COND_VAL [STS$V_FAC_NO] NEQ BAS$K_FAC_NO)
;  2294          3760  3            THEN
;  2295          3761  4                BEGIN
;  2296          3762  4                GONE_BACK = (IF (.TOP_LEVEL) THEN 0 ELSE 1);
;  2297          3763  4                RETURN (SS$_RESIGNAL);
;  2298          3764  3                END;
;  2299          3765  3
;  2300          3766  2            END;
;  2301          3767  2
;  2302          3768  2            !+
;  2303          3769  2            ! Give the user's BASIC program a chance to process the error.
;  2304          3770  2            ! If it succeeds, give a success return, otherwise do a special
;  2305          3771  2            ! resignal by extending the signal argument list and signaling again.
```

```
;  2306          3772  2 !-
;  2307          3773  2
;  2308          3774  2     IF (( NOT .SYSTEM_ERROR) AND (.COND_VAL [STS$V_SEVERITY] NEQU STS$K_INFO))
;  2309          3775  2     THEN
;  2310          3776  2         USER_RESULT = BAS$$USER_HAND (.COND_VAL [STS$V_CODE], .FMP, .MECHANISM_ARGS [CHF$L_MCH_DEPTH])
;  2311          3777  2     ELSE
;  2312          3778  2         USER_RESULT = USER_HAND_FAIL;
;  2313          3779  2
;  2314          3780  2 !+
;  2315          3781  2 ! If the user processes the error to his own satisfaction, skip
;  2316          3782  2 ! most of the remainder of this handler.
;  2317          3783  2 !-
;  2318          3784  2
;  2319          3785  2     IF (.USER_RESULT NEQ USER_HAND_CONT)
;  2320          3786  2     THEN
;  2321          3787  2         BEGIN
;  2322          3788  2 !+
;  2323          3789  2 ! If the user specified system handling, set the flag so that the
;  2324          3790  2 ! deeper levels of BAS$HANDLER won't call BAS$$USER_HAND.
;  2325          3791  2 !-
;  2326          3792  2
;  2327          3793  2         IF (.USER_RESULT EQL USER_HAND_FAIL) THEN SYSTEM_ERROR = 1;
;  2328          3794  2
;  2329          3795  2 !+
;  2330          3796  2 ! If we are at the top level purge the terminal's output buffer so
;  2331          3797  2 ! that, if a message is printed, it will print after the program's
;  2332          3798  2 ! output.
;  2333          3799  2 !-
;  2334          3800  2
;  2335          3801  2         IF (.TOP_LEVEL) THEN BAS$$PUR_IO_ERR ();
;  2336          3802  2
;  2337          3803  3 !<BLF/PAGE>
```

```
;  2339     3804  3 !+
;  2340     3805  3 ! Append a message about the current frame to the signal argument
;  2341     3806  3 ! list.  This requires recopying the list.  If we have translated
;  2342     3807  3 ! the signal condition, append the new condition rather than
;  2343     3808  3 ! overwrite the old one, so that a message like 'floating point error'
;  2344     3809  3 ! can have with it a clue as to why it happened.
;  2345     3810  3 !
;  2346     3811  3 ! Compute the length of the new signal argument list.
;  2347     3812  3 !-
;  2348     3813  3         LEN_VECTOR = (.SIGNAL_ARGS [CHF$L_SIG_ARGS] + 3);
;  2349     3814  3
;  2350     3815  3         CASE .FMP [BSF$B_PROC_CODE] FROM BSF$K_PROC_MAIN TO BSF$K_PROC_IOL OF
;  2351     3816  3             SET
;  2352     3817  3
;  2353     3818  3             [BSF$K_PROC_MAIN, BSF$K_PROC_SUB, BSF$K_PROC_EXTF] :
;  2354     3819  3 !+
;  2355     3820  3 ! These frames only have two variables in the FAO list
;  2356     3821  3 !-
;  2357     3822  3                 LEN_VECTOR = .LEN_VECTOR + 2;
;  2358     3823  3
;  2359     3824  3             [BSF$K_PROC_DEF, BSF$K_PROC_DEFS, BSF$K_PROC_GOSB, BSF$K_PROC_ONER] :
;  2360     3825  3 !+
;  2361     3826  3 ! These frames have three variables in the FAO list
;  2362     3827  3 !-
;  2363     3828  3                 LEN_VECTOR = .LEN_VECTOR + 3;
;  2364     3829  3
;  2365     3830  3             [BSF$K_PROC_IOL] :
;  2366     3831  3 !+
;  2367     3832  3 ! This frame has only one variable in the FAO list
;  2368     3833  3 !-
;  2369     3834  3                 LEN_VECTOR = .LEN_VECTOR + 1;
;  2370     3835  3
;  2371     3836  3             [OUTRANGE] :
;  2372     3837  3 !+
;  2373     3838  3 ! If the BSF$B_PROC_CODE byte is out of range then the frame
;  2374     3839  3 ! has been garbaged.  There is no point in attempting to continue,
;  2375     3840  3 ! so we mearly return to CHF.  It is likely that some error message
;  2376     3841  3 ! will be printed.
;  2377     3842  3 !-
;  2378     3843  4                 BEGIN
;  2379     3844  4                 GONE_BACK = (IF (.TOP_LEVEL) THEN 0 ELSE 1);
;  2380     3845  4                 RETURN (SS$_RESIGNAL);
;  2381     3846  3                 END;
;  2382     3847  3             TES;
;  2383     3848  3
;  2384     3849  3 !+
;  2385     3850  3 ! Take into account translation of a math error and adding a FAO count
;  2386     3851  3 !  to a short list.
;  2387     3852  3 !-
;  2388     3853  3
;  2389     3854  4         IF (.COND_VAL_CHANGE)
;  2390     3855  3         THEN
;  2391     3856  4             BEGIN
;  2392     3857  4             LEN_VECTOR = .LEN_VECTOR + 6;
;  2393     3858  4             END
;  2394     3859  3         ELSE
;  2395     3860  3
```

```
;  2396     3861  3              IF (.SIGNAL_ARGS [CHF$L_SIG_ARGS] EQL 3) THEN LEN_VECTOR = .LEN_VECTOR + 1;
;  2397     3862
;  2398     3863  !+
;  2399     3864  ! If the argument list is too long, quit.  This should only happen if
;  2400     3865  ! there is a tall stack of subroutines.
;  2401     3866  !-
;  2402     3867
;  2403     3868          IF (.LEN_VECTOR GTR 250)
;  2404     3869          THEN
;  2405     3870              BEGIN
;  2406     3871              GONE_BACK = (IF (.TOP_LEVEL) THEN 0 ELSE 1);
;  2407     3872              RETURN (SS$_RESIGNAL);
;  2408     3873              END;
;  2409     3874
;  2410     3875  !+
;  2411     3876  ! Get space to hold the new signal argument list.
;  2412     3877  !-
;  2413     3878
;  2414     3879          IF ( NOT (GET_VM_RESULT = LIB$GET_VM (%REF (.LEN_VECTOR*%UPVAL), NEW_VECTOR)))
;  2415     3880          THEN
;  2416     3881  !+
;  2417     3882  ! If we are out of space just quit.  This should happen only for
;  2418     3883  ! very unreasonable BASIC programs.  The BASIC program is given
;  2419     3884  ! no chance to recover.
;  2420     3885  !-
;  2421     3886  4               BEGIN
;  2422     3887  4               LIB$STOP (.GET_VM_RESULT);
;  2423     3888  4               GONE_BACK = (IF (.TOP_LEVEL) THEN 0 ELSE 1);
;  2424     3889  4               RETURN (SS$_RESIGNAL);
;  2425     3890  3               END;
;  2426     3891
;  2427     3892  !+
;  2428     3893  ! Now copy data into the new vector.  If we have not translated
;  2429     3894  ! the signal condition then our new data goes between the last
;  2430     3895  ! of the BASIC data and the first non-BASIC data.  If we have
;  2431     3896  ! translated the signal condition then our data goes first.
;  2432     3897  !
;  2433     3898  ! First set the length.  Don't count the count longword or the two
;  2434     3899  ! trailing longwords.
;  2435     3900  !-
;  2436     3901          NEW_VECTOR [0, 0, %BPVAL, 1] = .LEN_VECTOR - 3;
;  2437     3902          PUTTER = 1;
;  2438     3903          GETTER = 1;
;  2439     3904  !+
;  2440     3905  ! If we translated the signal code, store it and a 0 for its FAO count.
;  2441     3906  ! Also, store a special message which prints the original PC and PSL.
;  2442     3907  !-
;  2443     3908
;  2444     3909  4           IF (.COND_VAL_CHANGE)
;  2445     3910  3           THEN
;  2446     3911  4               BEGIN
;  2447     3912  4               NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = .COND_VAL;
;  2448     3913  4               PUTTER = .PUTTER + 1;
;  2449     3914  4               NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 0;
;  2450     3915  4               PUTTER = .PUTTER + 1;
;  2451     3916  4               NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_PCPSL);
;  2452     3917  4                                                           ! user PC=!XL, PS[=!XL
```

```
;  2453          3918  4              PUTTER = .PUTTER + 1;
;  2454          3919  4              NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 2;      ! FAO count
;  2455          3920  4              PUTTER = .PUTTER + 1;
;  2456          3921  4              NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] =         !
;  2457          3922  4              .SIGNAL_ARGS [(.SIGNAL_ARGS [CHF$L_SIG_ARGS] - 1)*%UPVAL, 0, %BPVAL, 0];    ! PC
;  2458          3923  4              PUTTER = .PUTTER + 1;
;  2459          3924  4              NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] =         !
;  2460          3925  4              .SIGNAL_ARGS [(.SIGNAL_ARGS [CHF$L_SIG_ARGS])*%UPVAL, 0, %BPVAL, 0];        ! PSL
;  2461          3926  4              PUTTER = .PUTTER + 1;
;  2462          3927  4              END
;  2463          3928  3          ELSE
;  2464          3929  4              BEGIN
;  2465          3930  4      !+
;  2466          3931  4      ! Otherwise copy all the BASIC data.
;  2467          3932  4      !-
;  2468          3933  4              SCAN_DONE = 0;
;  2469          3934  4
;  2470          3935  4              UNTIL (.SCAN_DONE) DO
;  2471          3936  5                  BEGIN
;  2472          3937  5                  TEMP_COND_VAL = .SIGNAL_ARGS [.GETTER*%UPVAL, 0, %BPVAL, 0];
;  2473          3938  5
;  2474          3939  6                  IF (.TEMP_COND_VAL [STS$V_FAC_NO] NEQ BAS$K_FAC_NO)
;  2475          3940  5                  THEN
;  2476          3941  5                      SCAN_DONE = 1
;  2477          3942  5                  ELSE
;  2478          3943  6                      BEGIN
;  2479          3944  6                      GETTER = .GETTER + 1;
;  2480          3945  6                      NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = .TEMP_COND_VAL;
;  2481          3946  6                      PUTTER = .PUTTER + 1;
;  2482          3947  6      !+
;  2483          3948  6      ! Copy the FAO arguments, unless we have reached the end of the list
;  2484          3949  6      !-
;  2485          3950  6
;  2486          3951  7                      IF (.GETTER NEQU (.SIGNAL_ARGS [CHF$L_SIG_ARGS] - 1))
;  2487          3952  6                      THEN
;  2488          3953  7                          BEGIN
;  2489          3954  7                          NUM_FAO_ARGS = .SIGNAL_ARGS [.GETTER*%UPVAL, 0, %BPVAL, 0];
;  2490          3955  7                          GETTER = .GETTER + 1;
;  2491          3956  7                          NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = .NUM_FAO_ARGS;
;  2492          3957  7                          PUTTER = .PUTTER + 1;
;  2493          3958  7
;  2494          3959  7                          INCR COUNTER FROM 1 TO .NUM_FAO_ARGS DO
;  2495          3960  8                              BEGIN
;  2496          3961  8                              NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] =         !
;  2497          3962  8                              .SIGNAL_ARGS [.GETTER*%UPVAL, 0, %BPVAL, 0];
;  2498          3963  8                              GETTER = .GETTER + 1;
;  2499          3964  8                              PUTTER = .PUTTER + 1;
;  2500          3965  7                              END;
;  2501          3966  7
;  2502          3967  7                          END
;  2503          3968  6                      ELSE
;  2504          3969  7                          BEGIN
;  2505          3970  7      !+
;  2506          3971  7      ! We have reached the end of the list, finding a BASIC condition there.
;  2507          3972  7      ! Insert a zero FAO argument count since we will be adding more
;  2508          3973  7      ! condition values.
;  2509          3974  7      !-
```

