F 1

```
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
```

FILEID**DBGNPNP

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  PPPPPPPP    NN      NN  PPPPPPPP
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  PPPPPPPP    NN      NN  PPPPPPPP
DD      DD  BB      BB  GG          NN      NN  PP      PP  NN      NN  PP      PP
DD      DD  BB      BB  GG          NN      NN  PP      PP  NN      NN  PP      PP
DD      DD  BB      BB  GG          NNNN    NN  PP      PP  NNNN    NN  PP      PP
DD      DD  BB      BB  GG          NNNN    NN  PP      PP  NNNN    NN  PP      PP
DD      DD  BBBBBBBB    GG          NN  NN  NN  PPPPPPPP    NN  NN  NN  PPPPPPPP
DD      DD  BBBBBBBB    GG          NN  NN  NN  PPPPPPPP    NN  NN  NN  PPPPPPPP
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  PP          NN    NNNN  PP
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  PP          NN    NNNN  PP
DD      DD  BB      BB  GG      GG  NN      NN  PP          NN      NN  PP              ....
DD      DD  BB      BB  GG      GG  NN      NN  PP          NN      NN  PP              ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  PP          NN      NN  PP              ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  PP          NN      NN  PP              ....

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

```
0 MODULE DBGNPNP (IDENT = 'V04-000') =
0
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:
1 !
1 !       DEBUG
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains routines which collectively permform pathname
1 !       parsing according to the DEBUG syntax for pathnames. The lexical
1 !       scanner used by the parser is language dependent and is provided by
1 !       the caller of dbg$npathname_parser.
1 !
1 !       The method of parsing is that of ATNs.
1 !
1 !       This module also contains a routine which parses the objects of a
1 !       SET SCOPE command. This routine invokes the pathname parser, supplying
1 !       the address of a kernel lexical scanner routine.
1 !
1 ! ENVIRONMENT:
1 !
1 !       VAX/VMS
1 !
1 ! AUTHOR:
1 !
1 !       David Plummer
1 !
1 ! CREATION DATE:
1 !
```

```
:    58         0058  1 !       9-SEPT-80
:    59         0059  1 !
:    60         0060  1 ! VERSION:
:    61         0061  1 !
:    62         0062  1 !       V03-004
:    63         0063  1 !
:    64         0064  1 ! MODIFIED BY:
:    65         0065  1 !
:    66         0066  1 !       John Francis      3-Jun-81
:    67         0067  1 !
:    68         0068  1 ! EDIT HISTORY
:    69         0069  1 !
:    70         0070  1 ! 002   13-Mar-81       JF      Change max length to 255 (not 511)
:    71         0071  1 ! 003   30-Apr-81       JF      Add support for %NAME construct
:    72         0072  1 ! 004    3-Jun-81       JF      Print A\B.C rather than A\B\C!
:    73         0073  1 !--
```

```
;    75        0074  1 !
;    76        0075  1 ! TABLE OF CONTENTS:
;    77        0076  1 !
;    78        0077  1
;    79        0078  1 FORWARD ROUTINE
;    80        0079  1         DBG$NPATHNAME_PARSER,                          ! Entry point to parse network
;    81        0080  1         PARSE_PATHNAME,                                ! PN parse network
;    82        0081  1         FIRST_LINE,                                    ! Parse first line reference
;    83        0082  1         LINE_LOOKAHEAD,                                ! Resolves line number -
;    84        0083  1                                                        ! numeric scope conflict
;    85        0084  1         FIRST_LABEL,                                   ! Parses first label reference
;    86        0085  1         LABEL_LOOKAHEAD,                               ! Resolves label num - n.s.
;    87        0086  1         GLOBAL_ITEM,                                   ! Parses global id
;    88        0087  1         NUMERIC_PATHNAME,                              ! Parses numeric pathnames
;    89        0088  1         LINE_ITEM,                                     ! Parses line references
;    90        0089  1         LABEL_ITEM,                                    ! Parses label references
;    91        0090  1         QNAME_ITEM,                                    ! Parses %NAME construct
;    92        0091  1         ID_ITEM,                                       ! Parses ID references
;    93        0092  1         INTEGER_ITEM,                                  ! Parses dangling line or
;    94        0093  1                                                        ! label numbers
;    95        0094  1         SHORT_SCOPE,                                   ! Parses global and numeric scopes
;    96        0095  1         CHECK_PATHNAME         : NOVALUE,              ! Sets value state by scanning pathname desc
;    97        0096  1         DBG$NPATHDESC_TO_CS    : NOVALUE,              ! Translates a p.n. desc to a c.s.
;    98        0097  1         SCOPE_SCANNER          : NOVALUE,              ! Kernel scanner for parsing scopes
;    99        0098  1         DBG$NPARSE_SCOPE_LIST;                         ! Parses scopes list
;   100        0099  1
;   101        0100  1 !
;   102        0101  1 ! INCLUDE FILES:
;   103        0102  1 !
;   104        0103  1
;   105        0104  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;   106        0238  1
;   107        0239  1 !
;   108        0240  1 ! MACROS:
;   109        0241  1 !
;   110        0242  1
;   111        0243  1 MACRO   ! These are parsing and semantic action macros
;   112        0244  1
;   113        0245  1
;   114        0246  1     ! ADVANCE updates the input descriptor to reflect the ingestion
;   115        0247  1     ! of the current lexical string. In addition, a copy of the lexical string
;   116        0248  1     ! descriptor is made.
;   117        0249  1     !
;   118      M 0250  1     ADVANCE =
;   119      M 0251  1         BEGIN
;   120      M 0252  1
;   121      M 0253  1         ch$move (8, lex_string_desc, last_token_desc);
;   122      M 0254  1
;   123      M 0255  1         input_desc [dsc$w_length] = .input_desc [dsc$w_length] -
;   124      M 0256  1                                     (.lex_string_desc [dsc$a_pointer] -
;   125      M 0257  1                                     .input_desc [dsc$a_pointer] +
;   126      M 0258  1                                     .lex_string_desc [dsc$w_length]);
;   127      M 0259  1         input_desc [dsc$a_pointer] = .lex_string_desc [dsc$a_pointer] +
;   128      M 0260  1                                      .lex_string_desc [dsc$w_length];
;   129      M 0261  1
;   130      M 0262  1         last_token = .token;
;   131      M 0263  1
;  132           0264   1          END %,
;  133           0265   1
;  134           0266   1
;  135           0267   1      ! GET_TOKEN calls the lexical scanner for a token
;  136           0268   1      !
;  137         M 0269   1      GET_TOKEN =
;  138         M 0270   1          BEGIN
;  139         M 0271   1
;  140         M 0272   1          BIND
;  141         M 0273   1              ROUTINE LEXICAL_SCANNER = .token_scanner_addr; ! Lexical analyzer
;  142         M 0274   1
;  143         M 0275   1          lexical_scanner (.input_desc, lex_string_desc, token);
;  144         M 0276   1
;  145         M 0277   1
;  146         M 0278   1          ! Check for an integer with a length GTR than 9. If this is the case,
;  147         M 0279   1          ! change token to invalid.
;  148         M 0280   1          !
;  149         M 0281   1          IF .token EQL dbg$k_tok_int
;  150         M 0282   1          THEN
;  151         M 0283   1              IF .lex_string_desc [dsc$w_length] GTR 9
;  152         M 0284   1              THEN
;  153         M 0285   1                  token = dbg$k_tok_inval;
;  154         M 0286   1
;  155           0287   1          END %,
;  156           0288   1
;  157           0289   1
;  158           0290   1      ! SAVE extracts and saves the values of the present input descriptor
;  159           0291   1      !
;  160         M 0292   1      SAVE (LEN, PTR) =
;  161         M 0293   1          BEGIN
;  162         M 0294   1
;  163         M 0295   1          len = .input_desc [dsc$w_length];
;  164         M 0296   1          ptr = .input_desc [dsc$a_pointer];
;  165         M 0297   1
;  166           0298   1          END %,
;  167           0299   1
;  168           0300   1
;  169           0301   1      ! RESTORE sets the present input descriptor values to the ones supplied
;  170           0302   1      !
;  171         M 0303   1      RESTORE (LEN, PTR) =
;  172         M 0304   1          BEGIN
;  173         M 0305   1
;  174         M 0306   1          input_desc [dsc$w_length] = len;
;  175         M 0307   1          input_desc [dsc$a_pointer] = ptr;
;  176         M 0308   1
;  177           0309   1          END %,
;  178           0310   1
;  179           0311   1
;  180           0312   1      ! ADD_TO_LIST adds a counted string to the name list. If there is no room
;  181           0313   1      ! to add the name, a string truncation message is issued. The count fields
;  182           0314   1      ! of the pathname vector are updated.
;  183           0315   1      !
;  184         M 0316   1      ADD_TO_LIST (COUNTED_STRING) =
;  185         M 0317   1          BEGIN
;  186         M 0318   1
;  187         M 0319   1          IF .name_index GEQ dbg$k_max_pathname
;  188         M 0320   1          THEN
```

```
            SIGNAL (dbg$_pathtlong)  ! No return
        ELSE
            BEGIN
            name_vect [.name_index] = counted_string;
            name_index = .name_index + 1;
            END;


        ! Update the count fields
        !
        pathname_desc [pth$b_totcnt] = .pathname_desc [pth$b_totcnt] + 1;
        pathname_desc [pth$b_pathcnt] = .pathname_desc [pth$b_totcnt];

        END %,


    ! ADD_ID adds a non_null name to the name vector. The contents of the lexical
    ! string buffer is copied into a new buffer.
    !
    ADD_ID =
        BEGIN

        LOCAL
            NAME_STRING : REF VECTOR [,BYTE];   ! Vector for counted string

        ! Determine how large a buffer is needed and allocate it.
        !
        name_string = dbg$get_tempmem
                ((.lex_string_desc [dsc$w_length] / %UPVAL) + 1);


        ! Copy the buffer pointed to by the lexical string into the name buffer.
        !
        ch$move (.lex_string_desc [dsc$w_length],
                 .lex_string_desc [dsc$a_pointer],
                 name_string [1]);

        name_string [0] = .lex_string_desc [dsc$w_length];


        ! Add the buffer to the name vector
        !
        add_to_list (.name_string);

        END %,


    ! ADD_INVOCATION_NUMBER attaches an invocation number to the last name added
    ! to the name list. The invocation number augmentation is set.
    !
    ADD_INVOCATION_NUMBER =
        BEGIN

        LOCAL
            POINTER,                                    ! Temporary pointer
            NUMBER_DESC         : dbg$stg_desc,         ! Descriptor for number
            NUM_BUF             : REF VECTOR [,BYTE],   ! Number buffer
        NUMBER;                                         ! Translated number

    augmentations [invocation_found] = true;

    ! A copy of the present lexical string descriptor must be made which
    ! contains a terminating character (<CR>).
    !
    number_desc [dsc$w_length] = .lex_string_desc [dsc$w_length] + 1;

    ! Allocate storage for the number string and terminator
    !
    num_buf = dbg$get_tempmem((.number_desc [dsc$w_length] / %UPVAL) + 1);

    ! Copy over the number string and place the terminator
    !
    pointer = ch$move (.lex_string_desc [dsc$w_length],
                       .lex_string_desc [dsc$a_pointer],
                       .num_buf);
    ch$move (1, UPLIT BYTE (dbg$k_car_return), .pointer);
    number_desc [dsc$a_pointer] = .num_buf;

    ! The descriptor has been set up. Now convert the number.
    !
    IF NOT dbg$nsave_decimal_integer (number_desc, number, dummy)
    THEN
        RETURN sts$k_severe;

    ! Store the invocation number and the index
    !
    pathname_desc [pth$b_locinvoc] = .name_index;
    pathname_desc [pth$l_invocnum] = .number;

    END %,


! ADD_NULL_ID adds a null name string to the name vector to represent a
! global reference or numeric scope. The null string is always the first name.
!
ADD_NULL_ID =
    BEGIN

    ! Write in the address of the null name into the first name spot
    !
    name_vect [0] = null_string;
    pathname_desc [pth$b_totcnt] = .pathname_desc [pth$b_totcnt] + 1;
    pathname_desc [pth$b_pathcnt] = .pathname_desc [pth$b_totcnt];
    name_index = 1;

    END %,


! ADD_GLOBAL_ID inserts the null string into the name list, followed by the
! present id (in the lexical string)
!
ADD_GLOBAL_ID =
    BEGIN

    add_null_id;
    add_id;

    END %,


! ADD_NUMERIC_SCOPE places the null string into the name list and sets up an
! invocation number for it (corresponding to the numeric scope). The invocations
! augmentation is set by add_invocation_number.
!
ADD_NUMERIC_SCOPE =
    BEGIN

    add_null_id;
    add_invocation_number;

    END %,


! ADD_LINE inserts a '%LINE' followed by the line number into the name list.
! LINE augmentations are set.
!
ADD_LINE =
    BEGIN

    LOCAL
        LINE_ITEM : REF VECTOR [,BYTE];

    augmentations [line_found] = true;
    augmentations [line_pending] = false;


    ! Get storage for the string
    !
    line_item = dbg$get_tempmem(((.number_buffer [0] + 6) / %UPVAL) + 1);


    ! Copy in the 'LINE'
    !
    ch$move (6, UPLIT BYTE ('%LINE '), line_item [1]);


    ! Copy over the number
    !
    ch$move (.number_buffer [0], number_buffer [1], line_item [7]);


    ! Fill in the count
    !
    line_item [0] = 6 + .number_buffer [0];

```

```
            ! Add the string to the name list
            !
            add_to_list (.line_item);

            END %,


        ! ADD_LABEL adds '%LABEL' followed by the label number to the name list and
        ! sets the label found augmentation.
        !
        ADD_LABEL =
            BEGIN

            LOCAL
                LABEL_ITEM : REF VECTOR [,BYTE];

            augmentations [label_found] = true;
            augmentations [label_pending] = false;


            ! Get storage for the string
            !
            label_item = dbg$get_tempmem(((.number_buffer [0] + 7) / %UPVAL) + 1);


            ! Copy in the 'LABEL'
            !
            ch$move (7, UPLIT BYTE ('%LABEL '), label_item [1]);


            ! Copy over the number
            !
            ch$move (.number_buffer [0], number_buffer [1], label_item [8]);


            ! Fill in the count
            !
            label_item [0] = 7 + .number_buffer [0];


            ! Add the string to the name list
            !
            add_to_list (.label_item);

            END %,


        ! ADD_TO_L_NUMBER adds pieces of a line or label number to the number buffer.
        ! An augmentation is used to check if this is the first part of the number or
        ! a continuation.
        !
        ADD_TO_L_NUMBER =
            BEGIN

            LOCAL
                NUMBER_DESC : dbg$stg_desc,
                TEMP : REF VECTOR [,BYTE];
```

```

number_desc [dsc$a_pointer] = .lex_string_desc [dsc$a_pointer];
number_desc [dsc$w_length] = .lex_string_desc [dsc$w_length];

! Delete leading '0's
!
WHILE .number_desc [dsc$w_length] GTR 1
DO
    BEGIN
    IF ch$rchar (.number_desc [dsc$a_pointer]) NEQ '0'
    THEN
        EXITLOOP;

    number_desc [dsc$w_length] = .number_desc [dsc$w_length] - 1;
    number_desc [dsc$a_pointer] = .number_desc [dsc$a_pointer] + 1;
    END;        ! End of loop


! Check for new number or continuation
!
IF .augmentations [l_number_started]
THEN
    BEGIN

    ! Add the new number to what we already have
    !
    temp = .number_buffer;
    number_buffer = dbg$get_tempmem
        ((.temp [0] + .number_desc [dsc$w_length]) / %UPVAL) + 1);


    ! concatenate the old string with the new
    !
    ch$move (.temp [0],  temp [1], number_buffer [1]);
    ch$move (.number_desc [dsc$w_length],
             .number_desc [dsc$a_pointer],
             number_buffer [.temp [0] + 1]);

    number_buffer [0] = .temp [0] + .number_desc [dsc$w_length];
    END
ELSE
    BEGIN

    ! Start a new number buffer
    !
    augmentations [l_number_started] = true;

    number_buffer = dbg$get_tempmem
        ((.number_desc [dsc$w_length] / %UPVAL) + 1));

    ch$move (.number_desc [dsc$w_length],
             .number_desc [dsc$a_pointer],
             number_buffer [1]);

    number_buffer [0] = .number_desc [dsc$w_length];
    END;
```

```
        END %;


!
! EQUATED SYMBOLS:
!

LITERAL

        ! These are augmentation literals

        LINE_PENDING            = 0,
        LINE_FOUND              = 1,
        LABEL_PENDING           = 2,
        LABEL_FOUND             = 3,
        INVOCATION_FOUND        = 4,
        L_NUMBER_STARTED        = 5,
        TERMINAL_PENDING        = 6,
        TERMINAL_STATE          = 7;

!
! OWN STORAGE:
!

OWN
        LAST_TOKEN_DESC : dbg$stg_desc,                 ! Copy of last lex string desc
                                                        ! accepted during parsing
        LAST_TOKEN,                                     ! Last token found
        DUMMY,                                          ! Dummy variable
        INPUT_DESC      : REF dbg$stg_desc,             ! Input string descriptor
        PATHNAME_DESC   : REF pth$pathname,             ! Path name descriptor
        NAME_VECT       : REF VECTOR,                   ! Name vector for pathname descriptor
        NAME_INDEX,                                     ! Index into name vector
        VALUE_STATE,                                    ! Return state value
        NUMBER_BUFFER   : REF VECTOR [,BYTE],           ! Buffer for l number
        AUGMENTATIONS   : BITVECTOR [8],                ! Augmentation vector
        TOKEN,                                          ! Lexical token
        TOKEN_SCANNER_ADDR,                             ! Address of lexical scanner
        LEX_STRING_DESC : dbg$stg_desc;                 ! Descriptor of string for token

BIND
        NULL_STRING     = UPLIT BYTE (0);               ! Null string

!
! EXTERNAL REFERENCES:
!

EXTERNAL ROUTINE
        SYS$FAO         : ADDRESSING_MODE (ABSOLUTE),   ! System service
        DBG$NNEXT_WORD,                                 ! Returns next word of input
        DBG$NSYNTAX_ERROR,                              ! Constructs a syntax error
        DBG$NMATCH,                                     ! Matches input to counted strings
        DBG$NOUT_INFO,                                  ! Outputs an informational message
        DBG$NMAKE_ARG_VECT,                             ! Constructs a message argument vector
        DBG$GET_TEMPMEM,                                ! Gets listed dynamic storage
        DBG$NSAVE_DECIMAL_INTEGER;                      ! Converts ascii to integer
```

```
;  531          0663  1
;  532          0664  1 EXTERNAL
;  533          0665  1     DBG$GB_LANGUAGE: BYTE,                        ! Current language
;  534          0666  1     DBG$GL_ORIG_COMMAND_PTR,                      ! Pointer to original command string
;  535          0667  1     DBG$GL_UPCASE_COMMAND_PTR: VECTOR[2];
;  536          0668  1                                                   ! Pointers to start and end
;  537          0669  1                                                   .    of current command string
;  538          0670  1
```

```
GLOBAL ROUTINE DBG$NPATHNAME_PARSER (INPUT, SCANNER, PATHNAME, VALUE, LAST_DESC) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Top level parse network for DEBUG pathname parsing. This network
!       accepts valid DEBUG pathnames and constructs a partial pathname descriptor.
!       Upon return, the caller of this routine must analyze the pathname descriptor
!       in conjunction with the return value, and complete the pathname descriptor.
!
!       This routine will not terminate the collection of a pathname until a
!       null or invalid token has been returned by the scanner routine, or an
!       invalid pathname construct has been encountered. This means
!       that the collected pathname may include part or all of a data item reference.
!
!       This routine expects to have the address of a language specific lexical
!       analyzer routine passed to it. This lexical analyzer supplies tokens to
!       the parser. The tokens recognized are:
!
!       dbg$k_tok_null          -  end of input
!
!       dbg$k_tok_line          - '%LINE'
!
!       dbg$k_tok_label         - '%LABEL'
!
!       dbg$k_tok_bs            - '\' (back slash)
!
!       dbg$k_tok_id            - language specific symbolic identifier
!
!       dbg$k_tok_int           - unsigned integer
!
!       dbg$k_tok_dot           - '.'
!
!       dbg$k_tok_reg           - '%register'
!
!       dbg$k_tok_qname         - '%NAME'

!       dbg$k_tok_inval         - any other string
!
!
!       In conjunction with a token, the scanner routine returns a lexical string
!       which contains the ascii characters associated with the token. Note that
!       integers are not translated into binary values by the scanner.
!
!       The pathname parser assumes the responsibility of updating the input string
!       to reflect the acceptance of a lexical string corresponding to a token.
!
!       Upon success or failure, the input string descriptor is updated to reflect
!       the point at which processing stopped. That is, the dsc$a_pointer field
!       contains the address of the first character not accepted.
!
!
! FORMAL PARAMETERS:
!
!       INPUT                   - The address of a VAX standard string descriptor
!                                 representing the input string
!
```

```
1 !          SCANNER             - The address of a language specific lexical analyzer
1 !
1 !          PATHNAME            - The address of a longword to contain the address
1 !                                of a pathname descriptor
1 !
1 !          VALUE               - The address of a longword to contain an unsigned
1 !                                integer encoding of the type of pathname collected:
1 !
1 !                              dbg$k_line      - pathname describes %LINE entity,
1 !                                                NOT a data item
1 !
1 !                              dbg$k_label     - pathname describes %LABEL entity,
1 !                                                NOT a data item
1 !
1 !                              dbg$k_pn_reg    _ pathname qualified register (not
1 !                                                supported yet), or unqualified
1 !                                                register reference (supported).
1 !                                                In both cases, the register name
1 !                                                is NOT written into the pathname
1 !                                                descriptor, but is left as the
1 !                                                first token in the input buffer.
1 !                                                This means that the pathname
1 !                                                descriptor for an unqualified register
1 !                                                will have an item count of 0.
1 !
1 !                              dbg$k_pn        - pathname may describe a data or
1 !                                                lexical entity
1 !
1 !          LAST_DESC           - The address of a longword to contain the address
1 !                                of a standard string descriptor. This descriptor
1 !                                is a copy of the last lexical string descriptor
1 !                                accepted during parsing
1 !
1 !          [SCOPE_FLAG]        - Optional parameter. If supplied, and if true,
1 !                                then accept global and numeric scopes as well
1 !                                as regular pathnames.
1 !
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !          NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !          NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !          An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !          STS$K_SUCCESS       - Success. Some flavor of pathname returned
1 !
1 !          STS$K_SEVERE        - Failure. Syntax error encountered. VALUE parameter
1 !                                not defined. Input descriptor returned to original state.
1 !
```

```
;  654        0785  1 ! SIDE EFFECTS:
;  655        0786  1 !
;  656        0787  1 !       The input string descriptor is updated to reflect one character beyond the
;  657        0788  1 !       last character accepted.
;  658        0789  1 !--
;  659        0790  1
;  660        0791  2     BEGIN
;  661        0792  2
;  662        0793  2     BUILTIN
;  663        0794  2         ACTUALCOUNT,
;  664        0795  2         ACTUALPARAMETER;
;  665        0796  2
;  666        0797  2     LOCAL
;  667        0798  2         SCOPE_FLAG;                         ! Optional parameter value
;  668        0799  2
;  669        0800  2     ! Set the scope flag
;  670        0801  2     !
;  671        0802  2     scope_flag = (IF actualcount () GTR 5 THEN actualparameter (6) ELSE 0);
;  672        0803  2
;  673        0804  2
;  674        0805  2     ! All this routine does is to initialize the control variables and call the
;  675        0806  2     ! the real parse network.
;  676        0807  2     !
;  677        0808  2     input_desc = .input;
;  678        0809  2     token_scanner_addr = .scanner;
;  679        0810  2
;  680        0811  2     lex_string_desc [dsc$b_class] = dsc$k_class_s;
;  681        0812  2     lex_string_desc [dsc$b_dtype] = dsc$k_dtype_t;
;  682        0813  2     lex_string_desc [dsc$w_length] = 0;
;  683        0814  2     lex_string_desc [dsc$a_pointer] = 0;
;  684        0815  2
;  685        0816  2     last_token_desc [dsc$a_pointer] = .input_desc [dsc$a_pointer];
;  686        0817  2     last_token_desc [dsc$w_length] = .input_desc [dsc$w_length];
;  687        0818  2
;  688        0819  2
;  689        0820  2     ! Obtain storage for the pathname descriptor and line up the name vector
;  690        0821  2     !
;  691        0822  2     pathname_desc = dbg$get_tempmem(dbg$k_pathname_size);
;  692        0823  2     name_vect = pathname_desc [pth$a_pathvector];
;  693        0824  2     name_index = 0;
;  694        0825  2
;  695        0826  2
;  696        0827  2     ! Initialize the fields of the pathname descriptor.
;  697        0828  2     !
;  698        0829  2     pathname_desc [pth$b_totcnt] = 0;
;  699        0830  2     pathname_desc [pth$b_locinvoc] = 0;
;  700        0831  2     pathname_desc [pth$l_invocnum] = 0;
;  701        0832  2
;  702        0833  2
;  703        0834  2     ! Initialize the augmentation vector and set the value state
;  704        0835  2     !
;  705        0836  2     augmentations [line_pending]       = false;
;  706        0837  2     augmentations [line_found]         = false;
;  707        0838  2     augmentations [label_pending]      = false;
;  708        0839  2     augmentations [label_found]        = false;
;  709        0840  2     augmentations [invocation_found]   = false;
;  710        0841  2     augmentations [l_number_started]   = false;
;  711          0842   2      augmentations [terminal_pending]    = false;
;  712          0843   2      augmentations [terminal_state]      = false;
;  713          0844   2
;  714          0845   2      value_state = -1;
;  715          0846   2
;  716          0847   2
;  717          0848   2      ! Variables are initialized. Try to do the parsing.
;  718          0849   2      ! Check for scope acceptance.
;  719          0850   2      !
;  720          0851   2      IF .scope_flag
;  721          0852   2      THEN
;  722          0853   3          BEGIN
;  723          0854   3          IF short_scope ()
;  724          0855   3          THEN
;  725          0856   4              BEGIN
;  726          0857   4              .pathname = .pathname_desc;
;  727          0858   4              RETURN sts$k_success;
;  728          0859   4              END
;  729          0860   3          ELSE
;  730          0861   4              BEGIN
;  731          0862   4              IF parse_pathname ()
;  732          0863   4              THEN
;  733          0864   5                  BEGIN
;  734          0865   5                  .pathname = .pathname_desc;
;  735          0866   5                  RETURN sts$k_success;
;  736          0867   5                  END
;  737          0868   4              ELSE
;  738          0869   4                  RETURN sts$k_severe;
;  739          0870   3              END;
;  740          0871   3          END
;  741          0872   2      ELSE
;  742          0873   3          BEGIN
;  743          0874   3          IF NOT parse_pathname () THEN RETURN sts$k_severe;
;  744          0875   2          END;
;  745          0876   2
;  746          0877   2
;  747          0878   2      ! Set the value state
;  748          0879   2      !
;  749          0880   2      check_pathname ();
;  750          0881   2
;  751          0882   2
;  752          0883   2      ! Return all the expected values.
;  753          0884   2      !
;  754          0885   2      .pathname = .pathname_desc;
;  755          0886   2      .value = .value_state;
;  756          0887   2      .last_desc = last_token_desc;
;  757          0888   2
;  758          0889   2      RETURN sts$k_success;
;  759          0890   2
;  760          0891   1      END;                            !End of DBG$NPATHNAME_PARSER

                                                                     .TITLE  DBGNPNP
                                                                     .IDENT  \V04-000\

                                                                     .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
```

```
                                00  00000 P.AAA:  .BYTE   0                                      ;

                                                  .PSECT  DBG$OWN,NOEXE,  PIC,2

                                    00000 LAST_TOKEN_DESC:
                                                  .BLKB   12
                                    0000C LAST_TOKEN:
                                                  .BLKB   4
                                    00010 DUMMY:  .BLKB   4
                                    00014 INPUT_DESC:
                                                  .BLKB   4
                                    00018 PATHNAME_DESC:
                                                  .BLKB   4
                                    0001C NAME_VECT:
                                                  .BLKB   4
                                    00020 NAME_INDEX:
                                                  .BLKB   4
                                    00024 VALUE_STATE:
                                                  .BLKB   4
                                    00028 NUMBER_BUFFER:
                                                  .BLKB   4
                                    0002C AUGMENTATIONS:
                                                  .BLKB   1
                                    0002D         .BLKB   3
                                    00030 TOKEN:  .BLKB   4
                                    00034 TOKEN_SCANNER_ADDR:
                                                  .BLKB   4
                                    00038 LEX_STRING_DESC:
                                                  .BLKB   12

                                          NULL_STRING=        P.AAA
                                                  .EXTRN  SYS$FAO, DBG$NNEXT_WORD
                                                  .EXTRN  DBG$NSYNTAX_ERROR
                                                  .EXTRN  DBG$NMATCH, DBG$NOUT_INFO
                                                  .EXTRN  DBG$NMAKE_ARG_VECT
                                                  .EXTRN  DBG$GET_TEMPMEM
                                                  .EXTRN  DBG$NSAVE_DECIMAL_INTEGER
                                                  .EXTRN  DBG$GB_LANGUAGE
                                                  .EXTRN  DBG$GL_ORIG_COMMAND_PTR
                                                  .EXTRN  DBG$GL_UPCASE_COMMAND_PTR

                                                  .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                              000C  00000         .ENTRY  DBG$NPATHNAME_PARSER, Save R2,R3       ; 0671
         53 00000000'      EF   9E  00002         MOVAB   PATHNAME_DESC, R3                      ;
         05                6C   91  00009         CMPB    (AP), #5                               ; 0802
                           06   1B  0000C         BLEQU   1$                                     ;
         52         18     AC   D0  0000E         MOVL    24(AP), SCOPE_FLAG                     ;
                           02   11  00012         BRB     2$                                     ;
                           52   D4  00014 1$:     CLRL    SCOPE_FLAG                             ;
   FC    A3         04     AC   D0  00016 2$:     MOVL    INPUT, INPUT_DESC                      ; 0808
   1C    A3         08     AC   D0  0001B         MOVL    SCANNER, TOKEN_SCANNER_ADDR            ; 0809
   20    A3 010E0000       8F   D0  00020         MOVL    #17694720, LEX_STRING_DESC             ; 0813
                    24     A3   D4  00028         CLRL    LEX_STRING_DESC+4                      ; 0814
         50         FC     A3   D0  0002B         MOVL    INPUT_DESC, R0                         ; 0816
   EC    A3         04     A0   D0  0002F         MOVL    4(R0), LAST_TOKEN_DESC+4               ;
   E8    A3                60   B0  00034         MOVW    (R0), LAST_TOKEN_DESC                  ; 0817
```

```
                                    34  DD 00038            PUSHL   #52                                           ; 0822
         00000000G  00              01  FB 0003A            CALLS   #1, DBG$GET_TEMPMEM                            ;
                    63              50  D0 00041            MOVL    R0, PATHNAME_DESC                              ;
             04     A3        08    A0  9E 00044            MOVAB   8(R0), NAME_VECT                               ; 0823
                              08    A3  D4 00049            CLRL    NAME_INDEX                                     ; 0824
                                    60  94 0004C            CLRB    (R0)                                           ; 0829
                              02    A0  94 0004E            CLRB    2(R0)                                          ; 0830
                              04    A0  D4 00051            CLRL    4(R0)                                          ; 0831
                              14    A3  94 00054            CLRB    AUGMENTATIONS                                  ; 0843
             0C     A3              01  CE 00057            MNEGL   #1, VALUE_STATE                                ; 0845
                    16              52  E9 0005B            BLBC    SCOPE_FLAG, 4$                                 ; 0851
          0000V     CF              00  FB 0005E            CALLS   #0, SHORT_SCOPE                                ; 0854
                    08              50  E8 00063            BLBS    R0, 3$                                         ;
          0000V     CF              00  FB 00066            CALLS   #0, PARSE_PATHNAME                             ; 0862
                    0E              50  E9 0006B            BLBC    R0, 5$                                         ;
             0C     BC              63  D0 0006E 3$:        MOVL    PATHNAME_DESC, @PATHNAME                       ; 0865
                                    1F  11 00072            BRB     7$                                             ; 0869
          0000V     CF              00  FB 00074 4$:        CALLS   #0, PARSE_PATHNAME                             ; 0874
                    04              50  E8 00079            BLBS    R0, 6$                                         ;
                    50              04  D0 0007C 5$:        MOVL    #4, R0                                         ;
                                        04 0007F            RET                                                    ;
          0000V     CF              00  FB 00080 6$:        CALLS   #0, CHECK_PATHNAME                             ; 0880
             0C     BC              63  D0 00085            MOVL    PATHNAME_DESC, @PATHNAME                       ; 0885
             10     BC        0C    A3  D0 00089            MOVL    VALUE_STATE, @VALUE                            ; 0886
             14     BC        E8    A3  9E 0008E            MOVAB   LAST_TOKEN_DESC, @LAST_DESC                    ; 0887
                    50              01  D0 00093 7$:        MOVL    #1, R0                                         ; 0889
                                        04 00096            RET                                                    ; 0891

; Routine Size:  151 bytes,    Routine Base:  DBG$CODE + 0000


;  761         0892  1
```

```
1 ROUTINE PARSE_PATHNAME =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine recognizes legal DEBUG pathnames. All special cases are
1 !       trapped first, then the routine goes into a loop to accept the remaining
1 !       elements of the pathname. Augmentations are used to assure the the '%L'
1 !       constructs appear only one time, as well as to check the validity of
1 !       invocation numbers and numeric pathnames.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       Numerious MODULE LEVEL OWN'ed variables.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       The pathname descriptor is constructed for valid pathname references.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS   (1)     - Success. Pathname constructed.