```
;  2510          3975  7                        NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 0;
;  2511          3976  7                        PUTTER = .PUTTER + 1;
;  2512          3977  6                        END;
;  2513          3978  6
;  2514          3979  6 !+
;  2515          3980  6 ! Check for the end of the signal arguments.
;  2516          3981  6 !-
;  2517          3982  6
;  2518          3983  6                    IF (.GETTER EQLU (.SIGNAL_ARGS [CHF$L_SIG_ARGS] - 1)) THEN SCAN_DONE = 1;
;  2519          3984  6
;  2520          3985  5                    END;
;  2521          3986  5
;  2522          3987  4                END;
;  2523          3988  4
;  2524          3989  3            END;
;  2525          3990  3
;  2526          3991  3 !+
;  2527          3992  3 ! Now put our data in the parameter list we are building.
;  2528          3993  3 ! This data varies depending on the frame type.
;  2529          3994  3 !-
;  2530          3995  3
;  2531          3996  3        CASE .FMP [BSF$B_PROC_CODE] FROM BSF$K_PROC_MAIN TO BSF$K_PROC_IOL OF
;  2532          3997  3            SET
;  2533          3998  3
;  2534          3999  3            [BSF$K_PROC_MAIN] :                 ! main program
;  2535          4000  4                BEGIN
;  2536          4001  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_MAIN);
;  2537          4002  4                                                ! message code
;  2538          4003  4                PUTTER = .PUTTER + 1;
;  2539          4004  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 2;  ! number of FAO arguments
;  2540          4005  4                PUTTER = .PUTTER + 1;
;  2541          4006  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$LINE (.FMP);   ! current line number
;  2542          4007  4                PUTTER = .PUTTER + 1;
;  2543          4008  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$MODULE (.FMP); ! module name
;  2544          4009  4                PUTTER = .PUTTER + 1;
;  2545          4010  3                END;
;  2546          4011  3
;  2547          4012  3            [BSF$K_PROC_SUB] :                  ! external subroutine
;  2548          4013  4                BEGIN
;  2549          4014  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_SUB);
;  2550          4015  4                                                ! message code
;  2551          4016  4                PUTTER = .PUTTER + 1;
;  2552          4017  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 2;  ! number of FAO arguments
;  2553          4018  4                PUTTER = .PUTTER + 1;
;  2554          4019  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$LINE (.FMP);   ! current line number
;  2555          4020  4                PUTTER = .PUTTER + 1;
;  2556          4021  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$MODULE (.FMP); ! module name
;  2557          4022  4                PUTTER = .PUTTER + 1;
;  2558          4023  3                END;
;  2559          4024  3
;  2560          4025  3            [BSF$K_PROC_EXTF] :                 ! external function
;  2561          4026  4                BEGIN
;  2562          4027  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_EXTF);
;  2563          4028  4                                                ! message code
;  2564          4029  4                PUTTER = .PUTTER + 1;
;  2565          4030  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 2;  ! number of FAO arguments
;  2566          4031  4                PUTTER = .PUTTER + 1;
```

```
;  2567          4032  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$LINE (.FMP);   ! current line number
;  2568          4033  4                PUTTER = .PUTTER + 1;
;  2569          4034  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$MODULE (.FMP); ! module name
;  2570          4035  4                PUTTER = .PUTTER + 1;
;  2571          4036  3                END;
;  2572          4037  3
;  2573          4038  3            [BSF$K_PROC_DEF] :                    ! DEF procedure
;  2574          4039  4                BEGIN
;  2575          4040  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_DEF);
;  2576          4041  4                                                  ! message code
;  2577          4042  4                PUTTER = .PUTTER + 1;
;  2578          4043  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 3;  ! number of FAO arguments
;  2579          4044  4                PUTTER = .PUTTER + 1;
;  2580          4045  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$LINE (.FMP);   ! current line number
;  2581          4046  4                PUTTER = .PUTTER + 1;
;  2582          4047  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$FUNCTION (.FMP);       ! function name
;  2583          4048  4                PUTTER = .PUTTER + 1;
;  2584          4049  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$MODULE (.FMP); ! module name
;  2585          4050  4                PUTTER = .PUTTER + 1;
;  2586          4051  3                END;
;  2587          4052  3
;  2588          4053  3            [BSF$K_PROC_DEFS] :                   ! DEF* procedure
;  2589          4054  4                BEGIN
;  2590          4055  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_DEFS);
;  2591          4056  4                                                  ! message code
;  2592          4057  4                PUTTER = .PUTTER + 1;
;  2593          4058  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 3;  ! number of FAO arguments
;  2594          4059  4                PUTTER = .PUTTER + 1;
;  2595          4060  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$LINE (.FMP);   ! current line number
;  2596          4061  4                PUTTER = .PUTTER + 1;
;  2597          4062  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$FUNCTION (.FMP);       ! function name
;  2598          4063  4                PUTTER = .PUTTER + 1;
;  2599          4064  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$MODULE (.FMP); ! module name
;  2600          4065  4                PUTTER = .PUTTER + 1;
;  2601          4066  3                END;
;  2602          4067  3
;  2603          4068  3            [BSF$K_PROC_GOSB] :                   ! GOSUB
;  2604          4069  4                BEGIN
;  2605          4070  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_GOSB);
;  2606          4071  4                                                  ! message code
;  2607          4072  4                PUTTER = .PUTTER + 1;
;  2608          4073  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 3;  ! number of FAO arguments
;  2609          4074  4                PUTTER = .PUTTER + 1;
;  2610          4075  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$LINE (.FMP);   ! current line number
;  2611          4076  4                PUTTER = .PUTTER + 1;
;  2612          4077  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$FUNCTION (.FMP);       ! function number
;  2613          4078  4                PUTTER = .PUTTER + 1;
;  2614          4079  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$MODULE (.FMP); ! module name
;  2615          4080  4                PUTTER = .PUTTER + 1;
;  2616          4081  3                END;
;  2617          4082  3
;  2618          4083  3            [BSF$K_PROC_ONER] :                   ! ON ERROR GOTO
;  2619          4084  4                BEGIN
;  2620          4085  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_ONER);
;  2621          4086  4                                                  ! message code
;  2622          4087  4                PUTTER = .PUTTER + 1;
;  2623          4088  4                NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 3;  ! number of FAO arguments
```

```
;  2624          4089  4                    PUTTER = .PUTTER + 1;
;  2625          4090  4                    NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$LINE (.FMP);   ! current line number
;  2626          4091  4                    PUTTER = .PUTTER + 1;
;  2627          4092  4                    NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$FUNCTION (.FMP);       ! function number
;  2628          4093  4                    PUTTER = .PUTTER + 1;
;  2629          4094  4                    NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$MODULE (.FMP); ! module name
;  2630          4095  4                    PUTTER = .PUTTER + 1;
;  2631          4096  3                    END;
;  2632          4097  3
;  2633          4098  3                [BSF$K_PROC_IOL] :                  ! Immediate mode code
;  2634          4099  4                    BEGIN
;  2635          4100  4                    NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$COND_VAL (ERR_TRACE_IOLST);
;  2636          4101  4                                                    ! message code
;  2637          4102  4                    PUTTER = .PUTTER + 1;
;  2638          4103  4                    NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = 1;  ! number of FAO arguments
;  2639          4104  4                    PUTTER = .PUTTER + 1;
;  2640          4105  4                    NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = BAS$$MODULE (.FMP); ! module name
;  2641          4106  4                    PUTTER = .PUTTER + 1;
;  2642          4107  3                    END;
;  2643          4108  3
;  2644          4109  3                [OUTRANGE] :
;  2645          4110  3            !+
;  2646          4111  3            ! If the BSF$B_PROC_CODE byte is out of range then the frame
;  2647          4112  3            ! has been garbaged.  There is no point in attempting to continue,
;  2648          4113  3            ! so we mearly return to CHF.  It is likely that some error message
;  2649          4114  3            ! will be printed.
;  2650          4115  3            !-
;  2651          4116  4                    BEGIN
;  2652          4117  4                    GONE_BACK = (IF (.TOP_LEVEL) THEN 0 ELSE 1);
;  2653          4118  4                    RETURN (SS$_RESIGNAL);
;  2654          4119  3                    END;
;  2655          4120  3                TES;
;  2656          4121  3
;  2657          4122  3            !+
;  2658          4123  3            ! Copy the remainder of the signal argument list, omitting the
;  2659          4124  3            ! PC and PSL unless we converted the signal condition.
;  2660          4125  3            !-
;  2661          4126  3                    COPY_LIMIT = .SIGNAL_ARGS [CHF$L_SIG_ARGS];
;  2662          4127  3
;  2663          4128  3                    IF ( NOT .COND_VAL_CHANGE) THEN COPY_LIMIT = .COPY_LIMIT - 2;
;  2664          4129  3
;  2665          4130  3                    WHILE (.GETTER LEQ .COPY_LIMIT) DO
;  2666          4131  4                        BEGIN
;  2667          4132  4                        NEW_VECTOR [.PUTTER*%UPVAL, 0, %BPVAL, 0] = .SIGNAL_ARGS [.GETTER*%UPVAL, 0, %BPVAL, 0];
;  2668          4133  4                        GETTER = .GETTER + 1;
;  2669          4134  4                        PUTTER = .PUTTER + 1;
;  2670          4135  3                        END;
;  2671          4136  3
;  2672          4137  3            !+
;  2673          4138  3            ! Set the severity code of the condition which we will signal depending
;  2674          4139  3            ! on the system handling of that signal.  However, if this is not an
;  2675          4140  3            ! I/O error from a terminal, don't restart.
;  2676          4141  3            !-
;  2677          4142  3                    RESTART_IO_FLAG = 0;
;  2678          4143  3
;  2679          4144  3                    CASE (.ERR_SYSTEM [.COND_VAL [STS$V_CODE]]) FROM K_SYS_CONT TO K_SYS_RESTART OF
;  2680          4145  3                        SET
```

```
;  2681          4146  3
;  2682          4147  3                [K_SYS_CONT] :
;  2683          4148  4                    BEGIN
;  2684          4149  4 !+
;  2685          4150  4 ! If the severity is INFO, don't promote it to WARNING.
;  2686          4151  4 !-
;  2687          4152  4
;  2688          4153  5                    IF (.COND_VAL [STS$V_SEVERITY] NEQ STS$K_INFO)  !
;  2689          4154  4                    THEN
;  2690          4155  4                        COND_VAL [STS$V_SEVERITY] = STS$K_WARNING;
;  2691          4156  4
;  2692          4157  3                    END;
;  2693          4158  3
;  2694          4159  3                [K_SYS_EXIT] :
;  2695          4160  4                    BEGIN
;  2696          4161  4 !+
;  2697          4162  4 ! If the severity is INFO, don't promote it to SEVERE.
;  2698          4163  4 !-
;  2699          4164  4
;  2700          4165  5                    IF (.COND_VAL [STS$V_SEVERITY] NEQ STS$K_INFO)  !
;  2701          4166  4                    THEN
;  2702          4167  4                        COND_VAL [STS$V_SEVERITY] = STS$K_SEVERE;
;  2703          4168  4
;  2704          4169  3                    END;
;  2705          4170  3
;  2706          4171  3                [K_SYS_RESTART] :
;  2707          4172  4                    BEGIN
;  2708          4173  4
;  2709          4174  5                    IF (LIB$MATCH_COND ((SIGNAL_ARGS [CHF$L_SIG_NAME] + (2*%UPVAL)), %REF (BAS$_ON_CHAFIL)))
;  2710          4175  4                    THEN
;  2711          4176  5                        BEGIN
;  2712          4177  5 !+
;  2713          4178  5 ! Because the error code is followed by BAS$_ON_CHAFIL the signal must
;  2714          4179  5 ! have been from BAS$$SIGNAL_IO, so this must be an I/O error.
;  2715          4180  5 ! If the I/O is to a terminal, the I/O statement can be restarted.
;  2716          4181  5 !-
;  2717          4182  5                        GLOBAL REGISTER
;  2718          4183  5                            CCB = K_CCB_REG : REF BLOCK [,BYTE];
;  2719          4184  5
;  2720          4185  5                        CCB = .OTS$$A_CUR_LUB;
;  2721          4186  5
;  2722          4187  6                        IF (OTS$$TERM_IO () OR
;  2723          4188  6                            .CCB [LUB$V_ANSI])
;  2724          4189  5                        THEN
;  2725          4190  5                            RESTART_IO_FLAG = 1;
;  2726          4191  5
;  2727          4192  4                        END;
;  2728          4193  4
;  2729          4194  5                    IF (.RESTART_IO_FLAG)
;  2730          4195  4                    THEN
;  2731          4196  4                        COND_VAL [STS$V_SEVERITY] = STS$K_WARNING
;  2732          4197  4                    ELSE
;  2733          4198  4                        COND_VAL [STS$V_SEVERITY] = STS$K_SEVERE;
;  2734          4199  4
;  2735          4200  3                    END;
;  2736          4201  3            TES;
;  2737          4202  3
```

```
; 2738          4203  3          NEW_VECTOR [1*%UPVAL, 0, %BPVAL, 0] = .COND_VAL;
; 2739          4204  3 !<BLF/PAGE>
```