1 !
1 !       STS$K_SEVERE    (4)     - Failure. Illegal pathname.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
```

```
BEGIN

! Get the first token and check for all the legal pathname beginnings
!
get_token,

CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
    OF
    SET

    [dbg$k_tok_line] :
        IF NOT first_line () THEN RETURN sts$k_severe;

    [dbg$k_tok_label] :
        IF NOT first_label () THEN RETURN sts$k_severe;

    [dbg$k_tok_bs] :          ! Looking for a global reference
        IF NOT global_item () THEN RETURN sts$k_severe;

    [dbg$k_tok_id] :          ! starting with an id
        IF NOT id_item () THEN RETURN sts$k_severe;

    [dbg$k_tok_int] :         ! Numeric scope
        IF NOT numeric_pathname () THEN RETURN sts$k_severe;

    [dbg$k_tok_reg] :
        RETURN sts$k_success;

    [dbg$k_tok_qname] :
        IF NOT qname_item () THEN RETURN sts$k_severe;

    [INRANGE, OUTRANGE] :     ! Error
        RETURN sts$k_severe;

    TES;
```

```
2       !
2       ! Loop, collecting the rest of the pathname
2       !
2       get_token;
2       WHILE .token EQL dbg$k_tok_bs AND NOT .augmentations [terminal_state]
2       DO
3           BEGIN
3           ! Check for one more trip through loop
3           !
3           IF .augmentations [terminal_pending]
3           THEN
3               augmentations [terminal_state] = true;
3
3           advance;
3           get_token;
3
3           CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
3               OF
3               SET
3
3               [dbg$k_tok_line] :  ! '%LINE'
3                   IF NOT line_item () THEN RETURN sts$k_severe;
3
3               [dbg$k_tok_label] : ! '%LABEL'
3                   IF NOT label_item () THEN RETURN sts$k_severe;
3
3               [dbg$k_tok_id] :    ! ID found. May have an invocation number.
3                   IF NOT id_item () THEN RETURN sts$k_severe;
3
3               [dbg$k_tok_int] :   ! LINE or LABEL number
3                   IF NOT integer_item () THEN RETURN sts$k_severe;
3
3               [dbg$k_tok_qname] : ! %NAME 'name'
3                   IF NOT qname_item () THEN RETURN sts$k_severe;
3
3               [INRANGE, OUTRANGE] :       ! Error
3                   RETURN sts$k_severe;
3
3               TES;
3
3           ! Obtain the next token
3           !
3           get_token;
3
2           END;            ! End of loop
```

```
;  884        1011  2      !
;  885        1012  2      ! Must end parsing on eol
;  886        1013  2      !
;  887        1014  2      IF .token NEQ dbg$k_tok_null
;  888        1015  2                   AND
;  889        1016  2         .token NEQ dbg$k_tok_inval
;  890        1017  2                   AND
;  891        1018  2         .token NEQ dbg$k_tok_id
;  892        1019  2                   AND
;  893        1020  3        (.token NEQ dbg$k_tok_dot OR .last_token NEQ dbg$k_tok_id)
;  894        1021  2      THEN
;  895        1022  2          RETURN sts$k_severe;
;  896        1023  2
;  897        1024  2
;  898        1025  2      ! See if a '%LINE' or '%LABEL' has been left dangling
;  899        1026  2      !
;  900        1027  2      IF .augmentations [line_pending] OR .augmentations [label_pending]
;  901        1028  2      THEN
;  902        1029  2          RETURN sts$k_severe;
;  903        1030  2
;  904        1031  2      RETURN sts$k_success;
;  905        1032  2
;  906        1033  1      END;                    ! End of parse_pathname



                                                007C 00000 PARSE_PATHNAME:
                                                           .WORD   Save R2,R3,R4,R5,R6                      ; 0893
                           56 00000000'     EF  9E 00002           MOVAB   TOKEN, R6
                                            56  DD 00009           PUSHL   R6                                       ; 0931
                                      08    A6  9F 0000B           PUSHAB  LEX_STRING_DESC                          ;
                                      E4    A6  DD 0000E           PUSHL   INPUT_DESC                               ;
                         04    B6           03  FB 00011           CALLS   #3, @TOKEN_SCANNER_ADDR                  ;
                               06           66  D1 00015           CMPL    TOKEN, #6                                ;
                                            09  12 00018           BNEQ    1$                                       ;
                               09     08    A6  B1 0001A           CMPW    LEX_STRING_DESC, #9                      ;
                                            03  1B 0001E           BLEQU   1$                                       ;
                               66           01  D0 00020           MOVL    #1, TOKEN                                ;
                09             00           66  CF 00023 1$:       CASEL   TOKEN, #0, #9                            ; 0937
     001E     0017           010D         010D     00027 2$:       .WORD   25$-2$,-                                 ;
     010D     0033           002C         0025     0002F                   25$-2$,-                                 ;
                             00DF         0111     00037                   4$-2$,-                                  ;
                                                                           5$-2$,-                                  ;
                                                                           6$-2$,-                                  ;
                                                                           7$-2$,-                                  ;
                                                                           8$-2$,-                                  ;
                                                                           25$-2$,-                                 ;
                                                                           26$-2$,-                                 ;
                                                                           21$-2$                                   ;
                                          00F6  31 0003B 3$:       BRW     25$                                      ; 0963
                      0000V  CF             00  FB 0003E 4$:       CALLS   #0, FIRST_LINE                           ; 0942
                                            1A  11 00043           BRB     9$                                       ;
                      0000V  CF             00  FB 00045 5$:       CALLS   #0, FIRST_LABEL                          ; 0945
                                            13  11 0004A           BRB     9$                                       ;
                      0000V  CF             00  FB 0004C 6$:       CALLS   #0, GLOBAL_ITEM                          ; 0948
```

```
                                         0C  11  00051         BRB     9$                                        ;
                    0000V    CF          00  FB  00053  7$:    CALLS   #0, ID_ITEM                               ; 0951
                                         05  11  00058         BRB     9$                                        ;
                    0000V    CF          00  FB  0005A  8$:    CALLS   #0, NUMERIC_PATHNAME                      ; 0954
                             D9          50  E9  0005F  9$:    BLBC    R0, 3$                                    ; 0960
                                         56  DD  00062  10$:   PUSHL   R6                                        ; 0965
                                   08    A6  9F  00064         PUSHAB  LEX_STRING_DESC                           ;
                                   E4    A6  DD  00067         PUSHL   INPUT_DESC                                ;
                       04    B6          03  FB  0006A         CALLS   #3, @TOKEN_SCANNER_ADDR                   ;
                             06          66  D1  0006E         CMPL    TOKEN, #6                                 ;
                                         09  12  00071         BNEQ    11$                                       ;
                             09    08    A6  B1  00073         CMPW    LEX_STRING_DESC, #9                       ;
                                         03  1B  00077         BLEQU   11$                                       ;
                             66          01  D0  00079         MOVL    #1, TOKEN                                 ;
                             04          66  D1  0007C  11$:   CMPL    TOKEN, #4                                 ; 0970
                                         03  13  0007F         BEQL    13$                                       ;
                                       008D  31  00081  12$:   BRW     23$                                       ;
                                   FC    A6  95  00084  13$:   TSTB    AUGMENTATIONS                             ;
                                         F8  19  00087         BLSS    12$                                       ;
                05     FC    A6          06  E1  00089         BBC     #6, AUGMENTATIONS, 14$                    ; 0975
                       FC    A6    80    8F  88  0008E         BISB2   #128, AUGMENTATIONS                       ; 0977
          D0    A6     08    A6          03  28  00093  14$:   MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC      ;
                             51    E4    A6  D0  00099         MOVL    INPUT_DESC, R1                            ;
                50     0C    A6    04    A1  C3  0009D         SUBL3   4(R1), LEX_STRING_DESC+4, R0              ;
                             52    08    A6  3C  000A3         MOVZWL  LEX_STRING_DESC, R2                       ;
                             50          52  C0  000A7         ADDL2   R2, R0                                    ;
                             61          50  A2  000AA         SUBW2   R0, (R1)                                  ;
                       04    A1    0C  B642  9E  000AD         MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)             ;
                       DC    A6          66  D0  000B3         MOVL    TOKEN, LAST_TOKEN                         ;
                                         56  DD  000B7         PUSHL   R6                                        ; 0979
                                   08    A6  9F  000B9         PUSHAB  LEX_STRING_DESC                           ;
                                         51  DD  000BC         PUSHL   R1                                        ;
                       04    B6          03  FB  000BE         CALLS   #3, @TOKEN_SCANNER_ADDR                   ;
                             06          66  D1  000C2         CMPL    TOKEN, #6                                 ;
                                         09  12  000C5         BNEQ    15$                                       ;
                             09    08    A6  B1  000C7         CMPW    LEX_STRING_DESC, #9                       ;
                                         03  1B  000CB         BLEQU   15$                                       ;
                             66          01  D0  000CD         MOVL    #1, TOKEN                                 ;
                09           00          66  CF  000D0  15$:   CASEL   TOKEN, #0, #9                             ; 0982
  001D        0016         0060        0060      000D4  16$:   .WORD   25$-16$,-                                 ;
  0060        002B         0024        0060      000DC                 25$-16$,-                                 ;
                           0032        0060      000E4                 17$-16$,-                                 ;
                                                                       18$-16$,-                                 ;
                                                                       25$-16$,-                                 ;
                                                                       19$-16$,-                                 ;
                                                                       20$-16$,-                                 ;
                                                                       25$-16$,-                                 ;
                                                                       25$-16$,-                                 ;
                                                                       21$-16$                                   ;
                                         4A  11  000E8         BRB     25$                                       ; 1002
                    0000V    CF          00  FB  000EA  17$:   CALLS   #0, LINE_ITEM                             ; 0987
                                         1A  11  000EF         BRB     22$                                       ;
                    0000V    CF          00  FB  000F1  18$:   CALLS   #0, LABEL_ITEM                            ; 0990
                                         13  11  000F6         BRB     22$                                       ;
                    0000V    CF          00  FB  000F8  19$:   CALLS   #0, ID_ITEM                               ; 0993
                                         0C  11  000FD         BR3     22$                                       ;
                    0000V    CF          00  FB  000FF  20$:   CALLS   #0, INTEGER_ITEM                          ; 0996
```

```
                                        05  11 00104          BRB     22$                                   ;
                 0000V  CF              00  FB 00106 21$:     CALLS   #0, QNAME_ITEM                        ; 0999
                        26              50  E9 0010B 22$:     BLBC    R0, 25$                               ;
                                      FF51  31 0010E          BRW     10$                                   ;
                        50              66  D0 00111 23$:     MOVL    TOKEN, R0                             ; 1014
                                        15  13 00114          BEQL    24$                                   ;
                        01              50  D1 00116          CMPL    R0, #1                                ; 1016
                                        10  13 00119          BEQL    24$                                   ;
                        05              50  D1 0011B          CMPL    R0, #5                                ; 1018
                                        0B  13 0011E          BEQL    24$                                   ;
                        07              50  D1 00120          CMPL    R0, #7                                ; 1020
                                        0F  12 00123          BNEQ    25$                                   ;
                        05        DC    A6  D1 00125          CMPL    LAST_TOKEN, #5                        ;
                                        09  12 00129          BNEQ    25$                                   ;
                        05        FC    A6  E8 0012B 24$:     BLBS    AUGMENTATIONS, 25$                    ; 1027
             04       FC        A6      02  E1 0012F          BBC     #2, AUGMENTATIONS, 26$                ;
                        50              04  D0 00134 25$:     MOVL    #4, R0                                ; 1029
                                            04 00137          RET                                           ;
                        50              01  D0 00138 26$:     MOVL    #1, R0                                ; 1031
                                            04 0013B          RET                                           ; 1033

; Routine Size:  316 bytes,    Routine Base:  DBG$CODE + 0097


;  907          1034  1
```

```
1 ROUTINE FIRST_LINE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is called when the pathname begins with '%LINE'. Special
1 !       handling is given to the resolution of line numbers vs. numeric pathnames.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       Augmentations and MODULE OWN'ed variables.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Part or all of pathname parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Illegal construct encountered.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       All or part of the pathname descriptor may be constructed.
1 !
1 !--
2    BEGIN
2
2    augmentations [line_pending] = true;
2    advance;
2
2    ! Get the next token. If it is an integer, we are going to have to
2    ! do some lookahead to see if it is a line number or numeric scope.
2    !
2    get_token;
2
2    CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
2        OF
2        SET
2
2        [dbg$k_tok_bs] :        ! Do nothing
3            BEGIN
3            0;
2            END;
2
2        [dbg$k_tok_id] :        ! ID followed by possible invocation number
2            IF NOT id_item () THEN RETURN sts$k_severe;     ! Save the id and advance
2
```

```
                                                        M  2
DBGNPNP                                          16-Sep-1984 01:50:44    VAX-11 Bliss-32 V4.0-742            Page  25
V04-000                                          14-Sep-1984 12:17:18    [DEBUG.SRC]DBGNPNP.B32;1                (8)

;  966          1092  2          [dbg$k_tok_int] : ! Here we must do lookahead to see if we have a line number
;  967          1093  2              IF NOT line_lookahead () THEN RETURN sts$k_severe;
;  968          1094  2
;  969          1095  2          [INRANGE,OUTRANGE] :     ! Error
;  970          1096  2              RETURN sts$k_severe;
;  971          1097  2
;  972          1098  2          TES;
;  973          1099  2
;  974          1100  2      RETURN sts$k_success;
;  975          1101  2
;  976          1102  1      END;                ! End of FIRST_LINE



                                          007C 00000 FIRST_LINE:
                                                           .WORD   Save R2,R3,R4,R5,R6                          ; 1035
                          56 00000000'   EF   9E 00002     MOVAB   TOKEN, R6                                    ;
                    FC   A6          01   88 00009         BISB2   #1, AUGMENTATIONS                            ; 1072
          D0   A6   08   A6          08   28 0000D         MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC         ;
                          51     E4   A6   D0 00013        MOVL    INPUT_DESC, R1                               ;
               50   0C   A6     04   A1   C3 00017         SUBL3   4(R1), LEX_STRING_DESC+4, R0                 ;
                          52     08   A6   3C 0001D        MOVZWL  LEX_STRING_DESC, R2                          ;
                          50          52   C0 00021        ADDL2   R2, R0                                       ;
                          61          50   A2 00024        SUBW2   R0, (R1)                                     ;
                    04   A1     0C B642   9E 00027         MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)                ;
                    DC   A6          66   D0 0002D         MOVL    TOKEN, LAST_TOKEN                            ;
                                     56   DD 00031         PUSHL   R6                                           ; 1073
                                08   A6   9F 00033         PUSHAB  LEX_STRING_DESC                              ;
                                     51   DD 00036         PUSHL   R1                                           ;
                    04   B6          03   FB 00038         CALLS   #3, @TOKEN_SCANNER_ADDR                      ;
                         06          66   D1 0003C         CMPL    TOKEN, #6                                    ;
                                     09   12 0003F         BNEQ    1$                                           ;
                         09     08   A6   B1 00041         CMPW    LEX_STRING_DESC, #9                          ;
                                     03   1B 00045         BLEQU   1$                                           ;
                         66          01   D0 00047         MOVL    #1, TOKEN                                    ;
               09        00          66   CF 0004A 1$:     CASEL   TOKEN, #0, #9                                ; 1080
     0025    0025      0025        0025      0004E 2$:     .WORD   6$-2$,-                                      ;
     0025    001D      0016        0029      00056                 6$-2$,-                                      ;
                       0025        0025      0005E                 6$-2$,-                                      ;
                                                                   6$-2$,-                                      ;
                                                                   7$-2$,-                                      ;
                                                                   3$-2$,-                                      ;
                                                                   4$-2$,-                                      ;
                                                                   6$-2$,-                                      ;
                                                                   6$-2$,-                                      ;
                                                                   6$-2$                                        ;
                                     0F   11 00062         BRB     6$                                           ; 1096
               0000V   CF            00   FB 00064 3$:     CALLS   #0, ID_ITEM                                  ; 1090
                                     05   11 00069         BRB     5$                                           ;
               0000V   CF            00   FB 0006B 4$:     CALLS   #0, LINE_LOOKAHEAD                           ; 1093
                         04          50   E8 00070 5$:     BLBS    R0, 7$                                       ;
                         50          04   D0 00073 6$:     MOVL    #4, R0                                       ;
                                          04 00076         RET                                                  ;
                         50          01   D0 00077 7$:     MOVL    #1, R0                                       ; 1100
                                          04 0007A         RET                                                  ; 1102
```

```
; Routine Size:  123 bytes,    Routine Base:  DBG$CODE + 01D3

;  977          1103  1
```

```
1 ROUTINE LINE_LOOKAHEAD =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Distinguishes between line numbers and numeric pathname items when '%LINE'
1 !       is encountered first in pathname parsing.
1 !
1 !       If the numeric pathname item is found, the entire pathname descriptor is
1 !       completed.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       MODULE OWN'ed variables
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Valid entire or partial pathname parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Illegal pathname found.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Part or all of the pathname descriptor may be constructed.
1 !
1 !--
2     BEGIN
2     LOCAL
2         LENGTH,
2         POINTER;
2
2     augmentations [line_pending] = true;
2
2     save (length, pointer);
2     advance;
2     get_token;
2
2     CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
2         OF
2         SET
2
2           ! We appear to have a properly terminated line number.  Note that we
2           ! must accept a line number terminated by an ID as valid because the
2           ! ID could be "DO" (as in "SET BREAK %LINE 20 DO(.....)").
2           !
```

```
2              [dbg$k_tok_null,
2               dbg$k_tok_inval,
2               dbg$k_tok_id]:
3                  BEGIN
3                  restore (.length, .pointer);
3                  get_token;
3
3                  IF NOT integer_item () THEN RETURN sts$k_severe;
3
2                  END;
2
2              [dbg$k_tok_bs] :  ! Lookahead one more time
3                  BEGIN
3                  advance;
3                  get_token;
3
3                  IF .token EQL dbg$k_tok_int
3                  THEN
4                      BEGIN
4
4                      ! The first integer we found was a numeric scope
4                      !
4                      restore (.length, .pointer);
4                      get_token;
4
4                      IF NOT numeric_pathname () THEN RETURN sts$k_severe;
4                      END
3                  ELSE
4                      BEGIN
4
4                      ! The integer was a line number
4                      !
4                      restore (.length, .pointer);
4                      get_token;
4
4                      IF NOT integer_item () THEN RETURN sts$k_severe;
4
3                      END;
2                  END;
2
2              [dbg$k_tok_dot] :        ! Line number with a dot
3                  BEGIN
3                  restore (.length, .pointer);
3                  get_token;
3
3                  IF NOT integer_item () THEN RETURN sts$k_severe;
2                  END;
2
2              [INRANGE,OUTRANGE] :     ! Error
2                  RETURN sts$k_severe;
2
2              TES;
2
2          RETURN sts$k_success;
2
1          END;                 ! End of LINE_LOOKAHEAD
```

```
                                             03FC 00000 LINE_LOOKAHEAD:
                                                              .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9                   ; 1104
                     59 00000000'  EF   9E 00002              MOVAB    TOKEN, R9                                      ;
               FC    A9            01   88 00009              BISB2    #1, AUGMENTATIONS                              ; 1147
                     56       E4   A9   D0 0000D              MOVL     INPUT_DESC, R6                                 ; 1149
                     58            66   3C 00011              MOVZWL   (R6), LENGTH                                   ;
                     57       04   A6   D0 00014              MOVL     4(R6), POINTER                                 ;
     D0    A9   08   A9            08   28 00018              MOVC3    #8, LEX_STRING_DESC, LAST_TOKEN_DESC           ;
           50   0C   A9       04   A6   C3 0001E              SUBL3    4(R6), LEX_STRING_DESC+4, R0                   ;
                     51       08   A9   3C 00024              MOVZWL   LEX_STRING_DESC, R1                            ;
                     50            51   C0 00028              ADDL2    R1, R0                                         ;
                     66            50   A2 0002B              SUBW2    R0, (R6)                                       ;
               04    A6       0C B941   9E 0002E              MOVAB    @LEX_STRING_DESC+4[R1], 4(R6)                  ;
               DC    A9            69   D0 00034              MOVL     TOKEN, LAST_TOKEN                              ;
                     59                 DD 00038              PUSHL    R9                                             ; 1150
                              08   A9   9F 0003A              PUSHAB   LEX_STRING_DESC                                ;
                                   56   DD 0003D              PUSHL    R6                                             ;
               04    B9            03   FB 0003F              CALLS    #3, @TOKEN_SCANNER_ADDR                        ;
                     06            69   D1 00043              CMPL     TOKEN, #6                                      ;
                                   09   12 00046              BNEQ     1$                                             ;
                     09       08   A9   B1 00048              CMPW     LEX_STRING_DESC, #9                            ;
                                   03   1B 0004C              BLEQU    1$                                             ;
                     69            01   D0 0004E              MOVL     #1, TOKEN                                      ;
           09        00            69   CF 00051 1$:          CASEL    TOKEN, #0, #9                                  ; 1153
 00CF      00CF      00A3        00A3      00055 2$:          .WORD    7$-2$,-                                        ;
 00A3      00CF      00A3        0017      0005D                       7$-2$,-                                        ;
                     00CF        00CF      00065                       11$-2$,-                                       ;
                                                                       11$-2$,-                                       ;
                                                                       3$-2$,-                                        ;
                                                                       7$-2$,-                                        ;
                                                                       11$-2$,-                                       ;
                                                                       7$-2$,-                                        ;
                                                                       11$-2$,-                                       ;
                                                                       11$-2$                                         ;
                                 00B8   31 00069              BRW      11$                                            ; 1210
     D0    A9   08   A9            08   28 0006C 3$:          MOVC3    #8, LEX_STRING_DESC, LAST_TOKEN_DESC           ; 1173
                     50       E4   A9   D0 00072              MOVL     INPUT_DESC, R0                                 ;
           51   0C   A9       04   A0   C3 00076              SUBL3    4(R0), LEX_STRING_DESC+4, R1                   ;
                     52       08   A9   3C 0007C              MOVZWL   LEX_STRING_DESC, R2                            ;
                     51            52   C0 00080              ADDL2    R2, R1                                         ;
                     60            51   A2 00083              SUBW2    R1, (R0)                                       ;
               04    A0       0C B942   9E 00086              MOVAB    @LEX_STRING_DESC+4[R2], 4(R0)                  ;
               DC    A9            69   D0 0008C              MOVL     TOKEN, LAST_TOKEN                              ;
                                   59   DD 00090              PUSHL    R9                                             ; 1174
                              08   A9   9F 00092              PUSHAB   LEX_STRING_DESC                                ;
                                   50   DD 00095              PUSHL    R0                                             ;
               04    B9            03   FB 00097              CALLS    #3, @TOKEN_SCANNER_ADDR                        ;
                     06            69   D1 0009B              CMPL     TOKEN, #6                                      ;
                                   09   12 0009E              BNEQ     4$                                             ;
                     09       08   A9   B1 000A0              CMPW     LEX_STRING_DESC, #9                            ;
                                   03   1B 000A4              BLEQU    4$                                             ;
                     69            01   D0 000A6              MOVL     #1, TOKEN                                      ;
                     50       E4   A9   D0 000A9 4$:          MOVL     INPUT_DESC, R0                                 ; 1183
```

```
              06          69  D1 000AD          CMPL    TOKEN, #6                                  ; 1177
                          27  12 000B0          BNEQ    6$                                         ;
              60          58  B0 000B2          MOVW    LENGTH, (R0)                               ; 1183
        04    A0          57  D0 000B5          MOVL    POINTER, 4(R0)                             ;
                          59  DD 000B9          PUSHL   R9                                         ;
                    08    A9  9F 000BB          PUSHAB  LEX_STRING_DESC                            ;
                          50  DD 000BE          PUSHL   R0                                         ;
        04    B9          03  FB 000C0          CALLS   #3, @TOKEN_SCANNER_ADDR                    ;
              06          69  D1 000C4          CMPL    TOKEN, #6                                  ;
                          09  12 000C7          BNEQ    5$                                         ;
              09    08    A9  B1 000C9          CMPW    LEX_STRING_DESC, #9                        ;
                          03  1B 000CD          BLEQU   5$                                         ;
              69          01  D0 000CF          MOVL    #1, TOKEN                                  ;
     0000V    CF          00  FB 000D2 5$:      CALLS   #0, NUMERIC_PATHNAME                       ; 1186
                          48  11 000D7          BRB     10$                                        ;
              60          58  B0 000D9 6$:      MOVW    LENGTH, (R0)                               ; 1193
        04    A0          57  D0 000DC          MOVL    POINTER, 4(R0)                             ;
                          59  DD 000E0          PUSHL   R9                                         ;
                    08    A9  9F 000E2          PUSHAB  LEX_STRING_DESC                            ;
                          50  DD 000E5          PUSHL   R0                                         ;
        04    B9          03  FB 000E7          CALLS   #3, @TOKEN_SCANNER_ADDR                    ;
              06          69  D1 000EB          CMPL    TOKEN, #6                                  ;
                          2C  12 000EE          BNEQ    9$                                         ;
              09    08    A9  B1 000F0          CMPW    LEX_STRING_DESC, #9                        ;
                          23  1A 000F4          BGTRU   8$                                         ;
                          24  11 000F6          BRB     9$                                         ; 1196
              50    E4    A9  D0 000F8 7$:      MOVL    INPUT_DESC, R0                             ; 1203
              60          58  B0 000FC          MOVW    LENGTH, (R0)                               ;
        04    A0          57  D0 000FF          MOVL    POINTER, 4(R0)                             ;
                          59  DD 00103          PUSHL   R9                                         ;
                    08    A9  9F 00105          PUSHAB  LEX_STRING_DESC                            ;
                          50  DD 00108          PUSHL   R0                                         ;
        04    B9          03  FB 0010A          CALLS   #3, @TOKEN_SCANNER_ADDR                    ;
              06          69  D1 0010E          CMPL    TOKEN, #6                                  ;
                          09  12 00111          BNEQ    9$                                         ;
              09    08    A9  B1 00113          CMPW    LEX_STRING_DESC, #9                        ;
                          03  1B 00117          BLEQU   9$                                         ;
              69          01  D0 00119 8$:      MOVL    #1, TOKEN                                  ;
     0000V    CF          00  FB 0011C 9$:      CALLS   #0, INTEGER_ITEM                           ; 1206
              04          50  E8 00121 10$:     BLBS    R0, 12$                                    ;
              50          04  D0 00124 11$:     MOVL    #4, R0                                     ;
                              04 00127          RET                                                ;
              50          01  D0 00128 12$:     MOVL    #1, R0                                     ; 1214
                              04 0012B          RET                                                ; 1216

; Routine Size:  300 bytes,    Routine Base:  DBG$CODE + 024E


; 1092          1217  1
```

```
1 ROUTINE FIRST_LABEL =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Invoked when the pathname begins with '%LABEL'. Lookahead may be needed to
1 !       distinguish a numeric pathname item from a label number.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       MODULE level OWN'ed variables
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Part or all of a valid pathname parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Illegal pathname encountered.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Part or all of the pahtname descriptor may be constructed
1 !
1 !--
2    BEGIN
2
2    augmentations [label_pending] = true;
2    advance;
2
2
2    ! Get the next token. If it is an integer, we are going to have to
2    ! do some lookahead to see if it is a label number or numeric scope.
2    !
2    get_token;
2
2    CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
2        OF
2        SET
2
2        [dbg$k_tok_bs] :          ! Do nothing
2            0;
2
2        [dbg$k_tok_id] :          ! ID followed by possible invocation number
2            IF NOT id_item () THEN RETURN sts$k_severe;
2
2        [dbg$k_tok_int] : ! Here we must do lookahead to see if we have
```

```
; 1151          1275  2                 ! a label number or a numeric scope
; 1152          1276  2         IF NOT label_lookahead () THEN RETURN sts$k_severe;
; 1153          1277  2
; 1154          1278  2     [INRANGE,OUTRANGE] :    ! Error
; 1155          1279  2         RETURN sts$k_severe;
; 1156          1280  2
; 1157          1281  2     TES;
; 1158          1282  2
; 1159          1283  2     RETURN sts$k_success;
; 1160          1284  2
; 1161          1285  1     END;                ! End of FIRST_LABEL



                                              007C 00000 FIRST_LABEL:
                                                                .WORD   Save R2,R3,R4,R5,R6                             ; 1218
                        56 00000000'    EF   9E 00002           MOVAB   TOKEN, R6
                  FC    A6              04   88 00009           BISB2   #4, AUGMENTATIONS                               ; 1255
          D0   A6    08    A6              08   28 0000D        MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC            ;
                        51          E4  A6   D0 00013           MOVL    INPUT_DESC, R1                                  ;
             50      0C    A6          04   A1   C3 00017       SUBL3   4(R1), LEX_STRING_DESC+4, R0                    ;
                        52          08  A6   3C 0001D           MOVZWL  LEX_STRING_DESC, R2                             ;
                        50              52   C0 00021           ADDL2   R2, R0                                          ;
                        61              50   A2 00024           SUBW2   R0, (R1)                                        ;
                  04    A1         0C B642   9E 00027           MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)                   ;
                  DC    A6              66   D0 0002D           MOVL    TOKEN, LAST_TOKEN                               ;
                                        56   DD 00031           PUSHL   R6                                              ; 1256
                                    08  A6   9F 00033           PUSHAB  LEX_STRING_DESC                                 ;
                                        51   DD 00036           PUSHL   R1                                              ;
                  04    B6              03   FB 00038           CALLS   #3, @TOKEN_SCANNER_ADDR                         ;
                        06              66   D1 0003C           CMPL    TOKEN, #6                                       ;
                                        09   12 0003F           BNEQ    1$                                              ;
                        09          08  A6   B1 00041           CMPW    LEX_STRING_DESC, #9                             ;
                                        03   1B 00045           BLEQU   1$                                              ;
                        66              01   D0 00047           MOVL    #1, TOKEN                                       ;
             09         00              66   CF 0004A 1$:       CASEL   TOKEN, #0, #9                                   ; 1264
 0025        0025       0025          0025      0004E 2$:       .WORD   6$-2$,-                                         ;
 0025        001D       0016          0029      00056                   6$-2$,-                                         ;
                        0025          0025      0005E                   6$-2$,-                                         ;
                                                                        6$-2$,-                                         ;
                                                                        7$-2$,-                                         ;
                                                                        3$-2$,-                                         ;
                                                                        4$-2$,-                                         ;
                                                                        6$-2$,-                                         ;
                                                                        6$-2$,-                                         ;
                                                                        6$-2$                                           ;
                                        0F   11 00062           BRB     6$                                              ; 1279
                  0000V CF              00   FB 00064 3$:       CALLS   #0, ID_ITEM                                     ; 1272
                                        05   11 00069           BRB     5$                                              ;
                  0000V CF              00   FB 0006B 4$:       CALLS   #0, LABEL_LOOKAHEAD                             ; 1276
                        04              50   E8 00070 5$:       BLBS    R0, 7$                                          ;
                        50              04   D0 00073 6$:       MOVL    #4, R0                                          ;
                                             04 00076           RET                                                     ;
                        50              01   D0 00077 7$:       MOVL    #1, R0                                          ; 1283
                                             04 0007A           RET                                                     ; 1285
```

```
; Routine Size:  123 bytes,    Routine Base:  DBG$CODE + 037A

; 1162          1286  1
```

```
1 ROUTINE LABEL_LOOKAHEAD =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Performs lookahead to distinguish a numeric pathname item from a label number.
1 !