```
; 2741          4205  3 !+
; 2742          4206  3 ! Now call LIB$SIGNAL with the argument list we have built.  It will
; 2743          4207  3 ! invoke this routine recursively for each active frame in the
; 2744          4208  3 ! user's BASIC program.  Intermediate levels in other languages will
; 2745          4209  3 ! be skipped over (provided that the other handlers do not intercept
; 2746          4210  3 ! BASIC error codes--if they do, they presumably know what they are
; 2747          4211  3 ! doing).  The signal argument list includes a traceback in the user's
; 2748          4212  3 ! terms.
; 2749          4213  3 !-
; 2750          4214  3
; 2751          4215  5         IF ((.ERR_SYSTEM [.COND_VAL [STS$V_CODE]] EQL K_SYS_EXIT)       !
; 2752          4216  5             AND (.SYSTEM_ERROR)                                         !
; 2753          4217  4             AND (.COND_VAL [STS$V_SEVERITY] NEQ STS$K_INFO))
; 2754          4218  3         THEN
; 2755          4219  3             CALLG (.NEW_VECTOR, LIB$STOP)
; 2756          4220  3         ELSE
; 2757          4221  3             CALLG (.NEW_VECTOR, LIB$SIGNAL);
; 2758          4222  3
; 2759          4223  3 !+
; 2760          4224  3 ! If we get here the condition is being continued.
; 2761          4225  3 ! Either system handling is not being called for, or the system handling
; 2762          4226  3 ! is not "EXIT".
; 2763          4227  3 !-
; 2764          4228  3         LIB$FREE_VM (%REF (.LEN_VECTOR*%UPVAL), NEW_VECTOR);
; 2765          4229  3         LEN_VECTOR = 0;
; 2766          4230  3 !+
; 2767          4231  3 ! If this error is restartable (as determined above) and we are at the
; 2768          4232  3 ! top level (that is, the level at which the I/O statement was executed)
; 2769          4233  3 ! and no unwind is called for (that is, the user has not executed an
; 2770          4234  3 ! error handler, since all error handlers end with a RESUME, which
; 2771          4235  3 ! causes an unwind) then restart the I/O statement.
; 2772          4236  3 !-
; 2773          4237  3
; 2774          4238  4         IF (.TOP_LEVEL)
; 2775          4239  3         THEN
; 2776          4240  4             BEGIN
; 2777          4241  4
; 2778          4242  5             IF ((.RESTART_IO_FLAG) AND (.UNWIND_COUNT EQL 0))
; 2779          4243  4             THEN
; 2780          4244  5                 BEGIN
; 2781          4245  5 !+
; 2782          4246  5 ! Unwind back to the beginning of the caller's I/O statement.
; 2783          4247  5 ! This cannot be done directly, because we don't know where the
; 2784          4248  5 ! beginning of the I/O statement is, so we call BAS$$RESTART_IO
; 2785          4249  5 ! which puts the I/O system back to the way it was when the I/O
; 2786          4250  5 ! list started, and then restarts the I/O list.  The call is done
; 2787          4251  5 ! through RESTART_IO to get SP restored properly.
; 2788          4252  5 !-
; 2789          4253  5                     SYS$UNWIND (MECHANISM_ARGS [CHF$L_MCH_DEPTH], RESTART_IO);
; 2790          4254  5 !+
; 2791          4255  5 ! Since we have taken care of this error, clear the error flag.
; 2792          4256  5 !-
; 2793          4257  5                     BAS$L_ERRFLG = 0;
; 2794          4258  5                     END
; 2795          4259  5
; 2796          4260  3                 END;
; 2797          4261  3
```

```
;  2798          4262  2          END;                                    ! of user does not handle the error
;  2799          4263  2
;  2800          4264  2      !+
;  2801          4265  2      ! If either this level or a deeper level has requested that the top level
;  2802          4266  2      ! do an unwind, and if this is the top level, do the unwind.
;  2803          4267  2      !-
;  2804          4268  2
;  2805          4269  3          IF (.TOP_LEVEL AND (.UNWIND_COUNT NEQ 0))
;  2806          4270  3          THEN
;  2807          4271  3              BEGIN
;  2808          4272  3              SYS$UNWIND (UNWIND_COUNT, RESTART);
;  2809          4273  3              UNWIND_COUNT = 0;
;  2810          4274  2              END;
;  2811          4275  2
;  2812          4276  2      !+
;  2813          4277  2      ! Set GONE_BACK and SYSTEM_ERROR (own cells) for the previous level
;  2814          4278  2      ! of BAS$HANDLER.
;  2815          4279  2      !-
;  2816          4280  2          GONE_BACK = (IF (.TOP_LEVEL) THEN 0 ELSE 1);
;  2817          4281  2
;  2818          4282  2          IF (.TOP_LEVEL) THEN SYSTEM_ERROR = 0;
;  2819          4283  2
;  2820          4284  2          RETURN (SS$_CONTINUE);
;  2821          4285  1          END;                                    ! of BAS$$HANDLER


                                                0FFC 00000          .ENTRY  BAS$$HANDLER, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 3487
                                                                            R10,R11
                                      5E    A4  AE  9E 00002          MOVAB   -92(SP), SP                                   ;
                                      54    AE  7C  00006          CLRQ    NEW_VECTOR                                    ; 3533
                                      6D  051D  CF  DE 00009          MOVAL   82$, (FP)                                     ;
                                      51        5D  D0 0000E          MOVL    FP, FRAME                                     ; 3587
                                      55    04  AC  D0 00011          MOVL    SIGNAL_ARGS, R5                               ; 3588
                                      50        65  D0 00015          MOVL    (R5), R0                                      ;
                                      01        50  D1 00018          CMPL    R0, #1                                        ;
                                                05  15 0001B          BLEQ    1$                                            ;
                                  04  A1      6540  F7 0001D          CVTLW   (R5)[R0], 4(FRAME)                            ; 3590
                                      52    08  AC  D0 00022 1$:      MOVL    MECHANISM_ARGS, R2                            ; 3596
                                      54    04  A2  D0 00026          MOVL    4(R2), FMP                                    ;
                                      04 00000000'  EF  E9 0002A      BLBC    GONE_BACK, 2$                                 ; 3602
                                                5A  D4 00031          CLRL    TOP_LEVEL                                     ;
                                                03  11 00033          BRB     3$                                            ;
                                      5A        01  D0 00035 2$:      MOVL    #1, TOP_LEVEL                                 ;
                              00000000'  EF     01  D0 00038 3$:      MOVL    #1, GONE_BACK                                 ; 3603
                                  50    AE    04  A5 D0 0003F          MOVL    4(R5), COND_VAL                               ; 3608
00000000G  8F       52    AE            0C      00  ED 00044          CMPZV   #0, #12, COND_VAL+2, #BAS$K_FAC_NO            ; 3610
                                                05  12 0004E          BNEQ    4$                                            ;
                                                56  D4 00050          CLRL    COND_VAL_CHANGE                               ; 3612
                                              0183  31 00052          BRW     24$                                           ;
                                  4C    AE  04A4    8F  3C 00055 4$:  MOVZWL  #1188, 76(SP)                                 ; 3636
                                      4C    AE      9F  0005B          PUSHAB  76(SP)                                        ;
                                  4C    AE  04B4    8F  3C 0005E      MOVZWL  #1204, 76(SP)                                 ; 3635
                                      4C    AE      9F  00064          PUSHAB  76(SP)                                        ;
                                  4C    AE  04BC    8F  3C 00067      MOVZWL  #1212, 76(SP)                                 ; 3634
```

```
                        4C   AE  9F 0006D          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 00070    MOVL    #STR$_STRTOOLON, 76(SP)                                        ; 3633
                        4C   AE  9F 00078          PUSHAB  76(SP)                                                          ;
                4C   AE      0920   8F  3C 0007B    MOVZWL  #2336, 76(SP)                                                  ; 3632
                        4C   AE  9F 00081          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 00084    MOVL    #OTS$_IO_CONCLO, 76(SP)                                        ; 3631
                        4C   AE  9F 0008C          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 0008F    MOVL    #STR$_DIVBY_ZER, 76(SP)                                        ; 3630
                        4C   AE  9F 00097          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 0009A    MOVL    #STR$_INSVIRMEM, 76(SP)                                        ; 3629
                        4C   AE  9F 000A2          PUSHAB  76(SP)                                                          ;
                4C   AE      04AC   8F  3C 000A5    MOVZWL  #1196, 76(SP)                                                  ; 3628
                        4C   AE  9F 000AB          PUSHAB  76(SP)                                                          ;
                4C   AE      0454   8F  3C 000AE    MOVZWL  #1108, 76(SP)                                                  ; 3627
                        4C   AE  9F 000B4          PUSHAB  76(SP)                                                          ;
                4C   AE             0C  D0 000B7    MOVL    #12, 76(SP)                                                    ; 3626
                        4C   AE  9F 000BB          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 000BE    MOVL    #MTH$_UNDEXP, 76(SP)                                           ; 3625
                        4C   AE  9F 000C6          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 000C9    MOVL    #MTH$_SIGLOSMAT, 76(SP)                                        ; 3624
                        4C   AE  9F 000D1          PUSHAB  76(SP)                                                          ;
                4C   AE      047C   8F  3C 000D4    MOVZWL  #1148, 76(SP)                                                  ; 3623
                        4C   AE  9F 000DA          PUSHAB  76(SP)                                                          ;
                4C   AE      048C   8F  3C 000DD    MOVZWL  #1164, 76(SP)                                                  ; 3622
                        4C   AE  9F 000E3          PUSHAB  76(SP)                                                          ;
                4C   AE      0484   8F  3C 000E6    MOVZWL  #1156, 76(SP)                                                  ; 3621
                        4C   AE  9F 000EC          PUSHAB  76(SP)                                                          ;
                4C   AE      0494   8F  3C 000EF    MOVZWL  #1172, 76(SP)                                                  ; 3620
                        4C   AE  9F 000F5          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 000F8    MOVL    #MTH$_FLOOVEMAT, 76(SP)                                        ; 3619
                        4C   AE  9F 00100          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 00103    MOVL    #MTH$_LOGZERNEG, 76(SP)                                        ; 3618
                        4C   AE  9F 0010B          PUSHAB  76(SP)                                                          ;
                4C   AE 00000000G   8F  D0 0010E    MOVL    #MTH$_SQUROONEG, 76(SP)                                        ; 3617
                        4C   AE  9F 00116          PUSHAB  76(SP)                                                          ;
                        F4   AD  9F 00119          PUSHAB  COND_VAL                                                        ; 3616
            00000000G   00       15  FB 0011C    CALLS   #21,_LIB$MATCH_COND
          14            00       50  CF 00123    CASEL   R0, #0, #20
006C      0034          002E     002A    00127 5$: .WORD   6$-5$,-
0040      006C          003A     003A    0012F             7$-5$,-
004C      0046          006C     006C    00137             8$-5$,-
007E      003A          0078     0072    0013F             15$-5$,-
006C      003A          008F     0084    00147             9$-5$,-
                                 0095    0014F             9$-5$,-
                                                           15$-5$,-
                                                           10$-5$,-
                                                           15$-5$,-
                                                           15$-5$,-
                                                           11$-5$,-
                                                           12$-5$,-
                                                           16$-5$,-
                                                           17$-5$,-
                                                           9$-5$,-
                                                           18$-5$,-
                                                           19$-5$,-
                                                           20$-5$,-
                                                           9$-5$,-
```