1 !       If a numeric pathname item is found, the entire pathname wiil be parsed.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       MODULE level OWN'ed variables, including the augmentation vector.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Part or all of a valid pathname parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Invalid pathname found.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Part or all of the pathname descriptor may be constructed.
1 !
1 !--
2     BEGIN
2
2     LOCAL
2         LENGTH,
2         POINTER;
2
2     augmentations [label_pending] = true;
2     save (length, pointer);
2     advance;
2     get_token;
2
2     CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
2         OF
2         SET
2
2         [dbg$k_tok_null,
2          dbg$k_tok_inval,
2          dbg$k_tok_id]:
3             BEGIN
3             restore (.length, .pointer);
3             get_token;
```

```
; 1221          1344  3
; 1222          1345  3                 IF NOT integer_item () THEN RETURN sts$k_severe;
; 1223          1346  3
; 1224          1347  2                 END;
; 1225          1348  2
; 1226          1349  2             [dbg$k_tok_bs] :  ! Lookahead one more time
; 1227          1350  3                 BEGIN
; 1228          1351  3                 advance;
; 1229          1352  3                 get_token;
; 1230          1353  3
; 1231          1354  3                 IF .token EQL dbg$k_tok_int
; 1232          1355  3                 THEN
; 1233          1356  4                     BEGIN
; 1234          1357  4
; 1235          1358  4                     ! The first integer we found was a numeric scope
; 1236          1359  4                     !
; 1237          1360  4                     restore (.length, .pointer);
; 1238          1361  4                     get_token;
; 1239          1362  4
; 1240          1363  4                     IF NOT numeric_pathname () THEN RETURN sts$k_severe;
; 1241          1364  4                     END
; 1242          1365  3                 ELSE
; 1243          1366  4                     BEGIN
; 1244          1367  4
; 1245          1368  4                     ! The integer was a label number
; 1246          1369  4                     !
; 1247          1370  4                     restore (.length, .pointer);
; 1248          1371  4                     get_token;
; 1249          1372  4
; 1250          1373  4                     IF NOT integer_item () THEN RETURN sts$k_severe;
; 1251          1374  4
; 1252          1375  3                     END;
; 1253          1376  2                 END;
; 1254          1377  2
; 1255          1378  2             [INRANGE,OUTRANGE] :     ! Error
; 1256          1379  2                 RETURN sts$k_severe;
; 1257          1380  2
; 1258          1381  2             TES;
; 1259          1382  2
; 1260          1383  2         RETURN sts$k_success;
; 1261          1384  2
; 1262          1385  1         END;                ! End of LABEL_LOOKAHEAD




                                              03FC 00000 LABEL_LOOKAHEAD:
                                                               .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                 ; 1287
                          59 00000000'   EF  9E 00002          MOVAB   TOKEN, R9                                    ;
                    FC    A9             04  88 00009          BISB2   #4, AUGMENTATIONS                            ; 1329
                          56       E4    A9  D0 0000D          MOVL    INPUT_DESC, R6                               ; 1330
                          58             66  3C 00011          MOVZWL  (R6), LENGTH                                 ;
                          57       04    A6  D0 00014          MOVL    4(R6), POINTER                               ;
      D0    A9      08    A9             08  28 00018          MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC         ;
            50      0C    A9       04    A6  C3 0001E          SUBL3   4(R6), LEX_STRING_DESC+4, R0                 ;
                          51       08    A9  3C 00024          MOVZWL  LEX_STRING_DESC, R1                          ;
```

```
                         50       51  C0 00028       ADDL2   R1, R0                            ;
                         66       50  A2 0002B       SUBW2   R0, (R6)                          ;
                    04   A6  0C B941  9E 0002E       MOVAB   @LEX_STRING_DESC+4[R1], 4(R6)     ;
                    DC   A9       69  D0 00034       MOVL    TOKEN, LAST_TOKEN                 ;
                                  59  DD 00038       PUSHL   R9                                ; 1331
                             08   A9  9F 0003A       PUSHAB  LEX_STRING_DESC                   ;
                                  56  DD 0003D       PUSHL   R6                                ;
                    04   B9       03  FB 0003F       CALLS   #3, @TOKEN_SCANNER_ADDR           ;
                         06       69  D1 00043       CMPL    TOKEN, #6                         ;
                                  09  12 00046       BNEQ    1$                                ;
                         09  08   A9  B1 00048       CMPW    LEX_STRING_DESC, #9               ;
                                  03  1B 0004C       BLEQU   1$                                ;
                         69       01  D0 0004E       MOVL    #1, TOKEN                         ;
                09       00       69  CF 00051 1$:   CASEL   TOKEN, #0, #9                     ; 1334
00D3            00D3   0017     0017     00055 2$:   .WORD   3$-2$,-                           ;
00D3            00D3   0017     003E     0005D               3$-2$,-                           ;
                       00D3     00D3     00065               13$-2$,-                          ;
                                                             13$-2$,-                          ;
                                                             6$-2$,-                           ;
                                                             3$-2$,-                           ;
                                                             13$-2$,-                          ;
                                                             13$-2$,-                          ;
                                                             13$-2$,-                          ;
                                                             13$-2$                            ;
                                0OBC  31 00069       BRW     13$                               ; 1379
                         50  E4   A9  D0 0006C 3$:   MOVL    INPUT_DESC, R0                    ; 1342
                         60       58  B0 00070       MOVW    LENGTH, (R0)                      ;
                    04   A0       57  D0 00073       MOVL    POINTER, 4(R0)                    ;
                                  59  DD 00077       PUSHL   R9                                ;
                             08   A9  9F 00079       PUSHAB  LEX_STRING_DESC                   ;
                                  50  DD 0007C       PUSHL   R0                                ;
                    04   B9       03  FB 0007E       CALLS   #3, @TOKEN_SCANNER_ADDR           ;
                         06       69  D1 00082       CMPL    TOKEN, #6                         ;
                                  03  13 00085       BEQL    5$                                ;
                                0096  31 00087 4$:   BRW     11$                               ;
                         09  08   A9  B1 0008A 5$:   CMPW    LEX_STRING_DESC, #9               ;
                                  F7  1B 0008E       BLEQU   4$                                ;
                                008A  31 00090       BRW     10$                               ;
          D0    A9  08   A9       08  28 00093 6$:   MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC ; 1350
                         50  E4   A9  D0 00099       MOVL    INPUT_DESC, R0                    ;
                51  0C   A9  04   A0  C3 0009D       SUBL3   4(R0), LEX_STRING_DESC+4, R1      ;
                         52  08   A9  3C 000A3       MOVZWL  LEX_STRING_DESC, R2               ;
                         51       52  C0 000A7       ADDL2   R2, R1                            ;
                         60       51  A2 000AA       SUBW2   R1, (R0)                          ;
                    04   A0  0C B942  9E 000AD       MOVAB   @LEX_STRING_DESC+4[R2], 4(R0)     ;
                    DC   A9       69  D0 000B3       MOVL    TOKEN, LAST_TOKEN                 ;
                                  59  DD 000B7       PUSHL   R9                                ; 1351
                             08   A9  9F 000B9       PUSHAB  LEX_STRING_DESC                   ;
                                  50  DD 000BC       PUSHL   R0                                ;
                    04   B9       03  FB 000BE       CALLS   #3, @TOKEN_SCANNER_ADDR           ;
                         06       69  D1 000C2       CMPL    TOKEN, #6                         ;
                                  09  12 000C5       BNEQ    7$                                ;
                         09  08   A9  B1 000C7       CMPW    LEX_STRING_DESC, #9               ;
                                  03  1B 000CB       BLEQU   7$                                ;
                         69       01  D0 000CD       MOVL    #1, TOKEN                         ;
                         50  E4   A9  D0 000D0 7$:   MOVL    INPUT_DESC, R0                    ; 1360
                         06       69  D1 000D4       CMPL    TOKEN, #6                         ; 1354
```

```
                            27  12 000D7          BNEQ    9$                                  ;
               60           58  B0 000D9          MOVW    LENGTH, (R0)                        ; 1360
          04   A0           57  D0 000DC          MOVL    POINTER, 4(R0)                      ;
                            59  DD 000E0          PUSHL   R9                                  ;
                      08    A9  9F 000E2          PUSHAB  LEX_STRING_DESC                     ;
                            50  DD 000E5          PUSHL   R0                                  ;
          04   B9           03  FB 000E7          CALLS   #3, @TOKEN_SCANNER_ADDR             ;
               06           69  D1 000EB          CMPL    TOKEN, #6                           ;
                            09  12 000EE          BNEQ    8$                                  ;
               09     08    A9  B1 000F0          CMPW    LEX_STRING_DESC, #9                 ;
                            03  1B 000F4          BLEQU   8$                                  ;
               69           01  D0 000F6          MOVL    #1, TOKEN                           ;
       0000V   CF           00  FB 000F9 8$:      CALLS   #0, NUMERIC_PATHNAME                ; 1363
                            25  11 000FE          BRB     12$                                 ;
               60           58  B0 00100 9$:      MOVW    LENGTH, (R0)                        ; 1370
          04   A0           57  D0 00103          MOVL    POINTER, 4(R0)                      ;
                            59  DD 00107          PUSHL   R9                                  ;
                      08    A9  9F 00109          PUSHAB  LEX_STRING_DESC                     ;
                            50  DD 0010C          PUSHL   R0                                  ;
          04   B9           03  FB 0010E          CALLS   #3, @TOKEN_SCANNER_ADDR             ;
               06           69  D1 00112          CMPL    TOKEN, #6                           ;
                            09  12 00115          BNEQ    11$                                 ;
               09     08    A9  B1 00117          CMPW    LEX_STRING_DESC, #9                 ;
                            03  1B 0011B          BLEQU   11$                                 ;
               69           01  D0 0011D 10$:     MOVL    #1, TOKEN                           ;
       0000V   CF           00  FB 00120 11$:     CALLS   #0, INTEGER_ITEM                    ; 1373
               04           50  E8 00125 12$:     BLBS    R0, 14$                             ;
               50           04  D0 00128 13$:     MOVL    #4, R0                              ;
                                04 0012B          RET                                         ;
               50           01  D0 0012C 14$:     MOVL    #1, R0                              ; 1383
                                04 0012F          RET                                         ; 1385

; Routine Size:  304 bytes,    Routine Base:  DBG$CODE + 03F5


; 1263          1386  1
```

```
1 ROUTINE GLOBAL_ITEM =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Invoked when the pathname begins with '\'. The entire pathname correspoding
1 !       to the global reference will be parsed.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       MODULE level OWN'ed variables.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Global reference parsed.
1 !
1 !       STS$K_SEVERE            _ Failure. Invalid pathname detected.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       All of the pathname descriptor will be constructed for a valid global
1 !       reference.
1 !
1 !--
2    BEGIN
2
2    advance;
2    get_token;
2
2
2    ! This must be an id or an id followed by an invocation number
2    !
2    IF .token NEQ dbg$k_tok_id THEN RETURN sts$k_severe;
2
2    add_global_id;
2    advance;
2    get_token;
2
2    CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
2        OF
2        SET
2
2        ! Success and end.
2        !
2        [dbg$k_tok_null,
; 1322          1444  2            dbg$k_tok_inval,
; 1323          1445  2            dbg$k_tok_id]:
; 1324          1446  3               BEGIN
; 1325          1447  3               0;
; 1326          1448  2               END;
; 1327          1449  2
; 1328          1450  2            [dbg$k_tok_int] : ! Invocation number
; 1329          1451  3               BEGIN
; 1330          1452  3               add_invocation_number;
; 1331          1453  3               advance;
; 1332          1454  2               END;
; 1333          1455  2
; 1334          1456  2            [INRANGE,OUTRANGE] :
; 1335          1457  3               BEGIN
; 1336          1458  3               RETURN sts$k_severe;
; 1337          1459  2               END;
; 1338          1460  2
; 1339          1461  2            TES;
; 1340          1462  2
; 1341          1463  2        augmentations [terminal_state] = true;
; 1342          1464  2
; 1343          1465  2        RETURN sts$k_success;
; 1344          1466  2
; 1345          1467  1        END;          ! End of GLOBAL_ITEM


                                                                  .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                            0D  00001 P.AAB:      .BYTE   13                                          ;


                                                                  .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                          01FC  00000 GLOBAL_ITEM:
                                                                  .WORD   Save R2,R3,R4,R5,R6,R7,R8                    ; 1387
                    58 00000000G    00    9E  00002               MOVAB   DBG$GET_TEMPMEM, R8                          ;
                    57 00000000'    EF    9E  00009               MOVAB   LEX_STRING_DESC, R7                          ;
                    5E              10    C2  00010               SUBL2   #16, SP                                      ;
        C8   A7     67              08    28  00013               MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC         ; 1423
                    51        DC    A7    D0  00018               MOVL    INPUT_DESC, R1                               ;
        50     04   A7        04    A1    C3  0001C               SUBL3   4(R1), LEX_STRING_DESC+4, R0                 ;
                    52              67    3C  00022               MOVZWL  LEX_STRING_DESC, R2                          ;
                    50              52    C0  00025               ADDL2   R2, R0                                       ;
                    61              50    A2  00028               SUBW2   R0, (R1)                                     ;
               04   A1        04 B742     9E  0002B               MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)                ;
               D4   A7        F8    A7    D0  00031               MOVL    TOKEN, LAST_TOKEN                            ;
                              F8    A7    9F  00036               PUSHAB  TOKEN                                        ; 1425
                            0082    8F    BB  00039               PUSHR   #^M<R1,R7>                                   ;
               FC   B7              03    FB  0003D               CALLS   #3, @TOKEN_SCANNER_ADDR                      ;
                    06        F8    A7    D1  00041               CMPL    TOKEN, #6                                    ;
                                    09    12  00045               BNEQ    1$                                           ;
                    09              67    B1  00047               CMPW    LEX_STRING_DESC, #9                          ;
                                    04    1B  0004A               BLEQU   1$                                           ;
               F8   A7              01    D0  0004C               MOVL    #1, TOKEN                                    ;
                    05        F8    A7    D1  00050 1$:           CMPL    TOKEN, #5                                    ; 1431
```

```
                                          03  13 00054          BEQL    2$                                              ;
                                        00EC  31 00056          BRW     8$                                              ;
                     E4   B7 00000000'    EF  9E 00059 2$:      MOVAB   NULL_STRING, @NAME_VECT                         ;
                     50             E0    A7  D0 00061          MOVL    PATHNAME_DESC, R0                               ;
                                          60  96 00065          INCB    (R0)                                            ;
                     01   A0              60  90 00067          MOVB    (R0), 1(R0)                                     ;
                     E8   A7              01  D0 0006B          MOVL    #1, NAME_INDEX                                  ;
                          50              67  3C 0006F          MOVZWL  LEX_STRING_DESC, R0                             ;
                          50              04  C6 00072          DIVL2   #4, R0                                          ;
                                    01    A0  9F 00075          PUSHAB  1(R0)                                           ;
                          68              01  FB 00078          CALLS   #1, DBG$GET_TEMPMEM                             ;
                          56              50  D0 0007B          MOVL    R0, NAME_STRING                                 ;
          01   A6    04   B7              67  28 0007E          MOVC3   LEX_STRING_DESC, @LEX_STRING_DESC+4, -          ;
                                                                        1(NAME_STRING)                                  ;
                          66              67  90 00084          MOVB    LEX_STRING_DESC, (NAME_STRING)                  ;
                          52        E8    A7  D0 00087          MOVL    NAME_INDEX, R2                                  ;
                          32              52  D1 0008B          CMPL    R2, #50                                         ;
                                          0F  19 0008E          BLSS    3$                                              ;
                                00028200  8F  DD 00090          PUSHL   #164352                                         ;
               00000000G  00              01  FB 00096          CALLS   #1, LIB$SIGNAL                                  ;
                                          08  11 0009D          BRB     4$                                              ;
                     E4 B742              56  D0 0009F 3$:      MOVL    NAME_STRING, @NAME_VECT[R2]                     ;
                                    E8    A7  D6 000A4          INCL    NAME_INDEX                                      ;
                          50        E0    A7  D0 000A7 4$:      MOVL    PATHNAME_DESC, R0                               ;
                                          60  96 000AB          INCB    (R0)                                            ;
                     01   A0              60  90 000AD          MOVB    (R0), 1(R0)                                     ;
          C8   A7         67              08  28 000B1          MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC            ; 1433
                          51        DC    A7  D0 000B6          MOVL    INPUT_DESC, R1                                  ;
               50    04   A7        04    A1  C3 000BA          SUBL3   4(R1), LEX_STRING_DESC+4, R0                    ;
                          52              67  3C 000C0          MOVZWL  LEX_STRING_DESC, R2                             ;
                          50              52  C0 000C3          ADDL2   R2, R0                                          ;
                          61              50  A2 000C6          SUBW2   R0, (R1)                                        ;
                     04   A1        04  B742  9E 000C9          MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)                   ;
                     D4   A7        F8    A7  D0 000CF          MOVL    TOKEN, LAST_TOKEN                               ;
                                    F8    A7  9F 000D4          PUSHAB  TOKEN                                           ; 1434
                                  0082    8F  BB 000D7          PUSHR   #^M<R1,R7>                                      ;
                     FC   B7              03  FB 000DB          CALLS   #3, @TOKEN_SCANNER_ADDR                         ;
                          06        F8    A7  D1 000DF          CMPL    TOKEN, #6                                       ;
                                          09  12 000E3          BNEQ    5$                                              ;
                          09              67  B1 000E5          CMPW    LEX_STRING_DESC, #9                             ;
                                          04  1B 000E8          BLEQU   5$                                              ;
                     F8   A7              01  D0 000EA          MOVL    #1, TOKEN                                       ;
               09         00        F8    A7  CF 000EE 5$:      CASEL   TOKEN, #0, #9                                   ; 1437
   0052      0052       0086            0086     000F3 6$:      .WORD   10$-6$,-                                        ;
   0052      0016       0086            0052     000FB                  10$-6$,-                                        ;
                        0052            0052     00103                  8$-6$,-                                         ;
                                                                        8$-6$,-                                         ;
                                                                        8$-6$,-                                         ;
                                                                        10$-6$,-                                        ;
                                                                        7$-6$,-                                         ;
                                                                        8$-6$,-                                         ;
                                                                        8$-6$,-                                         ;
                                                                        8$-6$                                           ;
                                          3C  11 00107          BRB     8$                                              ; 1458
                     F4   A7              10  88 00109 7$:      BISB2   #16, AUGMENTATIONS                              ; 1451
          04   AE         67              01  A1 0010D          ADDW3   #1, LEX_STRING_DESC, NUMBER_DESC                ;
                          50        04    AE  3C 00112          MOVZWL  NUMBER_DESC, R0                                 ;
```

```
                                50              04  C6 00116          DIVL2   #4, R0                                        ;
                                         01     A0  9F 00119          PUSHAB  1(R0)                                         ;
                                68              01  FB 0011C          CALLS   #1, DBG$GET_TEMPMEM                           ;
                                56              50  D0 0011F          MOVL    R0, NUM_BUF                                   ;
                  66      04    B7              67  28 00122          MOVC3   LEX_STRING_DESC, @LEX_STRING_DESC+4, -        ;
                                                                              (NUM_BUF)                                     ;
                                63 00000000'    EF  90 00127          MOVB    P.AAB, (POINTER)                              ;
                          08    AE              56  D0 0012E          MOVL    NUM_BUF, NUMBER_DESC+4                        ;
                                         D8     A7  9F 00132          PUSHAB  DUMMY                                         ;
                                         04     AE  9F 00135          PUSHAB  NUMBER                                        ;
                                         0C     AE  9F 00138          PUSHAB  NUMBER_DESC                                   ;
                     00000000G  00              03  FB 0013B          CALLS   #3, DBG$NSAVE_DECIMAL_INTEGER                 ;
                                04              50  E8 00142          BLBS    R0, 9$                                        ;
                                50              04  D0 00145 8$:      MOVL    #4, R0                                        ;
                                                    04 00148          RET                                                   ;
                                50       E0     A7  D0 00149 9$:      MOVL    PATHNAME_DESC, R0                             ;
                          02    A0       E8     A7  90 0014D          MOVB    NAME_INDEX, 2(R0)                             ;
                          04    A0              6E  D0 00152          MOVL    NUMBER, 4(R0)                                 ;
              C8  A7            67              08  28 00156          MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC          ; 1452
                                50       DC     A7  D0 0015B          MOVL    INPUT_DESC, R0                                ;
                  51      04    A7       04     A0  C3 0015F          SUBL3   4(R0), LEX_STRING_DESC+4, R1                  ;
                                52              67  3C 00165          MOVZWL  LEX_STRING_DESC, R2                           ;
                                51              52  C0 00168          ADDL2   R2, R1                                        ;
                                60              51  A2 0016B          SUBW2   R1, (R0)                                      ;
                          04    A0       04 B742    9E 0016E          MOVAB   @LEX_STRING_DESC+4[R2], 4(R0)                 ;
                          D4    A7       F8     A7  D0 00174          MOVL    TOKEN, LAST_TOKEN                             ;
                          F4    A7       80     8F  88 00179 10$:     BISB2   #128, AUGMENTATIONS                           ; 1463
                                50              01  D0 0017E          MOVL    #1, R0                                        ; 1465
                                                    04 00181          RET                                                   ; 1467

; Routine Size:  386 bytes,    Routine Base.  DBG$CODE + 0525


; 1346          1468  1
```

```
1 ROUTINE NUMERIC_PATHNAME =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Parse the entire pathname when a numeric pathname item is encountered at
1 !       the start of a pathname.
1 !
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       MODULE level OWN'ed variables.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Valid numeric pathname parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Invalid pathname found.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The entire pathname descriptor for a valid numeric pathname is constructed.
1 !
1 !--
2    BEGIN
2
2    add_numeric_scope;
2    advance;
2    get_token;
2
2    ! Looking for backslash
2    !
2    IF .token NEQ dbg$k_tok_bs THEN RETURN sts$k_severe;
2
2    advance;
2    get_token;
2
2
2    ! The data item or '%line', '%label' must immediately follow the numeric scope
2    !
2    CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
2        OF
2        SET
2
```

```
; 1405          1526  2           [dbg$k_tok_line] :      ! %line
; 1406          1527  2               IF NOT line_item () THEN RETURN sts$k_severe;
; 1407          1528  2
; 1408          1529  2           [dbg$k_tok_label] : ! '%LABEL'
; 1409          1530  2               IF NOT label_item () THEN RETURN sts$k_severe;
; 1410          1531  2
; 1411          1532  2           [dbg$k_tok_id] :        ! Data reference
; 1412          1533  2               IF NOT id_item () THEN RETURN sts$k_severe;
; 1413          1534  2
; 1414          1535  2           [dbg$k_tok_int] :       ! Possible line or label number
; 1415          1536  2               IF NOT integer_item () THEN RETURN sts$k_severe;
; 1416          1537  2
; 1417          1538  2           [INRANGE, OUTRANGE] :   ! Error
; 1418          1539  2               RETURN sts$k_severe;
; 1419          1540  2
; 1420          1541  2           TES;
; 1421          1542  2
; 1422          1543  2       augmentations [terminal_state] = true;
; 1423          1544  2
; 1424          1545  2       RETURN sts$k_success;
; 1425          1546  2
; 1426          1547  1       END;        ! End of NUMERIC_PATHNAME


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                            0D  00002 P.AAC:    .BYTE   13                                          ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                          00FC 00000 NUMERIC_PATHNAME:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7                      ; 1469
                       57 00000000'   EF   9E 00002            MOVAB   LEX_STRING_DESC, R7                         ;
                       5E             10   C2 00009            SUBL2   #16, SP                                     ;
                 E4    B7 00000000'   EF   9E 0000C            MOVAB   NULL_STRING, @NAME_VECT                     ; 1505
                       50        E0   A7   D0 00014            MOVL    PATHNAME_DESC, R0                           ;
                                      60   96 00018            INCB    (R0)                                        ;
                 01    A0             60   90 0001A            MOVB    (R0), 1(R0)                                 ;
                 E8    A7             01   D0 0001E            MOVL    #1, NAME_INDEX                              ;
                 F4    A7             10   88 00022            BISB2   #16, AUGMENTATIONS                          ;
  04      AE           67             01   A1 00026            ADDW3   #1, LEX_STRING_DESC, NUMBER_DESC            ;
                       50        04   AE   3C 0002B            MOVZWL  NUMBER_DESC, R0                             ;
                       50             04   C6 0002F            DIVL2   #4, R0                                      ;
                                 01   A0   9F 00032            PUSHAB  1(R0)                                       ;
          00000000G    00             01   FB 00035            CALLS   #1, DBG$GET_TEMPMEM                         ;
                       56             50   D0 0003C            MOVL    R0, NUM_BUF                                 ;
        66       04    B7             67   28 0003F            MOVC3   LEX_STRING_DESC, @LEX_STRING_DESC+4, -      ;
                                                                       (NUM_BUF)                                   ;
                       63 00000000'   EF   90 00044            MOVB    P.AAC, (POINTER)                            ;
                 08    AE             56   D0 0004B            MOVL    NUM_BUF, NUMBER_DESC+4                      ;
                                 D8   A7   9F 0004F            PUSHAB  DUMMY                                       ;
                                 04   AE   9F 00052            PUSHAB  NUMBER                                      ;
                                 0C   AE   9F 00055            PUSHAB  NUMBER_DESC                                 ;
          00000000G    00             03   FB 00058            CALLS   #3, DBG$NSAVE_DECIMAL_INTEGER               ;
```

```
                        03           50  E8 0005F          BLBS    R0, 1$                                       :
                                   00C5  31 00062          BRW     10$                                          :
                        50     E0    A7  D0 00065 1$:      MOVL    PATHNAME_DESC, R0                            :
                   02   A0     E8    A7  90 00069          MOVB    NAME_INDEX, 2(R0)                            :
                   04   A0           6E  D0 0006E          MOVL    NUMBER, 4(R0)                                :
          C8   A7       67           08  28 00072          MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC         : 1507
                        51     DC    A7  D0 00077          MOVL    INPUT_DESC, R1                               :
               50  04   A7     04    A1  C3 0007B          SUBL3   4(R1), LEX_STRING_DESC+4, R0                 :
                        52           67  3C 00081          MOVZWL  LEX_STRING_DESC, R2                          :
                        50           52  C0 00084          ADDL2   R2, R0                                       :
                        61           50  A2 00087          SUBW2   R0, (R1)                                     :
                   04   A1     04  B742  9E 0008A          MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)                :
                   D4   A7     F8    A7  D0 00090          MOVL    TOKEN, LAST_TOKEN                            :
                               F8    A7  9F 00095          PUSHAB  TOKEN                                        : 1508
                             0082    8F  BB 00098          PUSHR   #^M<R1,R7>                                   :
                   FC   B7           03  FB 0009C          CALLS   #3, @TOKEN_SCANNER_ADDR                      :
                        06     F8    A7  D1 000A0          CMPL    TOKEN, #6                                    :
                                     09  12 000A4          BNEQ    2$                                           :
                        09           67  B1 000A6          CMPW    LEX_STRING_DESC, #9                          :
                                     04  1B 000A9          BLEQU   2$                                           :
                   F8   A7           01  D0 000AB          MOVL    #1, TOKEN                                    :
                        04     F8    A7  D1 000AF 2$:      CMPL    TOKEN, #4                                    : 1514
                                     75  12 000B3          BNEQ    10$                                          :
          C8   A7       67           08  28 000B5          MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC         :
                        51     DC    A7  D0 000BA          MOVL    INPUT_DESC, R1                               :
               50  04   A7     04    A1  C3 000BE          SUBL3   4(R1), LEX_STRING_DESC+4, R0                 :
                        52           67  3C 000C4          MOVZWL  LEX_STRING_DESC, R2                          :
                        50           52  C0 000C7          ADDL2   R2, R0                                       :
                        61           50  A2 000CA          SUBW2   R0, (R1)                                     :
                   04   A1     04  B742  9E 000CD          MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)                :
                   D4   A7     F8    A7  D0 000D3          MOVL    TOKEN, LAST_TOKEN                            :
                               F8    A7  9F 000D8          PUSHAB  TOKEN                                        : 1516
                             0082    8F  BB 000DB          PUSHR   #^M<R1,R7>                                   :
                   FC   B7           03  FB 000DF          CALLS   #3, @TOKEN_SCANNER_ADDR                      :
                        06     F8    A7  D1 000E3          CMPL    TOKEN, #6                                    :
                                     09  12 000E7          BNEQ    3$                                           :
                        09           67  B1 000E9          CMPW    LEX_STRING_DESC, #9                          :
                                     04  1B 000EC          BLEQU   3$                                           :
                   F8   A7           01  D0 000EE          MOVL    #1, TOKEN                                    :
               09       00     F8    A7  CF 000F2 3$:      CASEL   TOKEN, #0, #9                                : 1522
    001D     0016     0033         0033     000F7 4$:      .WORD   10$-4$,-                                     :
    0033     002B     0024         0033     000FF                  10$-4$,-                                     :
                      0033         0033     00107                  5$-4$,-                                      :
                                                                   6$-4$,-                                      :
                                                                   10$-4$,-                                     :
                                                                   7$-4$,-                                      :
                                                                   8$-4$,-                                      :
                                                                   10$-4$,-                                     :
                                                                   10$-4$,-                                     :
                                                                   10$-4$                                       :
                                     1D  11 0010B          BRB     10$                                          : 1539
                0000V   CF           00  FB 0010D 5$:      CALLS   #0, LINE_ITEM                                : 1527
                                     13  11 00112          BRB     9$                                           :
                0000V   CF           00  FB 00114 6$:      CALLS   #0, LABEL_ITEM                               : 1530
                                     0C  11 00119          BRB     9$                                           :
                0000V   CF           00  FB 0011B 7$:      CALLS   #0, ID_ITEM                                  : 1533
                                     05  11 00120          BRB     9$                                           :
```

```
                                  0000V  CF              00  FB 00122 8$:        CALLS   #0, INTEGER_ITEM                       ; 1536
                                         04              50  E8 00127 9$:        BLBS    R0, 11$                                ;
                                         50              04  D0 0012A 10$:       MOVL    #4, R0                                 ;
                                                             04 0012D            RET                                            ;
                                     F4  A7        80    8F  88 0012E 11$:       BISB2   #128, AUGMENTATIONS                    ; 1543
                                         50              01  D0 00133            MOVL    #1, R0                                 ; 1545
                                                             04 00136            RET                                            ; 1547

; Routine Size:  311 bytes,    Routine Base:  DBG$CODE + 06A7


; 1427          1548  1
```

```
1 ROUTINE LINE_ITEM =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Accepts a '%LINE' line_number construct.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       MODULE level OWN'ed variables.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Line item parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Invalid line item.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The '%LINE' reference is added to the pathname descriptor
1 !
1 !--
2     BEGIN
2
2     ! Check to see if we have already encountered '%LINE' or '%LABEL'
2     !
2     IF .augmentations [line_pending] OR .augmentations [label_pending]
2                                      OR
2        .augmentations [line_found]   OR .augmentations [label_found]
2
2     THEN
2         RETURN sts$k_severe;
2
2     augmentations [line_pending] = true;
2     advance;
2     get_token;
2
2
2     ! Accept the line number
2     !