```
                                                                           15$-5$,-                          ;
                                                                           21$-5$                            ;
                                             56  D4 00151 6$:     CLRL     COND_VAL_CHANGE                   ; 3641
                                             77  11 00153         BRB      23$                               ;
                          7E          00G    8F  9A 00155 7$:     MOVZBL   #BAS$K_IMASQUROO, -(SP)           ; 3645
                                             65  11 00159         BRB      22$                               ;
                          7E          00G    8F  9A 0015B 8$:     MOVZBL   #BAS$K_ILLARGLOG, -(SP)           ; 3651
                                             5F  11 0015F         BRB      22$                               ;
                          7E          00G    8F  9A 00161 9$:     MOVZBL   #BAS$K_DIVBY_ZER, -(SP)           ; 3657
                                             59  11 00165         BRB      22$                               ;
                          7E          00G    8F  9A 00167 10$:    MOVZBL   #BAS$K_INTERR, -(SP)              ; 3663
                                             53  11 0016B         BRB      22$                               ;
                          7E          00G    8F  9A 0016D 11$:    MOVZBL   #BAS$K_MEMMANVIO, -(SP)           ; 3669
                                             4D  11 00171         BRB      22$                               ;
                                             52  DD 00173 12$:    PUSHL    R2                                ; 3679
                                             55  DD 00175         PUSHL    R5                                ;
                 00000000G  00               02  FB 00177         CALLS    #2, LIB$FIXUP_FLT                 ;
                            12               50  E8 0017E         BLBS     FIXUP_RESULT, 15$                 ; 3684
                            04               5A  E9 00181         BLBC     TOP_LEVEL, 13$                    ; 3687
                                             51  D4 00184         CLRL     R1                                ;
                                             03  11 00186         BRB      14$                               ;
                            51               01  D0 00188 13$:    MOVL     #1, R1                            ;
                 00000000'  EF               51  D0 0018B 14$:    MOVL     R1, GONE_BACK                     ;
                                                 04 00192         RET                                        ; 3688
                          7E          00G    8F  9A 00193 15$:    MOVZBL   #BAS$K_FLOPOIERR, -(SP)           ; 3700
                                             27  11 00197         BRB      22$                               ;
                          7E          00G    8F  9A 00199 16$:    MOVZBL   #BAS$K_SUBOUTRAN, -(SP)           ; 3706
                                             21  11 0019D         BRB      22$                               ;
                          7E          00G    8F  9A 0019F 17$:    MOVZBL   #BAS$K_MAXMEMEXC, -(SP)           ; 3712
                                             1B  11 001A3         BRB      22$                               ;
                          7E          00G    8F  9A 001A5 18$:    MOVZBL   #BAS$K_ILLBYTCOU, -(SP)           ; 3724
                                             15  11 001A9         BRB      22$                               ;
                                             54  DD 001AB 19$:    PUSHL    FMP                               ; 3736
                 00000000G  00               01  FB 001AD         CALLS    #1, BAS$$UNWIND                   ;
                                             16  11 001B4         BRB      23$                               ; 3616
                          7E          00G    8F  9A 001B6 20$:    MOVZBL   #BAS$K_STRTOOLON, -(SP)           ; 3741
                                             04  11 001BA         BRB      22$                               ;
                          7E          00G    8F  9A 001BC 21$:    MOVZBL   #BAS$K_DECERR, -(SP)              ; 3747
                     F9D9  CF                01  FB 001C0 22$:    CALLS    #1, BAS$$COND_VAL                 ;
                       50  AE                50  D0 001C5         MOVL     R0, COND_VAL                      ;
                           56                01  D0 001C9         MOVL     #1, COND_VAL_CHANGE               ; 3748
00000000G  8F     52  AE   0C                00  ED 001CC 23$:    CMPZV    #0, #12, COND_VAL+2, #BAS$K_FAC_NO ; 3759
                                             5E  12 001D6         BNEQ     31$                               ;
                           1A  00000000'     EF  E8 001D8 24$:    BLBS     SYSTEM_ERROR, 25$                 ; 3774
           03     50  AE   03                00  ED 001DF         CMPZV    #0, #3, COND_VAL, #3              ;
                                             12  13 001E5         BEQL     25$                               ;
                                      08     A2  DD 001E7         PUSHL    8(R2)                             ; 3776
                                             54  DD 001EA         PUSHL    FMP                               ;
           7E     58  AE   0C                03  EF 001EC         EXTZV    #3, #12, COND_VAL, -(SP)          ;
                     FADF  CF                03  FB 001F2         CALLS    #3, BAS$$USER_HAND                ;
                                             03  11 001F7         BRB      26$                               ;
                            50               02  D0 001F9 25$:    MOVL     #2, USER_RESULT                   ; 3778
                                             50  D5 001FC 26$:    TSTL     USER_RESULT                       ; 3785
                                             03  12 001FE         BNEQ     27$                               ;
                                           02E7  31 00200         BRW      77$                               ;
                            02               50  D1 00203 27$:    CMPL     USER_RESULT, #2                   ; 3793
                                             07  12 00206         BNEQ     28$                               ;
```

```
                00000000'  EF              01  D0 00208          MOVL    #1, SYSTEM_ERROR                                    ;
                           07               5A  E9 0020F 28$:     BLBC    TOP_LEVEL, 29$                                      ; 3801
                00000000G  00               00  FB 00212          CALLS   #0, BAS$$PUR_IO_ERR                                 ;
                           53               65  D0 00219 29$:     MOVL    (R5), R3                                            ; 3813
                      58   AE         03   A3  9E 0021C          MOVAB   3(R3), LEN_VECTOR                                   ;
                     07    01       E5   A4  8F 00221          CASEB   -27(FMP), #1, #7                                    ; 3815
              0018   0012      0012        0012 00226 30$:     .WORD   32$-30$,-                                           ;
              001E   0018      0018        0018 0022E                  32$-30$,-                                           ;
                                                                        32$-30$,-                                           ;
                                                                        33$-30$,-                                           ;
                                                                        33$-30$,-                                           ;
                                                                        33$-30$,-                                           ;
                                                                        33$-30$,-                                           ;
                                                                        34$-30$                                             ;
                                               49  11 00236 31$:     BRB     38$                                         ; 3844
                      58   AE                 02  C0 00238 32$:     ADDL2   #2, LEN_VECTOR                              ; 3822
                                               09  11 0023C          BRB     35$                                         ;
                      58   AE                 03  C0 0023E 33$:     ADDL2   #3, LEN_VECTOR                              ; 3828
                                               03  11 00242          BRB     35$                                         ;
                                         58    AE  D6 00244 34$:     INCL    LEN_VECTOR                                  ; 3834
                           06                  56  E9 00247 35$:     BLBC    COND_VAL_CHANGE, 36$                        ; 3854
                      58   AE                 06  C0 0024A          ADDL2   #6, LEN_VECTOR                              ; 3857
                                               08  11 0024E          BRB     37$                                         ; 3854
                           03                  53  D1 00250 36$:     CMPL    R3, #3                                      ; 3861
                                               03  12 00253          BNEQ    37$                                         ;
                                         58    AE  D6 00255          INCL    LEN_VECTOR                                  ;
                00000FA   8F              58    AE  D1 00258 37$:     CMPL    LEN_VECTOR, #250                            ; 3868
                                               1F  14 00260          BGTR    38$                                         ;
                                         54    AE  9F 00262          PUSHAB  NEW_VECTOR                                  ; 3879
          50    AE        5C    AE              02  78 00265          ASHL    #2, LEN_VECTOR, 80(SP)                      ;
                                         50    AE  9F 0026B          PUSHAB  80(SP)                                      ;
                00000000G  00               02  FB 0026E          CALLS   #2, LIB$GET_VM                              ;
                           0C               50  E8 00275          BLBS    GET_VM_RESULT, 39$                          ;
                                               50  DD 00278          PUSHL   GET_VM_RESULT                               ;
                00000000G  00               01  FB 0027A          CALLS   #1, LIB$STOP                                ; 3887
                                         00BC  31  00281 38$:     BRW     49$                                         ; 3888
          54    BE        58    AE              03  C3 00284 39$:     SUBL3   #3, LEN_VECTOR, @NEW_VECTOR                 ; 3901
                           5B                  01  D0 0028A          MOVL    #1, PUTTER                                  ; 3902
                           52                  01  D0 0028D          MOVL    #1, GETTER                                  ; 3903
                           39                  56  E9 00290          BLBC    COND_VAL_CHANGE, 40$                        ; 3909
                      54   BE4B          50    AE  D0 00293          MOVL    COND_VAL, @NEW_VECTOR[PUTTER]               ; 3912
                                               5B  D6 00299          INCL    PUTTER                                      ; 3913
                                         54    BE4B  D4 0029B          CLRL    @NEW_VECTOR[PUTTER]                         ; 3914
                                               5B  D6 0029F          INCL    PUTTER                                      ; 3915
                           7E                  8F  3C 002A1          MOVZWL  #4086, -(SP)                                ; 3916
                     F8F3  CF                  01  FB 002A6          CALLS   #1, BAS$$COND_VAL                           ;
                      54   BE4B                50  D0 002AB          MOVL    R0, @NEW_VECTOR[PUTTER]                     ;
                                               5B  D6 002B0          INCL    PUTTER                                      ; 3918
                      54   BE4B                02  D0 002B2          MOVL    #2, @NEW_VECTOR[PUTTER]                     ; 3919
                                               5B  D6 002B7          INCL    PUTTER                                      ; 3920
                      54   BE4B          FC    A543  D0 002B9          MOVL    -4(R5)[R3], @NEW_VECTOR[PUTTER]             ; 3922
                                               5B  D6 002C0          INCL    PUTTER                                      ; 3923
                      54   BE4B              6543  D0 002C2          MOVL    (R5)[R3], @NEW_VECTOR[PUTTER]               ; 3925
                                               5B  D6 002C8          INCL    PUTTER                                      ; 3926
                                               5F  11 002CA          BRB     47$                                         ; 3909
                                               57  D4 002CC 40$:     CLRL    SCAN_DONE                                   ; 3933
                           5A                  57  E8 002CE 41$:     BLBS    SCAN_DONE, 47$                              ; 3935
```

```
                        59          6542  D0 002D1          MOVL    (R5)[GETTER], TEMP_COND_VAL                     ; 3937
00000000G  8F           59            0C    ED 002D5          CMPZV   #16, #12, TEMP_COND_VAL, #BAS$K_FAC_NO  ; 3939
                                        46  12 002DE          BNEQ    46$
                                        52  D6 002E0          INCL    GETTER                                  ; 3944
                              54 BE4B       59  D0 002E2          MOVL    TEMP_COND_VAL, @NEW_VECTOR[PUTTER]      ; 3945
                                        5B  D6 002E7          INCL    PUTTER                                  ; 3946
                        50              5B    02  78 002E9          ASHL    #2, PUTTER, R0                          ; 3956
                                        51  FF  A3  9E 002ED          MOVAB   -1(R3), R1                      ; 3951
                                        51    52  D1 002F1          CMPL    GETTER, R1
                                        23  13 002F4          BEQL    44$
                                        58  6542  D0 002F6          MOVL    (R5)[GETTER], NUM_FAO_ARGS              ; 3954
                                        52  D6 002FA          INCL    GETTER                                  ; 3955
                                        50  54  AE  C0 002FC          ADDL2   NEW_VECTOR, R0                  ; 3956
                                        60    58  D0 00300          MOVL    NUM_FAO_ARGS, (R0)
                                        5B  D6 00303          INCL    PUTTER                                  ; 3957
                                        50  D4 00305          CLRL    COUNTER                                 ; 3959
                                        0A  11 00307          BRB     43$
                              54 BE4B       6542  D0 00309  42$:    MOVL    (R5)[GETTER], @NEW_VECTOR[PUTTER]       ; 3962
                                        52  D6 0030F          INCL    GETTER                                  ; 3963
                                        5B  D6 00311          INCL    PUTTER                                  ; 3964
                        F2              50    58  F3 00313  43$:    AOBLEQ  NUM_FAO_ARGS, COUNTER, 42$      ; 3959
                                        08  11 00317          BRB     45$                                     ; 3951
                                        50  54  AE  C0 00319  44$:    ADDL2   NEW_VECTOR, R0                  ; 3975
                                        60    58  D4 0031D          CLRL    (R0)
                                        5B  D6 0031F          INCL    PUTTER                                  ; 3976
                                        51    52  D1 00321  45$:    CMPL    GETTER, R1                      ; 3983
                                        A8  12 00324          BNEQ    41$
                                        57    01  D0 00326  46$:    MOVL    #1, SCAN_DONE
                                        A3  11 00329          BRB     41$                                     ; 3935
                        07              01  E5  A4  8F 0032B  47$:    CASEB   -27(FMP), #1, #7                ; 3996
            0056        0035          002E      0027      00330  48$:    .WORD   52$-48$,-
            009F        006B          0064      005D      00338           53$-48$,-
                                                                          54$-48$,-
                                                                          56$-48$,-
                                                                          57$-48$,-
                                                                          58$-48$,-
                                                                          59$-48$,-
                                                                          62$-48$
                                04          5A  E9 00340  49$:    BLBC    TOP_LEVEL, 50$                  ; 4117
                                        50  D4 00343          CLRL    R0
                                        03  11 00345          BRB     51$
                                50          01  D0 00347  50$:    MOVL    #1, R0
                        00000000'  EF       50  D0 0034A  51$:    MOVL    R0, GONE_BACK
                                50      0918    8F  3C 00351          MOVZWL  #2328, R0                       ; 4118
                                        04  00356          RET
                                7E      0FF9    8F  3C 00357  52$:    MOVZWL  #4089, -(SP)                    ; 4001
                                        0C  11 0035C          BRB     55$
                                7E      0FFA    8F  3C 0035E  53$:    MOVZWL  #4090, -(SP)                    ; 4014
                                        05  11 00363          BRB     55$
                                7E      0FFB    8F  3C 00365  54$:    MOVZWL  #4091, -(SP)                    ; 4027
                          F82F    CF            01  FB 0036A  55$:    CALLS   #1, BAS$$COND_VAL
                            54 BE4B             50  D0 0036F          MOVL    R0, @NEW_VECTOR[PUTTER]
                                                5B  D6 00374          INCL    PUTTER                          ; 4029
                            54 BE4B             02  D0 00376          MOVL    #2, @NEW_VECTOR[PUTTER]         ; 4030
                                                5B  D6 0037B          INCL    PUTTER                          ; 4031
                                                54  DD 0037D          PUSHL   FMP                             ; 4032
                          F89A    CF            01  FB 0037F          CALLS   #1, BAS$$LINE
```