2     IF .token NEQ dbg$k_tok_int THEN RETURN sts$k_severe;
2
2     IF NOT integer_item () THEN RETURN sts$k_severe;
2
2     RETURN sts$k_success;
```

```
; 1486          1606  2
; 1487          1607  1     END;        ! END of LINE_ITEM



                                                007C 00000 LINE_ITEM:
                                                                      .WORD     Save R2,R3,R4,R5,R6                           ; 1549
                            56 00000000'    EF  9E 00002              MOVAB     AUGMENTATIONS, R6                             ;
                            5E              66  E8 00009              BLBS      AUGMENTATIONS, 2$                             ; 1587
                5A          66              02  E0 0000C              BBS       #2, AUGMENTATIONS, 2$                         ;
                56          66              01  E0 00010              BBS       #1, AUGMENTATIONS, 2$                         ; 1589
                52          66              03  E0 00014              BBS       #3, AUGMENTATIONS, 2$                         ;
                            66              01  88 00018              BISB2     #1, AUGMENTATIONS                             ; 1594
          D4    A6      0C  A6              08  28 0001B              MOVC3     #8, LEX_STRING_DESC, LAST_TOKEN_DESC          ;
                            51          E8  A6  D0 00021              MOVL      INPUT_DESC, R1                                ;
                50      10  A6          04  A1  C3 00025              SUBL3     4(R1), LEX_STRING_DESC+4, R0                  ;
                            52          0C  A6  3C 0002B              MOVZWL    LEX_STRING_DESC, R2                           ;
                            50              52  C0 0002F              ADDL2     R2, R0                                        ;
                            61              50  A2 00032              SUBW2     R0, (R1)                                      ;
                        04  A1          10 B642 9E 00035              MOVAB     @LEX_STRING_DESC+4[R2], 4(R1)                 ;
                        E0  A6          04  A6  D0 0003B              MOVL      TOKEN, LAST_TOKEN                             ;
                                        04  A6  9F 00040              PUSHAB    TOKEN                                         ; 1595
                                        0C  A6  9F 00043              PUSHAB    LEX_STRING_DESC                               ;
                                            51  DD 00046              PUSHL     R1                                            ;
                        08  B6              03  FB 00048              CALLS     #3, @TOKEN_SCANNER_ADDR                       ;
                            06          04  A6  D1 0004C              CMPL      TOKEN, #6                                     ;
                                            0A  12 00050              BNEQ      1$                                            ;
                            09          0C  A6  B1 00052              CMPW      LEX_STRING_DESC, #9                           ;
                                            04  1B 00056              BLEQU     1$                                            ;
                        04  A6              01  D0 00058              MOVL      #1, TOKEN                                     ;
                            06          04  A6  D1 0005C 1$:          CMPL      TOKEN, #6                                     ; 1601
                                            08  12 00060              BNEQ      2$                                            ;
                     0000V  CF              00  FB 00062              CALLS     #0, INTEGER_ITEM                              ; 1603
                            04              50  E8 00067              BLBS      R0, 3$                                        ;
                            50              04  D0 0006A 2$:          MOVL      #4, R0                                        ;
                                                04 0006D              RET                                                     ;
                            50              01  D0 0006E 3$:          MOVL      #1, R0                                        ; 1605
                                                04 00071              RET                                                     ; 1607

; Routine Size:  114 bytes,    Routine Base:  DBG$CODE + 07DE


; 1488          1608  1
```

```
1 ROUTINE LABEL_ITEM =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Parses a '%LABEL' item.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The augmentation vector.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Label item parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Invalid label item.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The label reference is added to the pathname descriptor.
1 !
1 !--
2    BEGIN
2
2    ! See if '%LINE' or '%LABEL' has already been found
2    !
2    IF .augmentations [line_pending] OR .augmentations [label_pending]
2                                     OR
2       .augmentations [line_found]   OR .augmentations [label_found]
2
2    THEN
2       RETURN sts$k_severe;
2
2    augmentations [label_pending] = true;
2    advance;
2    get_token;
2
2
2    ! Accept the label number
2    !
2    IF .token NEQ dbg$k_tok_int THEN RETURN sts$k_severe;
2
2    IF NOT integer_item () THEN RETURN sts$k  evere;
2
2    RETURN sts$k_success;
```

```
; 1547          1666  2
; 1548          1667  1     END;         ! End of LABEL_ITEM



                                              007C 00000 LABEL_ITEM:
                                                                  .WORD   Save R2,R3,R4,R5,R6                       ; 1609
                         56 00000000'   EF   9E 00002            MOVAB   AUGMENTATIONS, R6                          ;
                         5E         66   E8 00009                BLBS    AUGMENTATIONS, 2$                          ; 1647
            5A           66         02   E0 0000C                BBS     #2, AUGMENTATIONS, 2$                      ;
            56           66         01   E0 00010                BBS     #1, AUGMENTATIONS, 2$                      ; 1649
            52           66         03   E0 00014                BBS     #3, AUGMENTATIONS, 2$                      ;
                         66         04   88 00018                BISB2   #4, AUGMENTATIONS                          ; 1654
      D4    A6      0C   A6         08   28 0001B                MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC       ;
                         51     E8  A6   D0 00021                MOVL    INPUT_DESC, R1                             ;
            50      10   A6     04  A1   C3 00025                SUBL3   4(R1), LEX_STRING_DESC+4, R0               ;
                         52     0C  A6   3C 0002B                MOVZWL  LEX_STRING_DESC, R2                        ;
                         50         52   C0 0002F                ADDL2   R2, R0                                     ;
                         61         50   A2 00032                SUBW2   R0, (R1)                                   ;
                    04   A1  10 B642     9E 00035                MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)              ;
                    E0   A6     04  A6   D0 0003B                MOVL    TOKEN, LAST_TOKEN                          ;
                                04  A6   9F 00040                PUSHAB  TOKEN                                      ; 1655
                                0C  A6   9F 00043                PUSHAB  LEX_STRING_DESC                            ;
                                    51   DD 00046                PUSHL   R1                                         ;
                    08   B6         03   FB 00048                CALLS   #3, @TOKEN_SCANNER_ADDR                    ;
                         06     04  A6   D1 0004C                CMPL    TOKEN, #6                                  ;
                                    0A   12 00050                BNEQ    1$                                         ;
                         09     0C  A6   B1 00052                CMPW    LEX_STRING_DESC, #9                        ;
                                    04   1B 00056                BLEQU   1$                                         ;
                    04   A6         01   D0 00058                MOVL    #1, TOKEN                                  ;
                         06     04  A6   D1 0005C 1$:            CMPL    TOKEN, #6                                  ; 1661
                                    08   12 00060                BNEQ    2$                                         ;
                 0000V   CF         00   FB 00062                CALLS   #0, INTEGER_ITEM                           ; 1663
                         04         50   E8 00067                BLBS    R0, 3$                                     ;
                         50         04   D0 0006A 2$:            MOVL    #4, R0                                     ;
                                         04 0006D                RET                                                ;
                         50         01   D0 0006E 3$:            MOVL    #1, R0                                     ; 1665
                                         04 00071                RET                                                ; 1667

; Routine Size:  114 bytes,    Routine Base:  DBG$CODE + 0850


; 1549          1668  1
```

```
1 ROUTINE QNAME_ITEM =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Parses an QNAME item.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The augmentation vector.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Valid QNAME item parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Invalid QNAME item found.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       An ID item is added to the pathname descriptor.
1 !
1 !--
2     BEGIN
2     LOCAL
2         character : BYTE,
2         terminal  : BYTE;
2
2     BIND ROUTINE lexical_scanner = .token_scanner_addr;
2     BIND
2         lexeme_length   = lex_string_desc[dsc$w_length] : WORD,
2         lexeme_pointer  = lex_string_desc[dsc$a_pointer]: LONG;
```

```
;   2     ! First advance over '%NAME' and any following blanks
;   2     !
;   2     advance;
;   2
;   2     IF .input_desc[dsc$w_length] GTRU 0
;   2     THEN
;   3         BEGIN
;   3         character = ch$rchar(.input_desc[dsc$a_pointer]);
;   3         WHILE .character EQL ' ' AND .input_desc[dsc$w_length] GTRU 0
;   3         DO
;   4             BEGIN
;   4             input_desc[dsc$w_length] = .input_desc[dsc$w_length] - 1;
;   4             input_desc[dsc$a_pointer] = ch$plus(.input_desc[dsc$a_pointer],1);
;   4             character = ch$rchar(.input_desc[dsc$a_pointer]);
;   3             END;
;   2         END;
;   2
;   2     IF .input_desc[dsc$w_length] LEQU 0 THEN RETURN sts$k_severe;
;   2
;   2     IF .character EQL '(' OR .character EQL '"' OR .character EQL dbg$k_quote
;   2     THEN        ! Name is enclosed in delimiters
;   3         BEGIN
;   3         IF .input_desc[dsc$w_length] LEQU 2 THEN RETURN sts$k_severe;
;   3         terminal = (IF .character EQL '(' THEN ')' ELSE .character);
;   3         lexeme_length  = 0;
;   3         lexeme_pointer = ch$plus(.input_desc[dsc$a_pointer],1);
;   3         character = ch$rchar(ch$plus(.lexeme_pointer,.lexeme_length));
;   4         WHILE (.character NEQ .terminal)
;   3         DO
;   4             BEGIN
;   4             IF .character EQL dbg$k_car_return THEN RETURN sts$k_severe;
;   4             lexeme_length = .lexeme_length + 1;
;   4             IF .lexeme_length + 1 GEQU .input_desc[dsc$w_length]
;   4             THEN
;   4                 RETURN sts$k_severe;
;   4             character = ch$rchar(ch$plus(.lexeme_pointer,.lexeme_length));
;   3             END;
;   3         END
;   2     ELSE
;   3         BEGIN
;   3         lexical_scanner(.input_desc,lex_string_desc,token);
;   3         IF .token NEQ dbg$k_tok_id AND .token NEQ dbg$k_tok_int
;   3         THEN
;   3             RETURN sts$k_severe;
;   3         terminal = 0;
;   2         END;
; 1642          1758  2        token = dbg$k_tok_id;
; 1643          1759  2        add_id;
; 1644          1760  2        advance;
; 1645          1761  2        IF .terminal NEQ 0
; 1646          1762  2        THEN
; 1647          1763  3            BEGIN
; 1648          1764  3            input_desc[dsc$w_length]  = .input_desc[dsc$w_length] - 1;
; 1649          1765  3            input_desc[dsc$a_pointer] = ch$plus(.input_desc[dsc$a_pointer],1);
; 1650          1766  2            END;
; 1651          1767  2
; 1652          1768  2        get_token;
; 1653          1769  2
; 1654          1770  2        ! Check for invocation number
; 1655          1771  2        !
; 1656          1772  2        IF .token EQL dbg$k_tok_int
; 1657          1773  2        THEN
; 1658          1774  3            BEGIN   ! See if an invocation number has already been found.
; 1659          1775  3            IF .augmentations [invocation_found] THEN RETURN sts$k_severe;
; 1660          1776  3            add_invocation_number;
; 1661          1777  3            advance;
; 1662          1778  2            END;
; 1663          1779  2
; 1664          1780  2        RETURN sts$k_success;
; 1665          1781  2
; 1666          1782  1        END;        ! End of QNAME_ITEM


                                                            .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                           0D  00003 P.AAD:  .BYTE   13                                          ;

                                                      LEXEME_LENGTH=       LEX_STRING_DESC
                                                      LEXEME_POINTER=      LEX_STRING_DESC+4


                                                            .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                         03FC 00000 QNAME_ITEM:
                                                            .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9              ; 1669
                      59 00000000G  00   9E 00002           MOVAB   DBG$GET_TEMPMEM, R9                       ;
                      58 00000000'  EF   9E 00009           MOVAB   LEX_STRING_DESC, R8                       ;
                      5E            10   C2 00010           SUBL2   #16, SP                                   ;
      C8   A8         68            08   28 00013           MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC      ; 1711
                      51       DC   A8   D0 00018           MOVL    INPUT_DESC, R1                            ;
                      52       04   A1   9E 0001C           MOVAB   4(R1), R2                                 ;
           50      04 A8            62   C3 00020           SUBL3   (R2), LEX_STRING_DESC+4, R0               ;
                      53            68   3C 00025           MOVZWL  LEX_STRING_DESC, R3                       ;
                      50            53   C0 00028           ADDL2   R3, R0                                    ;
                      61            50   A2 0002B           SUBW2   R0, (R1)                                  ;
                      62       04 B843   9E 0002E           MOVAB   @LEX_STRING_DESC+4[R3], (R2)              ;
               D4     A8       F8   A8   D0 00033           MOVL    TOKEN, LAST_TOKEN                         ;
                                    61   B5 00038           TSTW    (R1)                                      ; 1716
                                    13   13 0003A           BEQL    2$                                        ;
                      53       00   B2   90 0003C 1$:       MOVB    @0(R2), CHARACTER                         ; 1719
                      20            53   91 00040           CMPB    CHARACTER, #32                            ; 1720
                                    0A   12 00043           BNEQ    2$                                        ;
```

```
                                         61  B5 00045      TSTW    (R1)                                      ;
                                         06  13 00047      BEQL    2$                                        ;
                                         61  B7 00049      DECW    (R1)                                      ; 1723
                                         62  D6 0004B      INCL    (R2)                                      ; 1724
                                         ED  11 0004D      BRB     1$                                        ; 1725
                                         61  B5 0004F 2$:  TSTW    (R1)                                      ; 1729
                                         51  13 00051      BEQL    8$                                        ;
                                         50  D4 00053      CLRL    R0                                        ; 1731
                          28             53  91 00055      CMPB    CHARACTER, #40                            ;
                                         04  12 00058      BNEQ    3$                                        ;
                                         50  D6 0005A      INCL    R0                                        ;
                                         0A  11 0005C      BRB     4$                                        ;
                          22             53  91 0005E 3$:  CMPB    CHARACTER, #34                            ;
                                         05  13 00061      BEQL    4$                                        ;
                          27             53  91 00063      CMPB    CHARACTER, #39                            ;
                                         3F  12 00066      BNEQ    9$                                        ;
                          02             61  B1 00068 4$:  CMPW    (R1), #2                                  ; 1734
                                         37  1B 0006B      BLEQU   8$                                        ;
                          05             50  E9 0006D      BLBC    R0, 5$                                    ; 1735
                          50             29  D0 00070      MOVL    #41, R0                                   ;
                                         03  11 00073      BRB     6$                                        ;
                          50             53  9A 00075 5$:  MOVZBL  CHARACTER, R0                             ;
                          57             50  90 00078 6$:  MOVB    R0, TERMINAL                              ;
                          68             B4 0007B          CLRW    LEXEME_LENGTH                             ; 1736
          04    A8        62             01  C1 0007D      ADDL3   #1, (R2), LEXEME_POINTER                  ; 1737
                          50             68  3C 00082 7$:  MOVZWL  LEXEME_LENGTH, R0                         ; 1738
                          50       04    A8  C0 00085      ADDL2   LEXEME_POINTER, R0                        ;
                          53             60  90 00089      MOVB    (R0), CHARACTER                           ;
                          57             53  91 0008C      CMPB    CHARACTER, TERMINAL                       ; 1739
                                         2F  13 0008F      BEQL    11$                                       ;
                          0D             53  91 00091      CMPB    CHARACTER, #13                            ; 1742
                                         0E  13 00094      BEQL    8$                                        ;
                                         68  B6 00096      INCW    LEXEME_LENGTH                             ; 1743
                          50             68  3C 00098      MOVZWL  LEXEME_LENGTH, R0                         ; 1744
                                         50  D6 0009B      INCL    R0                                        ;
    50          61        10             00  ED 0009D      CMPZV   #0, #16, (R1), R0                         ;
                                         DE  1A 000A2      BGTRU   7$                                        ;
                                       00EC  31 000A4 8$:  BRW     16$                                       ; 1746
                                   F8    A8  9F 000A7 9$:  PUSHAB  TOKEN                                     ; 1752
                                 0102    8F  BB 000AA      PUSHR   #^M<R1,R8>                                ;
                    FC    B8             03  FB 000AE      CALLS   #3, @TOKEN_SCANNER_ADDR                   ;
                          05       F8    A8  D1 000B2      CMPL    TOKEN, #5                                 ; 1753
                                         06  13 000B6      BEQL    10$                                       ;
                          06       F8    A8  D1 000B8      CMPL    TOKEN, #6                                 ;
                                         E6  12 000BC      BNEQ    8$                                        ;
                                         57  94 000BE 10$: CLRB    TERMINAL                                  ; 1756
                    F8    A8             05  D0 000C0 11$: MOVL    #5, TOKEN                                 ; 1758
                          50             68  3C 000C4      MOVZWL  LEX_STRING_DESC, R0                       ;
                          50             04  C6 000C7      DIVL2   #4, R0                                    ;
                                   01    A0  9F 000CA      PUSHAB  1(R0)                                     ;
                          69             01  FB 000CD      CALLS   #1, DBG$GET_TEMPMEM                       ;