```
                                       42  11 00384        BRB     61$                                          ;
                       7E      0FFC    8F  3C 00386  56$:  MOVZWL  #4092, -(SP)                                 ; 4040
                                       13  11 0038B        BRB     60$                                          ;
                       7E      0FFD    8F  3C 0038D  57$:  MOVZWL  #4093, -(SP)                                 ; 4055
                                       0C  11 00392        BRB     60$                                          ;
                       7E      0FFE    8F  3C 00394  58$:  MOVZWL  #4094, -(SP)                                 ; 4070
                                       05  11 00399        BRB     60$                                          ;
                       7E      0FFF    8F  3C 0039B  59$:  MOVZWL  #4095, -(SP)                                 ; 4085
                  F7F9 CF              01  FB 003A0  60$:  CALLS   #1, BAS$$COND_VAL                            ;
                  54 BE4B              50  D0 003A5        MOVL    R0, @NEW_VECTOR[PUTTER]                      ;
                                       5B  D6 003AA        INCL    PUTTER                                       ; 4087
                  54 BE4B              03  D0 003AC        MOVL    #3, @NEW_VECTOR[PUTTER]                      ; 4088
                                       5B  D6 003B1        INCL    PUTTER                                       ; 4089
                                       54  DD 003B3        PUSHL   FMP                                          ; 4090
                  F864 CF              01  FB 003B5        CALLS   #1, BAS$$LINE                                ;
                  54 BE4B              50  D0 003BA        MOVL    R0, @NEW_VECTOR[PUTTER]                      ;
                                       5B  D6 003BF        INCL    PUTTER                                       ; 4091
                                       54  DD 003C1        PUSHL   FMP                                          ; 4092
                  F866 CF              01  FB 003C3        CALLS   #1, BAS$$FUNCTION                            ;
                  54 BE4B              50  D0 003C8  61$:  MOVL    R0, @NEW_VECTOR[PUTTER]                      ;
                                       16  11 003CD        BRB     63$                                          ; 4093
                       7E      0FF7    8F  3C 003CF  62$:  MOVZWL  #4087, -(SP)                                 ; 4100
                  F7C5 CF              01  FB 003D4        CALLS   #1, BAS$$COND_VAL                            ;
                  54 BE4B              50  D0 003D9        MOVL    R0, @NEW_VECTOR[PUTTER]                      ;
                                       5B  D6 003DE        INCL    PUTTER                                       ; 4102
                  54 BE4B              01  D0 003E0        MOVL    #1, @NEW_VECTOR[PUTTER]                      ; 4103
                                       5B  D6 003E5  63$:  INCL    PUTTER                                       ; 4104
                                       54  DD 003E7        PUSHL   FMP                                          ; 4105
                  F86E CF              01  FB 003E9        CALLS   #1, BAS$$MODULE                              ;
                  54 BE4B              50  D0 003EE        MOVL    R0, @NEW_VECTOR[PUTTER]                      ;
                                       5B  D6 003F3        INCL    PUTTER                                       ; 4106
                       50              53  D0 003F5        MOVL    R3, COPY_LIMIT                               ; 4126
                       03              56  E8 003F8        BLBS    COND_VAL_CHANGE, 64$                         ; 4128
                       50              02  C2 003FB        SUBL2   #2, COPY_LIMIT                               ;
                       50              52  D1 003FE  64$:  CMPL    GETTER, COPY_LIMIT                           ; 4130
                                       0C  14 00401        BGTR    65$                                          ;
                  54 BE4B            6542  D0 00403        MOVL    (R5)[GETTER], @NEW_VECTOR[PUTTER]            ; 4132
                                       52  D6 00409        INCL    GETTER                                       ; 4133
                                       5B  D6 0040B        INCL    PUTTER                                       ; 4134
                                       EF  11 0040D        BRB     64$                                          ; 4130
                                       53  D4 0040F  65$:  CLRL    RESTART_IO_FLAG                              ; 4142
50        50   AE      0C              03  EF 00411        EXTZV   #3, #12, COND_VAL, R0                        ; 4144
               02      01     F65A CF40    8F 00417        CASEB   ERR_SYSTEM[R0], #1, #2                       ;
             001A    0010            0006     0041E  66$:  .WORD   67$-66$,-                                    ;
                                                                   68$-66$,-                                    ;
                                                                   69$-66$                                      ;
03        50   AE      03              00  ED 00424  67$:  CMPZV   #0, #3, COND_VAL, #3                         ; 4153
                                       4C  13 0042A        BEQL    74$                                          ;
                                       3E  11 0042C        BRB     72$                                          ; 4155
03        50   AE      03              00  ED 0042E  68$:  CMPZV   #0, #3, COND_VAL, #3                         ; 4165
                                       42  13 00434        BEQL    74$                                          ;
                                       3A  11 00436        BRB     73$                                          ; 4167
                4C     AE 00000000G    8F  D0 00438  69$:  MOVL    #BAS$_ON_CHAFIL, 76(SP)                      ; 4174
                               4C      AE  9F 00440        PUSHAB  76(SP)                                       ;
                               0C      A5  9F 00443        PUSHAB  12(R5)                                       ;
          00000000G  00                02  FB 00446        CALLS   #2, LIB$MATCH_COND                           ;
                     19                50  E9 0044D        BLBC    R0, 71$                                      ;
```

```
                                5B 00000000G   00   D0 00450          MOVL    OTS$$A_CUR_LUB, CCB                          ; 4185
                00000000G       00           00   FB 00457          CALLS   #0, OTS$$TERM_IO                             ; 4187
                                05           50   E8 0045E          BLBS    R0, 70$                                      ;
                    03      A1  AB           04   E1 00461          BBC     #4, -95(CCB), 71$                            ; 4188
                                53           01   D0 00466 70$:     MOVL    #1, RESTART_IO_FLAG                          ; 4190
                                06           53   E9 00469 71$:     BLBC    RESTART_IO_FLAG, 73$                         ; 4194
                            50  AE           07   8A 0046C 72$:     BICB2   #7, COND_VAL                                 ; 4196
                                             06   11 00470          BRB     74$                                          ;
    50  AE              03                   00   F0 00472 73$:     INSV    #4, #0, #3, COND_VAL                         ; 4198
                                50       54  AE   D0 00478 74$:     MOVL    NEW_VECTOR, R0                               ; 4203
                            04  A0       50  AE   D0 0047C          MOVL    COND_VAL, 4(R0)                              ;
        50          50      AE               0C   03   EF 00481          EXTZV   #3, #12, COND_VAL, R0                        ; 4215
                                             02       F5EA CF40   91 00487          CMPB    ERR_SYSTEM[R0], #2                           ;
                                             19   12 0048D          BNEQ    75$                                          ;
                                12 00000000'     EF   E9 0048F          BLBC    SYSTEM_ERROR, 75$                            ; 4216
        03          50      AE               03   00   ED 00496          CMPZV   #0, #3, COND_VAL, #3                         ; 4217
                                             0A   13 0049C          BEQL    75$                                          ;
                00000000G       00       54  BE   FA 0049E          CALLG   @NEW_VECTOR, LIB$STOP                        ; 4219
                                             08   11 004A6          BRB     76$                                          ;
                00000000G       00       54  BE   FA 004A8 75$:     CALLG   @NEW_VECTOR, LIB$SIGNAL                      ; 4221
                                         54  AE   9F 004B0 76$:     PUSHAB  NEW_VECTOR                                   ; 4228
                            50  AE   5C   AE   02   78 004B3          ASHL    #2, LEN_VECTOR, 80(SP)                       ;
                                         50  AE   9F 004B9          PUSHAB  80(SP)                                       ;
                00000000G       00           02   FB 004BC          CALLS   #2, LIB$FREE_VM                              ;
                                         58  AE   D4 004C3          CLRL    LEN_VECTOR                                   ; 4229
                                4A           5A   E9 004C6          BLBC    TOP_LEVEL, 79$                               ; 4238
                                1E           53   E9 004C9          BLBC    RESTART_IO_FLAG, 77$                         ; 4242
                                    00000000'    EF   D5 004CC          TSTL    UNWIND_COUNT                                 ;
                                             16   12 004D2          BNEQ    77$                                          ;
                                        F7E9     CF   9F 004D4          PUSHAB  RESTART_IO                                   ; 4253
            7E      08          AC           08   C1 004D8          ADDL3   #8, MECHANISM_ARGS, -(SP)                    ;
                00000000G       00           02   FB 004DD          CALLS   #2, SYS$UNWIND                               ;
                                    00000000'    EF   D4 004E4          CLRL    BAS$L_ERRFLG                                 ; 4257
                                26           5A   E9 004EA 77$:     BLBC    TOP_LEVEL, 79$                               ; 4269
                                    00000000'    EF   D5 004ED          TSTL    UNWIND_COUNT                                 ;
                                             17   13 004F3          BEQL    78$                                          ;
                                        F7AD     CF   9F 004F5          PUSHAB  RESTART                                      ; 4272
                                    00000000'    EF   9F 004F9          PUSHAB  UNWIND_COUNT                                 ;
                00000000G       00           02   FB 004FF          CALLS   #2, SYS$UNWIND                               ;
                                    00000000'    EF   D4 00506          CLRL    UNWIND_COUNT                                 ; 4273
                                04           5A   E9 0050C 78$:     BLBC    TOP_LEVEL, 79$                               ; 4280
                                             50   D4 0050F          CLRL    R0                                           ;
                                             03   11 00511          BRB     80$                                          ;
                                50           01   D0 00513 79$:     MOVL    #1, R0                                       ;
                00000000'       EF           50   D0 00516 80$:     MOVL    R0, GONE_BACK                                ;
                                06           5A   E9 0051D          BLBC    TOP_LEVEL, 81$                               ; 4282
                                    00000000'    EF   D4 00520          CLRL    SYSTEM_ERROR                                 ;
                                50           01   D0 00526 81$:     MOVL    #1, R0                                       ; 4284
                                             04   00529          RET                                                  ; 4285
                                         0000    0052A 82$:     .WORD   Save nothing                                 ; 3533
                                50       08  AC   D0 0052C          MOVL    8(AP), R0                                    ;
                                50       04  A0   D0 00530          MOVL    4(R0), R0                                    ;
                                         F8  A0   9F 00534          PUSHAB  NEW_VECTOR                                   ;
                                         FC  A0   9F 00537          PUSHAB  LEN_VECTOR                                   ;
                                             02   DD 0053A          PUSHL   #2                                           ;
                                             5E   DD 0053C          PUSHL   SP                                           ;
                                7E       04  AC   7D 0053E          MOVQ    4(AP), -(SP)                                 ;
```

```
                        F725   CF          03  FB 00542          CALLS   #3, HANDLER_HANDLER                    ;
                                               04 00547          RET                                          ;

; Routine Size:  1352 bytes,    Routine Base:  _BAS$CODE + 068A

; 2822          4286  1
```

```
;  2824          4287  1 GLOBAL ROUTINE BAS$ERL =                          ! error line number
;  2825          4288  1
;  2826          4289  1 !++
;  2827          4290  1 ! FUNCTIONAL DESCRIPTION:
;  2828          4291  1 !
;  2829          4292  1 !       Return the line number on which the current error happened.
;  2830          4293  1 !       If there is no error in progress, return the line number of
;  2831          4294  1 !       the last error, or 0.
;  2832          4295  1 !
;  2833          4296  1 ! FORMAL PARAMETERS:
;  2834          4297  1 !
;  2835          4298  1 !       NONE
;  2836          4299  1 !
;  2837          4300  1 ! IMPLICIT INPUTS:
;  2838          4301  1 !
;  2839          4302  1 !       BAS$L_ERL
;  2840          4303  1 !
;  2841          4304  1 ! IMPLICIT OUTPUTS:
;  2842          4305  1 !
;  2843          4306  1 !       NONE
;  2844          4307  1 !
;  2845          4308  1 ! ROUTINE VALUE:
;  2846          4309  1 !
;  2847          4310  1 !       The line number, as a 32-bit binary value.
;  2848          4311  1 !
;  2849          4312  1 ! COMPLETION CODES:
;  2850          4313  1 !
;  2851          4314  1 !       NONE
;  2852          4315  1 !
;  2853          4316  1 ! SIDE EFFECTS:
;  2854          4317  1 !
;  2855          4318  1 !       NONE
;  2856          4319  1 !
;  2857          4320  1 !--
;  2858          4321  1
;  2859          4322  2     BEGIN
;  2860          4323  2     RETURN (.BAS$L_ERL);
;  2861          4324  1     END;                                            ! of BAS$ERL


                                        0000 00000          .ENTRY  BAS$ERL, Save nothing                   ; 4287
                              50 00000000'  EF   D0 00002   MOVL    BAS$L_ERL, R0                           ; 4323
                                            04 00009        RET                                             ; 4324

; Routine Size:  10 bytes,    Routine Base:  _BAS$CODE + 0BD2

;  2862          4325  1
```

```
;  2864          4326  1 GLOBAL ROUTINE BAS$ERR =                          ! error number
;  2865          4327  1
;  2866          4328  1 !++
;  2867          4329  1 ! FUNCTIONAL DESCRIPTION:
;  2868          4330  1 !
;  2869          4331  1 !       Return the number of the current error.
;  2870          4332  1 !       If there is no error in progress, return the number
;  2871          4333  1 !       of the last error, or 0.
;  2872          4334  1 !
;  2873          4335  1 ! FORMAL PARAMETERS:
;  2874          4336  1 !
;  2875          4337  1 !       NONE
;  2876          4338  1 !
;  2877          4339  1 ! IMPLICIT INPUTS:
;  2878          4340  1 !
;  2879          4341  1 !       BAS$L_ERR
;  2880          4342  1 !
;  2881          4343  1 ! IMPLICIT OUTPUTS:
;  2882          4344  1 !
;  2883          4345  1 !       NONE
;  2884          4346  1 !
;  2885          4347  1 ! ROUTINE VALUE:
;  2886          4348  1 !
;  2887          4349  1 !       The error number, as a 32-bit binary value.
;  2888          4350  1 !
;  2889          4351  1 ! COMPLETION CODES:
;  2890          4352  1 !
;  2891          4353  1 !       NONE
;  2892          4354  1 !
;  2893          4355  1 ! SIDE EFFECTS:
;  2894          4356  1 !
;  2895          4357  1 !       NONE
;  2896          4358  1 !
;  2897          4359  1 !--
;  2898          4360  1
;  2899          4361  2     BEGIN
;  2900          4362  2     RETURN (.BAS$L_ERR);
;  2901          4363  1     END;                                            ! of BAS$ERR



                                        0000 00000          .ENTRY  BAS$ERR, Save nothing                   ; 4326
                              50 00000000'  EF   D0 00002   MOVL    BAS$L_ERR, R0                           ; 4362
                                            04 00009        RET                                             ; 4363

; Routine Size:  10 bytes,    Routine Base:  _BAS$CODE + 0BDC

;  2902          4364  1
```

```
;  2904          4365  1 GLOBAL ROUTINE BAS$ERN (                      ! error module name
;  2905          4366  1                 DESCRIP                       ! where to write the name
;  2906          4367  1     ) =
;  2907          4368  1
;  2908          4369  1 !++
;  2909          4370  1 ! FUNCTIONAL DESCRIPTION:
;  2910          4371  1 !
;  2911          4372  1 !       Return the name of the module in which the current error
;  2912          4373  1 !       happened.  If there is no error in progress, return
;  2913          4374  1 !       the module name for the last error, or a zero-length string.
;  2914          4375  1 !
;  2915          4376  1 ! FORMAL PARAMETERS:
;  2916          4377  1 !
;  2917          4378  1 !       DESCRIP.wt.dx   A descriptor into which to write the name of
;  2918          4379  1 !                       the module.
;  2919          4380  1 !
;  2920          4381  1 ! IMPLICIT INPUTS:
;  2921          4382  1 !
;  2922          4383  1 !       BAS$T_ERN
;  2923          4384  1 !
;  2924          4385  1 ! IMPLICIT OUTPUTS:
;  2925          4386  1 !
;  2926          4387  1 !       NONE
;  2927          4388  1 !
;  2928          4389  1 ! COMPLETION CODES:
;  2929          4390  1 !
;  2930          4391  1 !       Same as for STR$COPY
;  2931          4392  1 !
;  2932          4393  1 ! SIDE EFFECTS:
;  2933          4394  1 !
;  2934          4395  1 !       Calls STR$COPY; if it fails, this routine never returns.
;  2935          4396  1 !
;  2936          4397  1 !--
;  2937          4398  1
;  2938          4399  2     BEGIN
;  2939          4400  2
;  2940          4401  2     LOCAL
;  2941          4402  2         COPY_STATUS;
;  2942          4403  2
;  2943          4404  2     COPY_STATUS = STR$COPY_DX (.DESCRIP, BAS$T_ERN);
;  2944          4405  2     RETURN (COPY_STATUS);
;  2945          4406  1     END;                                        ! of BAS$ERN



                                        0000 00000          .ENTRY  BAS$ERN, Save nothing                   ;  4365
                                5E      04   C2 00002          SUBL2   #4, SP                                  ;
                          00000000'     EF   9F 00005          PUSHAB  BAS$T_ERN                               ;  4404
                                04      AC   DD 0000B          PUSHL   DESCRIP                                 ;
                00000000G       00      02   FB 0000E          CALLS   #2, STR$COPY_DX                         ;
                                6E      50   D0 00015          MOVL    R0, COPY_STATUS                         ;
                                50      6E   9E 00018          MOVAB   COPY_STATUS, R0                         ;  4405
                                        04   04 0001B          RET                                             ;  4406

; Routine Size:  28 bytes,    Routine Base:  _BAS$CODE + 0BE6
```

```
; 2946          4407  1
```

```
;  2948     4408  1 GLOBAL ROUTINE BAS$ERT (                              ! error text
;  2949     4409  1         DESCRIP,                                      ! where to put text
;  2950     4410  1         ERRNO                                         ! error number
;  2951     4411  1     ) =
;  2952     4412  1
;  2953     4413  1 !++
;  2954     4414  1 ! FUNCTIONAL DESCRIPTION:
;  2955     4415  1 !
;  2956     4416  1 !       Return the text of an error message.  If the error number is
;  2957     4417  1 !       unreasonable, the result is undefined.  The first character of
;  2958     4418  1 !       the message indicates the severity: "?" is a fatal error,
;  2959     4419  1 !       "%" is a warning, and all other messages start with a space.
;  2960     4420  1 !
;  2961     4421  1 ! FORMAL PARAMETERS:
;  2962     4422  1 !
;  2963     4423  1 !       DESCRIP.wt.d    A descriptor into which to write the text of
;  2964     4424  1 !                       the message.
;  2965     4425  1 !       ERRNO.rl.v      The error number for which we want the text.
;  2966     4426  1 !
;  2967     4427  1 ! IMPLICIT INPUTS:
;  2968     4428  1 !
;  2969     4429  1 !       The system error message file, SYS$MESSAGE:SYSMSG.EXE
;  2970     4430  1 !
;  2971     4431  1 ! IMPLICIT OUTPUTS:
;  2972     4432  1 !
;  2973     4433  1 !       NONE
;  2974     4434  1 !
;  2975     4435  1 ! COMPLETION CODES:
;  2976     4436  1 !
;  2977     4437  1 !       Same as STR$CONCAT
;  2978     4438  1 !
;  2979     4439  1 ! SIDE EFFECTS:
;  2980     4440  1 !
;  2981     4441  1 !       Calls several system functions.  If any fail, this routine
;  2982     4442  1 !       never returns.
;  2983     4443  1 !
;  2984     4444  1 !--
;  2985     4445  1
;  2986     4446  2     BEGIN
;  2987     4447  2
;  2988     4448  2     LOCAL
;  2989     4449  2         CONCAT_RESULT,                                ! Status from STR$CONCAT
;  2990     4450  2         GETMSG_RESULT,                                ! remembers status of SYS$GETMSG
;  2991     4451  2         LOCAL_DESCRIP : BLOCK [8, BYTE],              ! message descriptor
;  2992     4452  2         Q_DESC : BLOCK [8, BYTE],                     ! Points to ?, % or space
;  2993     4453  2         Q_BUF : VECTOR [1, BYTE],                     ! Holds the ?, % or space
;  2994     4454  2         LOCAL_BUF : VECTOR [256, BYTE],               ! Buffer got SYS$GETMSG
;  2995     4455  2         DUMMY;                                        ! used to discard last value from SYS$GETMSG
;  2996     4456  2
;  2997     4457  2 !+
;  2998     4458  2 ! Set up the local descriptor.
;  2999     4459  2 !-
;  3000     4460  2     LOCAL_DESCRIP [DSC$W_LENGTH] = 256;
;  3001     4461  2     LOCAL_DESCRIP [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  3002     4462  2     LOCAL_DESCRIP [DSC$B_CLASS] = DSC$K_CLASS_S;
;  3003     4463  2     LOCAL_DESCRIP [DSC$A_POINTER] = LOCAL_BUF;
;  3004     4464  2 !+
```

```
;  3005          4465  2 ! Get the message text from SYS$MESSAGE:SYSMSG.EXE
;  3006          4466  2 !-
;  3007          4467  2     GETMSG_RESULT = SYS$GETMSG (BAS$$COND_VAL (.ERRNO), LOCAL_DESCRIP, LOCAL_DESCRIP, 1, DUMMY);
;  3008          4468  2
;  3009          4469  2     IF ( NOT .GETMSG_RESULT) THEN LIB$STOP (.GETMSG_RESULT);
;  3010          4470  2
;  3011          4471  2 !+
;  3012          4472  2 ! Copy the message text to the user's string, concatenating a ?, % or
;  3013          4473  2 ! space onto its front to indicate the severity of the error.
;  3014          4474  2 !-
;  3015          4475  2     Q_DESC [DSC$W_LENGTH] = 1;
;  3016          4476  2     Q_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  3017          4477  2     Q_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  3018          4478  2     Q_DESC [DSC$A_POINTER] = Q_BUF [0];
;  3019          4479  2     Q_BUF [0] =
;  3020          4480  2     BEGIN
;  3021          4481  2
;  3022          4482  2     CASE .ERR_SEVERITY [.ERRNO] FROM STS$K_WARNING TO STS$K_SEVERE OF
;  3023          4483  2         SET
;  3024          4484  2
;  3025          4485  2         [STS$K_WARNING] :
;  3026          4486  2             %C'%';
;  3027          4487  2
;  3028          4488  2         [STS$K_SEVERE, STS$K_ERROR] :
;  3029          4489  2             %C'?';
;  3030          4490  2
;  3031          4491  2         [INRANGE, OUTRANGE] :
;  3032          4492  2             %C' ';
;  3033          4493  2         TES
;  3034          4494  2
;  3035          4495  2     END;
;  3036          4496  2     CONCAT_RESULT = STR$CONCAT (.DESCRIP, Q_DESC, LOCAL_DESCRIP);
;  3037          4497  2     RETURN (.CONCAT_RESULT);
;  3038          4498  1     END;                                    ! of BAS$ERT


                                0000 00000          .ENTRY  BAS$ERT, Save nothing                          ;  4408
                     5E     FEE8 CE  9E 00002       MOVAB   -280(SP), SP                                   ;
                  F8 AD 010E0100 8F  D0 00007       MOVL    #17694976, LOCAL_DESCRIP                       ;  4460
                  FC AD       08 AE  9E 0000F       MOVAB   LOCAL_BUF, LOCAL_DESCRIP+4                     ;  4463
                                 5E  DD 00014       PUSHL   SP                                             ;  4467
                                 01  DD 00016       PUSHL   #1                                             ;
                                 F8  AD 9F 00018    PUSHAB  LOCAL_DESCRIP                                  ;
                                 F8  AD 9F 0001B    PUSHAB  LOCAL_DESCRIP                                  ;
                                 08  AC DD 0001E    PUSHL   ERRNO                                          ;
                   F600  CF          01  FB 00021    CALLS   #1, BAS$$COND_VAL                              ;
                                     50  DD 00026    PUSHL   R0                                             ;
              00000000G  00          05  FB 00028    CALLS   #5, SYS$GETMSG                                 ;
                         09          50  E8 0002F    BLBS    GETMSG_RESULT, 1$                              ;  4469
                                     50  DD 00032    PUSHL   GETMSG_RESULT                                  ;
              00000000G  00          01  FB 00034    CALLS   #1, LIB$STOP                                   ;
                   F0 AD 010E0001    8F  D0 0003B 1$: MOVL    #17694721, Q_DESC                              ;  4475
                   F4 AD       04    AE  9E 00043    MOVAB   Q_BUF, Q_DESC+4                                ;  4478
                         50  F3B2    CF  9E 00048    MOVAB   ERR_SEVERITY, R0                               ;  4482
```

```
                04           00        08 BC40  8F 0004D          CASEB    @ERRNO[R0], #0, #4                      :
     000A       0014         000A         000F     00053 2$:       .WORD    4$-2$,-                                 :
                                          0014     0005B                    3$-2$,-                                 :
                                                                            5$-2$,-                                 :
                                                                            3$-2$,-                                 :
                                                                            5$-2$                                   :
                             50              20 D0 0005D 3$:       MOVL     #32, R0                                 :
                                             08 11 00060           BRB      6$                                      :
                             50              25 D0 00062 4$:       MOVL     #37, R0                                 :
                                             03 11 00065           BRB      6$                                      :
                             50              3F D0 00067 5$:       MOVL     #63, R0                                 :
                        04   AE              50 90 0006A 6$:       MOVB     R0, Q_BUF                               : 4480
                                       F8    AD 9F 0006E           PUSHAB   LOCAL_DESCRIP                           : 4496
                                       F0    AD 9F 00071           PUSHAB   Q_DESC                                  :
                                       04    AC DD 00074           PUSHL    DESCRIP                                 :
                  00000000G  00              03 FB 00077           CALLS    #3, STR$CONCAT                          :
                                                04 0007E           RET                                              : 4498

; Routine Size:  127 bytes,    Routine Base:  _BAS$CODE + 0C02

; 3039          4499  1
```

```
;  3041          4500  1 GLOBAL ROUTINE BAS$ERROR (                    ! signal an error
;  3042          4501  1         ERRNO                                 ! error number
;  3043          4502  1     ) : NOVALUE =
;  3044          4503  1
;  3045          4504  1 !++
;  3046          4505  1 ! FUNCTIONAL DESCRIPTION:
;  3047          4506  1 !
;  3048          4507  1 !       Signals an error.  This is called from the compiled code when
;  3049          4508  1 !       an error condition is detected in line.  It can also be called
;  3050          4509  1 !       by user code directly.
;  3051          4510  1 !
;  3052          4511  1 ! FORMAL PARAMETERS:
;  3053          4512  1 !
;  3054          4513  1 !       ERRNO.rl.v      The error number which we want to signal.
;  3055          4514  1 !
;  3056          4515  1 ! IMPLICIT INPUTS:
;  3057          4516  1 !
;  3058          4517  1 !       NONE
;  3059          4518  1 !
;  3060          4519  1 ! IMPLICIT OUTPUTS:
;  3061          4520  1 !
;  3062          4521  1 !       NONE
;  3063          4522  1 !
;  3064          4523  1 ! ROUTINE VALUE:
;  3065          4524  1 !
;  3066          4525  1 !       NONE
;  3067          4526  1 !
;  3068          4527  1 ! COMPLETION CODES:
;  3069          4528  1 !
;  3070          4529  1 !       NONE
;  3071          4530  1 !
;  3072          4531  1 ! SIDE EFFECTS:
;  3073          4532  1 !
;  3074          4533  1 !       May never return to its caller.
;  3075          4534  1 !
;  3076          4535  1 !--
;  3077          4536  1
;  3078          4537  2     BEGIN
;  3079          4538  2 !+
;  3080          4539  2 ! Don't allow error numbers above 255, since we keep some internal
;  3081          4540  2 ! messages up there.
;  3082          4541  2 !-
;  3083          4542  2
;  3084          4543  2     IF (.ERRNO LEQU 255) THEN BAS$$SIGNAL (.ERRNO) ELSE BAS$$STOP (BAS$K_PROLOSSOR);
;  3085          4544  2
;  3086          4545  1     END;                                        ! of BAS$ERROR



                                0000 00000          .ENTRY  BAS$ERROR, Save nothing                 ; 4500
000000FF   8F       04    AC    D1 00002          CMPL    ERRNO, #255                             ; 4543
                          09    1A 0000A          BGTRU   1$                                      ;
                          04    AC    DD 0000C    PUSHL   ERRNO                                   ;
    F56B   CF             01    FB 0000F          CALLS   #1, BAS$$SIGNAL                         ;
                                04 00014          RET                                             ;
```

```
                          7E      00G  8F  9A 00015 1$:     MOVZBL  #BAS$K_PROLOSSOR, -(SP)                 ;
                    F575  CF           01  FB 00019         CALLS   #1, BA5$$STOP                           ;
                                       04  0001E           RET                                             ; 4545

; Routine Size:  31 bytes,    Routine Base:  _BAS$CODE + 0C81

; 3087          4546  1
```

```
;  3089          4547  1 GLOBAL ROUTINE BAS$$ERR_INIT : NOVALUE =        ! Initialize error flag
;  3090          4548  1
;  3091          4549  1 !++
;  3092          4550  1 ! FUNCTIONAL DESCRIPTION:
;  3093          4551  1 !
;  3094          4552  1 !       Initialize the error status.  This is used by the RUN command in case the user
;  3095          4553  1 !       does a RUN following an error.
;  3096          4554  1 !
;  3097          4555  1 ! FORMAL PARAMETERS:
;  3098          4556  1 !
;  3099          4557  1 !       NONE
;  3100          4558  1 !
;  3101          4559  1 ! IMPLICIT INPUTS:
;  3102          4560  1 !
;  3103          4561  1 !       NONE
;  3104          4562  1 !
;  3105          4563  1 ! IMPLICIT OUTPUTS:
;  3106          4564  1 !
;  3107          4565  1 !       All of the error cells that the user can see.
;  3108          4566  1 !
;  3109          4567  1 ! ROUTINE VALUE:
;  3110          4568  1 !
;  3111          4569  1 !       NONE
;  3112          4570  1 !
;  3113          4571  1 ! COMPLETION CODES:
;  3114          4572  1 !
;  3115          4573  1 !       NONE
;  3116          4574  1 !
;  3117          4575  1 ! SIDE EFFECTS:
;  3118          4576  1 !
;  3119          4577  1 !       After this routine has been called, we are not in an error handler.
;  3120          4578  1 !       Also, the error stack is empty.
;  3121          4579  1 !
;  3122          4580  1 !--
;  3123          4581  1
;  3124          4582  2     BEGIN
;  3125          4583  2 !+
;  3126          4584  2 ! First, empty the error stack.
;  3127          4585  2 !-
;  3128          4586  2
;  3129          4587  2     WHILE (.ERROR_STACK [0] NEQA .ERROR_STACK [1]) DO
;  3130          4588  2         BAS$POP_ERR ();
;  3131          4589  2
;  3132          4590  2 !+
;  3133          4591  2 ! Then make sure we are not in an error routine.
;  3134          4592  2 !-
;  3135          4593  2     BAS$L_ERRFLG = 0;
;  3136          4594  2 !+
;  3137          4595  2 ! See to it that all user-visible cells are in their initial states.
;  3138          4596  2 !-
;  3139          4597  2     BAS$A_CH_CUR_LN = 0;
;  3140          4598  2     BAS$L_GOING_BACK = 0;
;  3141          4599  2     SYSTEM_ERROR = 0;
;  3142          4600  2     GONE_BACK = 0;
;  3143          4601  2     BAS$L_ERL = 0;
;  3144          4602  2     BAS$L_ERR = 0;
;  3145          4603  2     BAS$T_ERN [DSC$W_LENGTH] = 0;
```

```
; 3146          4604  1     END;                                    ! of BAS$$ERR_INIT



                                                0004 00000          .ENTRY  BAS$$ERR_INIT, Save R2                  ; 4547
                                    52 00000000'  EF   9E 00002     MOVAB   ERROR_STACK, R2                         ;
                                          04    A2   62   D1 00009 1$: CMPL   ERROR_STACK, ERROR_STACK+4            ; 4587
                                                     07   13 0000D     BEQL    2$                                  ;
                                         0000V    CF   00   FB 0000F     CALLS   #0, BAS$POP_ERR                     ; 4588
                                                     F3   11 00014     BRB     1$                                  ;
                                                  0C    A2   7C 00016 2$: CLRQ    BAS$A_CH_CUR_LN                 ; 4597
                                                  F8    A2   7C 00019     CLRQ    SYSTEM_ERROR                    ; 4599
                                                  E0    A2   7C 0001C     CLRQ    BAS$L_ERL                       ; 4601
                                                  DC    A2   D4 0001F     CLRL    BAS$L_ERR                       ; 4602
                                                  D4    A2   B4 00022     CLRW    BAS$T_ERN                       ; 4603
                                                             04 00025     RET                                     ; 4604

; Routine Size:  38 bytes,    Routine Base:  _BAS$CODE + 0CA0

; 3147          4605  1
```

```
;  3149          4606  1 GLOBAL ROUTINE BAS$PUSH_ERR =                    ! Push error status
;  3150          4607  1
;  3151          4608  1 !++
;  3152          4609  1 ! FUNCTIONAL DESCRIPTION:
;  3153          4610  1 !
;  3154          4611  1 !       Save the error state on the error stack, so a BASIC program can
;  3155          4612  1 !       be run independent of the error flag.
;  3156          4613  1 !
;  3157          4614  1 ! CALLING SEQUENCE:
;  3158          4615  1 !
;  3159          4616  1 !       CALL BAS$PUSH_ERR ()
;  3160          4617  1 !
;  3161          4618  1 ! FORMAL PARAMETERS:
;  3162          4619  1 !
;  3163          4620  1 !       NONE
;  3164          4621  1 !
;  3165          4622  1 ! IMPLICIT INPUTS:
;  3166          4623  1 !
;  3167          4624  1 !       The OWN cells which represent the error status.
;  3168          4625  1 !
;  3169          4626  1 ! IMPLICIT OUTPUTS:
;  3170          4627  1 !
;  3171          4628  1 !       The error stack
;  3172          4629  1 !
;  3173          4630  1 ! SIDE EFFECTS:
;  3174          4631  1 !
;  3175          4632  1 !       Calls LIB$GET_VM to get virtual memory.
;  3176          4633  1 !--
;  3177          4634  1
;  3178          4635  2     BEGIN
;  3179          4636  2
;  3180          4637  2     BUILTIN
;  3181          4638  2         INSQUE;
;  3182          4639  2
;  3183          4640  2     LOCAL
;  3184          4641  2 !+
;  3185          4642  2 ! Declare the pointer to the block to push.
;  3186          4643  2 !-
;  3187          4644  2         PUSH : REF BLOCK [PUSH$K_LENGTH, BYTE] FIELD (PUSH_ITEM);
;  3188          4645  2
;  3189          4646  2 !+
;  3190          4647  2 ! If this is the first PUSH, initialize the queue.
;  3191          4648  2 !-
;  3192          4649  2
;  3193          4650  3     IF ( NOT .ERROR_STACK_INI)
;  3194          4651  3     THEN
;  3195          4652  3         BEGIN
;  3196          4653  3
;  3197          4654  3         LOCAL
;  3198          4655  3             AST_STATUS;
;  3199          4656  3
;  3200          4657  3         AST_STATUS = $SETAST (ENBFLG = 0);
;  3201          4658  3
;  3202          4659  4         IF ( NOT .ERROR_STACK_INI)
;  3203          4660  3         THEN
;  3204          4661  4             BEGIN
;  3205          4662  4             ERROR_STACK [0] = ERROR_STACK [1] = ERROR_STACK [0];
```

```
;  3206          4663  4              ERROR_STACK_INI = 1;
;  3207          4664                 END;
;  3208          4665
;  3209          4666           IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
;  3210          4667
;  3211          4668           END;
;  3212          4669
;  3213          4670  !+
;  3214          4671  ! Get virtual memory to hold the error state.
;  3215          4672  !-
;  3216          4673       BEGIN
;  3217          4674
;  3218          4675       LOCAL
;  3219          4676           GET_VM_RESULT;
;  3220          4677
;  3221          4678       GET_VM_RESULT = LIB$GET_VM (%REF (PUSH$K_LENGTH), PUSH);
;  3222          4679
;  3223          4680       IF ( NOT .GET_VM_RESULT) THEN BAS$$STOP (BAS$K_MAXMEMEXC);
;  3224          4681
;  3225          4682       END;
;  3226          4683  !+
;  3227          4684  ! Fill in.
;  3228          4685  !-
;  3229          4686       PUSH [PUSH$L_ERRFLG] = .BAS$L_ERRFLG;
;  3230          4687       CH$MOVE (8, BAS$T_ERN, PUSH [PUSH$T_ERN]);
;  3231          4688       PUSH [PUSH$L_ERR] = .BAS$L_ERR;
;  3232          4689       PUSH [PUSH$L_ERL] = .BAS$L_ERL;
;  3233          4690       PUSH [PUSH$L_HGH_LVL] = .HIGHEST_LEVEL;
;  3234          4691       PUSH [PUSH$A_HGH_FMP] = .HIGHEST_FMP;
;  3235          4692       PUSH [PUSH$L_ACC_LVL] = .ACCUM_LEVEL;
;  3236          4693       PUSH [PUSH$L_UNW_CNT] = .UNWIND_COUNT;
;  3237          4694       PUSH [PUSH$L_SYS_ERR] = .SYSTEM_ERROR;
;  3238          4695       PUSH [PUSH$L_GONE_BAK] = .GONE_BACK;
;  3239          4696       PUSH [PUSH$A_CUR_LIN] = .BAS$A_CH_CUR_LN;
;  3240          4697       PUSH [PUSH$L_GOING_BACK] = .BAS$L_GOING_BACK;
;  3241          4698       PUSH [PUSH$A_RESTART] = .BAS$A_RESTART;
;  3242          4699  !+
;  3243          4700  ! Put this item on the error stack.
;  3244          4701  !-
;  3245          4702       INSQUE (.PUSH, ERROR_STACK);
;  3246          4703  !+
;  3247          4704  ! Make sure there is no error outstanding.
;  3248          4705  !-
;  3249          4706       BAS$L_ERRFLG = 0;
;  3250          4707       BAS$A_CH_CUR_LN = 0;
;  3251          4708       BAS$L_GOING_BACK = 0;
;  3252          4709       SYSTEM_ERROR = 0;
;  3253          4710       GONE_BACK = 0;
;  3254          4711  !+
;  3255          4712  ! Successful completion.
;  3256          4713  !-
;  3257          4714       RETURN (SS$_NORMAL);
;  3258          4715  1    END;                                    ! of routine BAS$PUSH_ERR


                                                                  .EXTRN  SYS$SETAST
```

```
                                                01FC 00000          .ENTRY  BAS$PUSH_ERR, Save R2,R3,R4,R5,R6,R7,R8      ; 4606
                                    58 00000000G    00   9E 00002          MOVAB   SYS$SETAST, R8                       ;
                                    57 00000000'    EF   9E 00009          MOVAB   ERROR_STACK_INI, R7                  ;
                                    5E              08   C2 00010          SUBL2   #8, SP                               ;
                                    21              67   E8 00013          BLBS    ERROR_STACK_INI, 2$                  ; 4650
                                                    7E   D4 00016          CLRL    -(SP)                                ; 4657
                                    68              01   FB 00018          CALLS   #1, SYS$SETAST                       ;
                                    0F              67   E8 0001B          BLBS    ERROR_STACK_INI, 1$                  ; 4659
                                    51         F8   A7   9E 0001E          MOVAB   ERROR_STACK, R1                      ; 4662
                                FC  A7              51   D0 00022          MOVL    R1, ERROR_STACK+4                    ;
                                F8  A7              51   D0 00026          MOVL    R1, ERROR_STACK                      ;
                                    67              01   D0 0002A          MOVL    #1, ERROR_STACK_INI                  ; 4663
                                    09              50   D1 0002D 1$:      CMPL    AST_STATUS, #9                       ; 4666
                                                    05   12 00030          BNEQ    2$                                   ;
                                                    01   DD 00032          PUSHL   #1                                   ;
                                    68              01   FB 00034          CALLS   #1, SYS$SETAST                       ;
                                        04     AE   9F   00037 2$:         PUSHAB  PUSH                                 ; 4678
                                04  AE         40   8F   9A 0003A          MOVZBL  #64, 4(SP)                           ;
                                        04     AE   9F   0003F             PUSHAB  4(SP)                                ;
                            00000000G   00          02   FB 00042          CALLS   #2, LIB$GET_VM                       ;
                                        09          50   E8 00049          BLBS    GET_VM_RESULT, 3$                    ; 4680
                                        7E     00G  8F   9A 0004C          MOVZBL  #BAS$K_MAXMEMEXC, -(SP)              ;
                                F4F9    CF          01   FB 00050          CALLS   #1, BAS$$STOP                        ;
                                        56     04   AE   D0 00055 3$:      MOVL    PUSH, R6                             ; 4686
                                08      A6     DC   A7   D0 00059          MOVL    BAS$L_ERRFLG, 8(R6)                  ;
                        0C  A6          CC     A7   08   28 0005E          MOVC3   #8, BAS$T_ERN, 12(R6)                ; 4687
                                14      A6     D4   A7   7D 00064          MOVQ    BAS$L_ERR, 20(R6)                    ; 4688
                                1C      A6     E0   A7   7D 00069          MOVQ    HIGHEST_LEVEL, 28(R6)                ; 4690
                                24      A6     E8   A7   7D 0006E          MOVQ    ACCUM_LEVEL, 36(R6)                  ; 4692
                                2C      A6     F0   A7   7D 00073          MOVQ    SYSTEM_ERROR, 44(R6)                 ; 4694
                                34      A6     04   A7   7D 00078          MOVQ    BAS$A_CH_CUR_LN, 52(R6)              ; 4696
                                3C      A6     0C   A7   D0 0007D          MOVL    BAS$A_RESTART, 60(R6)                ; 4698
                                F8      A7              66   0E 00082          INSQUE  (R6), ERROR_STACK                    ; 4702
                                                DC  A7  D4   00086          CLRL    BAS$L_ERRFLG                         ; 4706
                                                04  A7  7C   00089          CLRQ    BAS$A_CH_CUR_LN                      ; 4707
                                                F0  A7  7C   0008C          CLRQ    SYSTEM_ERROR                         ; 4709
                                            50          01   D0 0008F          MOVL    #1, R0                               ; 4714
                                                        04   00092          RET                                          ; 4715

; Routine Size:  147 bytes,    Routine Base:  _BAS$CODE + 0CC6

; 3259          4716  1
```

```
;  3261          4717  1 GLOBAL ROUTINE BAS$POP_ERR =                    ! Pop error status
;  3262          4718  1
;  3263          4719  1 !++
;  3264          4720  1 ! FUNCTIONAL DESCRIPTION:
;  3265          4721  1 !
;  3266          4722  1 !       Restore the error state from the error stack.
;  3267          4723  1 !
;  3268          4724  1 ! CALLING SEQUENCE:
;  3269          4725  1 !
;  3270          4726  1 !       CALL BAS$POP_ERR ()
;  3271          4727  1 !
;  3272          4728  1 ! FORMAL PARAMETERS:
;  3273          4729  1 !
;  3274          4730  1 !       NONE
;  3275          4731  1 !
;  3276          4732  1 ! IMPLICIT INPUTS:
;  3277          4733  1 !
;  3278          4734  1 !       The error stack.
;  3279          4735  1 !
;  3280          4736  1 ! IMPLICIT OUTPUTS:
;  3281          4737  1 !
;  3282          4738  1 !       The OWN storage which represents the error state.
;  3283          4739  1 !
;  3284          4740  1 ! SIDE EFFECTS:
;  3285          4741  1 !
;  3286          4742  1 !       Calls LIB$FREE_VM to free virtual memory.
;  3287          4743  1 !--
;  3288          4744  1
;  3289          4745  2     BEGIN
;  3290          4746  2
;  3291          4747  2     BUILTIN
;  3292          4748  2         REMQUE;
;  3293          4749  2
;  3294          4750  2     LOCAL
;  3295          4751  2         PUSH : REF BLOCK [PUSH$K_LENGTH, BYTE] FIELD (PUSH_ITEM);
;  3296          4752  2
;  3297          4753  2 !+
;  3298          4754  2 ! Get an item off the error stack.  It had better be there.
;  3299          4755  2 !-
;  3300          4756  2
;  3301          4757  2     IF (REMQUE (.ERROR_STACK [0], PUSH)) THEN BAS$$STOP (BAS$K_PROLOSSOR);
;  3302          4758  2
;  3303          4759  2 !+
;  3304          4760  2 ! Copy the data from the stack into the OWN cells, thus reestablishing
;  3305          4761  2 ! the error environment at the time of the PUSH.
;  3306          4762  2 !-
;  3307          4763  2     BAS$L_ERRFLG = .PUSH [PUSH$L_ERRFLG];
;  3308          4764  2     CH$MOVE (8, PUSH [PUSH$T_ERN], BAS$T_ERN);
;  3309          4765  2     BAS$L_ERR = .PUSH [PUSH$L_ERR];
;  3310          4766  2     BAS$L_ERL = .PUSH [PUSH$L_ERL];
;  3311          4767  2     HIGHEST_LEVEL = .PUSH [PUSH$L_HGH_LVL];
;  3312          4768  2     HIGHEST_FMP = .PUSH [PUSH$A_HGH_FMP];
;  3313          4769  2     ACCUM_LEVEL = .PUSH [PUSH$L_ACC_LVL];
;  3314          4770  2     UNWIND_COUNT = .PUSH [PUSH$L_UNW_CNT];
;  3315          4771  2     SYSTEM_ERROR = .PUSH [PUSH$L_SYS_ERR];
;  3316          4772  2     GONE_BACK = .PUSH [PUSH$L_GONE_BAK];
;  3317          4773  2     BAS$A_CH_CUR_LN = .PUSH [PUSH$A_CUR_LIN];
```

```
;3318     4774  2     BAS$L_GOING_BACK = .PUSH [PUSH$L_GOING_BACK];
;3319     4775  2     BAS$A_RESTART = .PUSH [PUSH$A_RESTART];
;3320     4776  2 !+
;3321     4777  2 ! We are done with the item from error stack, free it.
;3322     4778  2 !-
;3323     4779  3     BEGIN
;3324     4780  3
;3325     4781  3     LOCAL
;3326     4782  3         FREE_VM_RESULT;
;3327     4783  3
;3328     4784  3     FREE_VM_RESULT = LIB$FREE_VM (%REF (PUSH$K_LENGTH), PUSH);
;3329     4785  3
;3330     4786  3     IF ( NOT .FREE_VM_RESULT) THEN BAS$$STOP (BAS$K_PROLOSSOR);
;3331     4787  3
;3332     4788  2     END;
;3333     4789  2     RETURN (SS$_NORMAL);
;3334     4790  1     END;                                    ! of routine BAS$POP_ERR



                                  00FC 00000          .ENTRY  BAS$POP_ERR, Save R2,R3,R4,R5,R6,R7          ; 4717
                57 00000000'   EF   9E 00002          MOVAB   ERROR_STACK, R7                              ;
                5E             08   C2 00009          SUBL2   #8, SP                                       ;
             04 AE        00   B7   0F 0000C          REMQUE  @ERROR_STACK, PUSH                           ; 4757
                               09   1C 00011          BVC     1$                                           ;
                7E        00G  8F   9A 00013          MOVZBL  #BAS$K_PROLOSSOR, -(SP)                      ;
           F49F CF             01   FB 00017          CALLS   #1, BAS$$STOP                                ;
                56        04   AE   D0 0001C 1$:      MOVL    PUSH, R6                                     ; 4763
             E4 A7        08   A6   D0 00020          MOVL    8(R6), BAS$L_ERRFLG                          ;
       D4    A7  0C   A6   08   28 00025          MOVC3   #8, 12(R6), BAS$T_ERN                        ; 4764
             DC A7        14   A6   7D 0002B          MOVQ    20(R6), BAS$L_ERR                            ; 4765
             E8 A7        1C   A6   7D 00030          MOVQ    28(R6), HIGHEST_LEVEL                        ; 4767
             F0 A7        24   A6   7D 00035          MOVQ    36(R6), ACCUM_LEVEL                          ; 4769
             F8 A7        2C   A6   7D 0003A          MOVQ    44(R6), SYSTEM_ERROR                         ; 4771
             0C A7        34   A6   7D 0003F          MOVQ    52(R6), BAS$A_CH_CUR_LN                      ; 4773
             14 A7        3C   A6   D0 00044          MOVL    60(R6), BAS$A_RESTART                        ; 4775
                          04   AE   9F 00049          PUSHAB  PUSH                                         ; 4784
             04 AE        40   8F   9A 0004C          MOVZBL  #64, 4(SP)                                   ;
                          04   AE   9F 00051          PUSHAB  4(SP)                                        ;
       00000000G  00              02   FB 00054          CALLS   #2, LIB$FREE_VM                              ;
                7E             09   50   E8 0005B          BLBS    FREE_VM_RESULT, 2$                           ; 4786
                7E        00G  8F   9A 0005E          MOVZBL  #BAS$K_PROLOSSOR, -(SP)                      ;
           F454 CF             01   FB 00062          CALLS   #1, BAS$$STOP                                ;
                50             01   D0 00067 2$:      MOVL    #1, R0                                       ; 4789
                               04 0006A            RET                                                  ; 4790

; Routine Size:  107 bytes,    Routine Base:  _BAS$CODE + 0D59

;3335     4791  1
;3336     4792  1 END
;3337     4793  1
;3338     4794  0 ELUDOM
```

```
;                                PSECT SUMMARY

;       Name                    Bytes                   Attributes

;  _BAS$DATA                       68  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _BAS$CODE                     3524  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                             Library Statistics

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

;  _$255$DUA28:[SYSLIB]STARLET.L32;1     9776         37         0          581        00:01.1



;                                COMMAND QUALIFIERS

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

; Size:          3012 code + 580 data bytes
; Run Time:         01:35.6
; Elapsed Time:     03:17.0
; Lines/CPU Min:     3009
; Lexemes/CPU-Min: 41958
; Memory Used:  550 pages
; Compilation Complete
```

BASENDDFS
LIS

BASERROR
LIS

BASENDDEF
LIS

BASEDIT
LIS

BASEND
LIS

BASEDIVP
LIS

BASEMULP
LIS

BASERRTXT
LIS

BASENDGSB
LIS