                          56             50  D0 000D0      MOVL    R0, NAME_STRING                           ;
          01    A6        04    B8       68  28 000D3      MOVC3   LEX_STRING_DESC, @LEX_STRING_DESC+4,      ;
                                                                   1(NAME_STRING)                            ;
                          66             68  90 000D9      MOVB    LEX_STRING_DESC, (NAME_STRING)            ;
                          52       E8    A8  D0 000DC      MOVL    NAME_INDEX, R2                            ;
                          32             52  D1 000E0      CMPL    R2, #50                                   ;
```

```
                               0F  19 000E3          BLSS     12$                                        ;
                      00028200 8F  DD 000E5          PUSHL    #164352                                    ;
         00000000G  00         01  FB 000EB          CALLS    #1, LIB$SIGNAL                             ;
                               08  11 000F2          BRB      13$                                        ;
              E4 B842          56  D0 000F4 12$:     MOVL     NAME_STRING, @NAME_VECT[R2]                ;
                          E8   A8  D6 000F9          INCL     NAME_INDEX                                 ;
                    50    E0   A8  D0 000FC 13$:     MOVL     PATHNAME_DESC, R0                          ;
                               60  96 00100          INCB     (R0)                                       ;
              01    A0         60  90 00102          MOVB     (R0), 1(R0)                                ;
C8     A8           68         08  28 00106          MOVC3    #8, LEX_STRING_DESC, LAST_TOKEN_DESC       ; 1759
                    50    DC   A8  D0 0010B          MOVL     INPUT_DESC, R0                             ;
       51     04    A8    04   A0  C3 0010F          SUBL3    4(R0), LEX_STRING_DESC+4, R1               ;
                    52         68  3C 00115          MOVZWL   LEX_STRING_DESC, R2                        ;
                    51         52  C0 00118          ADDL2    R2, R1                                     ;
                    60         51  A2 0011B          SUBW2    R1, (R0)                                   ;
              04    A0    04 B842  9E 0011E          MOVAB    @LEX_STRING_DESC+4[R2], 4(R0)              ;
              D4    A8    F8   A8  D0 00124          MOVL     TOKEN, LAST_TOKEN                          ;
                               57  95 00129          TSTB     TERMINAL                                   ; 1761
                               05  13 0012B          BEQL     14$                                        ;
                               60  B7 0012D          DECW     (R0)                                       ; 1764
                          04   A0  D6 0012F          INCL     4(R0)                                      ; 1765
                          F8   A8  9F 00132 14$:     PUSHAB   TOKEN                                      ; 1766
                        0101   8F  BB 00135          PUSHR    #^M<R0,R8>                                 ;
              FC    B8         03  FB 00139          CALLS    #3, @TOKEN_SCANNER_ADDR                    ;
                    06    F8   A8  D1 0013D          CMPL     TOKEN, #6                                  ;
                               09  12 00141          BNEQ     15$                                        ;
                    09         68  B1 00143          CMPW     LEX_STRING_DESC, #9                        ;
                               04  1B 00146          BLEQU    15$                                        ;
              F8    A8         01  D0 00148          MOVL     #1, TOKEN                                  ;
                    06    F8   A8  D1 0014C 15$:     CMPL     TOKEN, #6                                  ; 1772
                               75  12 00150          BNEQ     18$                                        ;
       3C     F4    A8         04  E0 00152          BBS      #4, AUGMENTATIONS, 16$                     ; 1775
              F4    A8         10  88 00157          BISB2    #16, AUGMENTATIONS                         ;
04     AE           68         01  A1 0015B          ADDW3    #1, LEX_STRING_DESC, NUMBER_DESC           ;
                    50    04   AE  3C 00160          MOVZWL   NUMBER_DESC, R0                            ;
                    50         04  C6 00164          DIVL2    #4, R0                                     ;
                          01   A0  9F 00167          PUSHAB   1(R0)                                      ;
                    69         01  FB 0016A          CALLS    #1, DBG$GET_TEMPMEM                        ;
                    56         50  D0 0016D          MOVL     R0, NUM_BUF                                ;
       66     04    B8         68  28 00170          MOVC3    LEX_STRING_DESC, @LEX_STRING_DESC+4, -     ;
                                                              (NUM_BUF)                                  ;
                    63 00000000'   EF  90 00175      MOVB     P.AAD, (POINTER)                           ;
              08    AE         56  D0 0017C          MOVL     NUM_BUF, NUMBER_DESC+4                     ;
                          D8   A8  9F 00180          PUSHAB   DUMMY                                      ;
                          04   AE  9F 00183          PUSHAB   NUMBER                                     ;
                          0C   AE  9F 00186          PUSHAB   NUMBER_DESC                                ;
         00000000G  00         03  FB 00189          CALLS    #3, DBG$NSAVE_DECIMAL_INTEGER              ;
                    04         50  E8 00190          BLBS     R0, 17$                                    ;
                    50         04  D0 00193 16$:     MOVL     #4, R0                                     ;
                                   04 00196          RET                                                 ;
                    50    E0   A8  D0 00197 17$:     MOVL     PATHNAME_DESC, R0                          ;
              02    A0    E8   A8  90 0019B          MOVB     NAME_INDEX, 2(R0)                          ;
              04    A0         6E  D0 001A0          MOVL     NUMBER, 4(R0)                              ;
C8     A8           68         08  28 001A4          MOVC3    #8, LEX_STRING_DESC, LAST_TOKEN_DESC       ; 1776
                    50    DC   A8  D0 001A9          MOVL     INPUT_DESC, R0                             ;
       51     04    A8    04   A0  C3 001AD          SUBL3    4(R0), LEX_STRING_DESC+4, R1               ;
                    52         68  3C 001B3          MOVZWL   LEX_STRING_DESC, R2                        ;
```

```
                                51          52  C0 001B6          ADDL2   R2, R1                                          ;
                                60          51  A2 001B9          SUBW2   R1, (R0)                                        ;
                          04    A0     04 B842  9E 001BC          MOVAB   @LEX_STRING_DESC+4[R2], 4(R0)                   ;
                          D4    A8      F8  A8  D0 001C2          MOVL    TOKEN, LAST_TOKEN                               ;
                                50          01  D0 001C7  18$:    MOVL    #1, R0                                          ; 1780
                                                04 001CA          RET                                                     ; 1782

; Routine Size:  459 bytes,    Routine Base:  DBG$CODE + 08C2


; 1667          1783  1
```

```
1 ROUTINE ID_ITEM =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Parses an ID item.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The augmentation vector.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Valid ID item parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Invalid ID item found.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The ID item is added to the pathname descriptor.
1 !
1 !--
2     BEGIN
2
2     add_id;
2     advance;
2     get_token;
2
2
2     ! Check for invocation number
2     !
2     IF .token EQL dbg$k_tok_int
2     THEN
3         BEGIN   ! See if an invocation number has already been found.
3         IF .augmentations [invocation_found] THEN RETURN sts$k_severe;
3         add_invocation_number;
3         advance;
2         END;
2
2     RETURN sts$k_success;
2
1     END;        ! End of ID_ITEM
```

```
                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                    0D  00004 P.AAE:   .BYTE   13                                          ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                          01FC 00000 ID_ITEM:.WORD  Save R2,R3,R4,R5,R6,R7,R8            ; 1784
                58 00000000G  00   9E 00002          MOVAB   DBG$GET_TEMPMEM, R8                       ;
                57 00000000'  EF   9E 00009          MOVAB   LEX_STRING_DESC, R7                       ;
                5E            10   C2 00010          SUBL2   #16, SP                                   ;
                50            67   3C 00013          MOVZWL  LEX_STRING_DESC, R0                       ; 1818
                50            04   C6 00016          DIVL2   #4, R0                                    ;
                          01  A0   9F 00019          PUSHAB  1(R0)                                     ;
                68            01   FB 0001C          CALLS   #1, DBG$GET_TEMPMEM                       ;
                56            50   D0 0001F          MOVL    R0, NAME_STRING                           ;
 01    A6     04  B7          67   28 00022          MOVC3   LEX_STRING_DESC, @LEX_STRING_DESC+4, -    ;
                                                             1(NAME_STRING)                            ;
                66            67   90 00028          MOVB    LEX_STRING_DESC, (NAME_STRING)            ;
                52        E8  A7   D0 0002B          MOVL    NAME_INDEX, R2                            ;
                32            52   D1 0002F          CMPL    R2, #50                                   ;
                              0F   19 00032          BLSS    1$                                        ;
                   00028200   8F   DD 00034          PUSHL   #164352                                   ;
         00000000G  00        01   FB 0003A          CALLS   #1, LIB$SIGNAL                            ;
                              08   11 00041          BRB     2$                                        ;
             E4 B742          56   D0 00043 1$:      MOVL    NAME_STRING, @NAME_VECT[R2]               ;
                          E8  A7   D6 00048          INCL    NAME_INDEX                                ;
                50        E0  A7   D0 0004B 2$:      MOVL    PATHNAME_DESC, R0                         ;
                              60   96 0004F          INCB    (R0)                                      ;
          01  A0              60   90 00051          MOVB    (R0), 1(R0)                               ;
 C8    A7     67              08   28 00055          MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC      ; 1820
              51          DC  A7   D0 0005A          MOVL    INPUT_DESC, R1                            ;
       50     04  A7      04  A1   C3 0005E          SUBL3   4(R1), LEX_STRING_DESC+4, R0              ;
              52              67   3C 00064          MOVZWL  LEX_STRING_DESC, R2                       ;
              50              52   C0 00067          ADDL2   R2, R0                                    ;
              61              50   A2 0006A          SUBW2   R0, (R1)                                  ;
          04  A1       04 B742     9E 0006D          MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)             ;
          D4  A7          F8  A7   D0 00073          MOVL    TOKEN, LAST_TOKEN                         ;
                          F8  A7   9F 00078          PUSHAB  TOKEN                                     ; 1821
                        0082  8F   BB 0007B          PUSHR   #^M<R1,R7>                                ;
          FC  B7              03   FB 0007F          CALLS   #3, @TOKEN_SCANNER_ADDR                   ;
              06          F8  A7   D1 00083          CMPL    TOKEN, #6                                 ;
                              09   12 00087          BNEQ    3$                                        ;
              09              67   B1 00089          CMPW    LEX_STRING_DESC, #9                       ;
                              04   1B 0008C          BLEQU   3$                                        ;
          F8  A7              01   D0 0008E          MOVL    #1, TOKEN                                 ;
              06          F8  A7   D1 00092 3$:      CMPL    TOKEN, #6                                 ; 1827
                              75   12 00096          BNEQ    6$                                        ;
       3C     F4  A7          04   E0 00098          BBS     #4, AUGMENTATIONS, 4$                     ; 1830
              F4  A7          10   88 0009D          BISB2   #16, AUGMENTATIONS                        ;
 04    AE     67              01   A1 000A1          ADDW3   #1, LEX_STRING_DESC, NUMBER_DESC          ;
              50          04  AE   3C 000A6          MOVZWL  NUMBER_DESC, R0                           ;
              50              04   C6 000AA          DIVL2   #4, R0                                    ;
                          01  A0   9F 000AD          PUSHAB  1(R0)                                     ;
              68              01   FB 000B0          CALLS   #1, DBG$GET_TEMPMEM                       ;
              56              50   D0 000B3          MOVL    R0, NUM_BUF                               ;
```

```
                      66      04  B7            67  28 000B6        MOVC3   LEX_STRING_DESC, @LEX_STRING_DESC+4, -     ;
                                                                            (NUM_BUF)                                   ;
                                  63 00000000'  EF  90 000BB        MOVB    P.AAE, (POINTER)                            ;
                              08  AE            56  D0 000C2        MOVL    NUM_BUF, NUMBER_DESC+4                      ;
                                        D8      A7  9F 000C6        PUSHAB  DUMMY                                       ;
                                        04      AE  9F 000C9        PUSHAB  NUMBER                                      ;
                                        0C      AE  9F 000CC        PUSHAB  NUMBER_DESC                                 ;
                       00000000G  00            03  FB 000CF        CALLS   #3, DBG$NSAVE_DECIMAL_INTEGER               ;
                                  04            50  E8 000D6        BLBS    R0, 5$                                      ;
                                  50            04  D0 000D9 4$:    MOVL    #4, R0                                      ;
                                                04 000DC            RET                                                 ;
                                  50    E0      A7  D0 000DD 5$:    MOVL    PATHNAME_DESC, R0                           ;
                              02  A0    E8      A7  90 000E1        MOVB    NAME_INDEX, 2(R0)                           ;
                              04  A0            6E  D0 000E6        MOVL    NUMBER, 4(R0)                               ;
                  C8  A7          67            08  28 000EA        MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC        ; 1831
                                  50    DC      A7  D0 000EF        MOVL    INPUT_DESC, R0                              ;
                      51      04  A7    04      A0  C3 000F3        SUBL3   4(R0), LEX_STRING_DESC+4, R1                ;
                                  52            67  3C 000F9        MOVZWL  LEX_STRING_DESC, R2                         ;
                                  51            52  C0 000FC        ADDL2   R2, R1                                      ;
                                  60            51  A2 000FF        SUBW2   R1, (R0)                                    ;
                              04  A0    04 B742     9E 00102        MOVAB   @LEX_STRING_DESC+4[R2], 4(R0)               ;
                              D4  A7    F8      A7  D0 00108        MOVL    TOKEN, LAST_TOKEN                           ;
                                  50            01  D0 0010D 6$:    MOVL    #1, R0                                      ; 1835
                                                04 00110            RET                                                 ; 1837

; Routine Size:  273 bytes,    Routine Base:  DBG$CODE + 0ABD


; 1723         1838  1
```

```
1 ROUTINE INTEGER_ITEM =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Parses a dangling line or label number.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The augmentation vector.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. LINE or LABEL number parsed.
1 !
1 !       STS$K_SEVERE            - Failure. Invalid integer item found.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The line or label number is added to the pathname descriptor.
1 !
1 !--
2    BEGIN
2
2    ! Determine if looking for line or label number
2    !
2    SELECTONE true
2        OF
2        SET
2
2        [.augmentations [line_pending]] :       ! Line number
3            BEGIN
3            add_to_l_number;
3            advance;
3            get_token;
3
3
3            ! See if more line number follows
3            !
3            IF .token EQL dbg$k_tok_dot
3            THEN
4                BEGIN
4                add_to_l_number;
4                advance;
4                get_token;
```

```
: 1782          1896  4
: 1783          1897  4             IF .token NEQ dbg$k_tok_int THEN RETURN sts$k_severe;
: 1784          1898  4
: 1785          1899  4             add_to_l_number;
: 1786          1900  4             add_line;
: 1787          1901  4             advance;
: 1788          1902  4             END
: 1789          1903  3         ELSE
: 1790          1904  3             add_line;
: 1791          1905  2         END;
: 1792          1906  2
: 1793          1907  2     [.augmentations [label_pending]] :      ! LABEL number
: 1794          1908  3         BEGIN
: 1795          1909  3         add_to_l_number;
: 1796          1910  3         add_label;
: 1797          1911  3         advance;
: 1798          1912  2         END;
: 1799          1913  2
: 1800          1914  2     [OTHERWISE] :
: 1801          1915  2         RETURN sts$k_severe;
: 1802          1916  2
: 1803          1917  2     TES;
: 1804          1918  2
: 1805          1919  2 augmentations [terminal_pending] = true;
: 1806          1920  2
: 1807          1921  2 RETURN sts$k_success;
: 1808          1922  2
: 1809          1923  1 END;        ! End of INTEGER_ITEM



                                                     .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

              20  45  4E  49  4C  25  00005 P.AAF:   .ASCII  \%LINE \                                  ;
              20  45  4E  49  4C  25  0000B P.AAG:   .ASCII  \%LINE \                                  ;
          20  4C  45  42  41  4C  25  00011 P.AAH:   .ASCII  \%LABEL \                                 ;



                                                     .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                             0FFC 00000 INTEGER_ITEM:
                                                     .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      ; 1839
            5B 00000000G  00  9E 00002               MOVAB   DBG$GET_TEMPMEM, R11                      ;
            5A 00000000'  EF  9E 00009               MOVAB   NUMBER_BUFFER, R10                        ;
            5E            0C  C2 00010               SUBL2   #12, SP                                   ;
            03       04   AA  E8 00013               BLBS    AUGMENTATIONS, 1$                         ; 1881
                        027F  31 00017               BRW     21$                                       ;
        04  AE       14   AA  D0 0001A 1$:           MOVL    LEX_STRING_DESC+4, NUMBER_DESC+4          ; 1882
            6E       10   AA  B0 0001F               MOVW    LEX_STRING_DESC, NUMBER_DESC              ;
            01            6E  B1 00023 2$:           CMPW    NUMBER_DESC, #1                           ;
                          0D  1B 00026               BLEQU   3$                                        ;
            30       04   BE  91 00028               CMPB    @NUMBER_DESC+4, #48                       ;
                          07  12 0002C               BNEQ    3$                                        ;
                          6E  B7 0002E               DECW    NUMBER_DESC                               ;
                     04   AE  D6 00030               INCL    NUMBER_DESC+4                             ;
                          EE  11 00033               BRB     2$                                        ;
                        58          6E  3C 00035 3$:     MOVZWL  NUMBER_DESC, R8                          ;
          31      04    AA          05  E1 00038         BBC     #5, AUGMENTATIONS, 4$                    ;
                        56          6A  D0 0003D         MOVL    NUMBER_BUFFER, TEMP                      ;
                        59          66  9A 00040         MOVZBL  (TEMP), R9                               ;
                        59          58  C0 00043         ADDL2   R8, R9                                   ;
          50            59          04  C7 00046         DIVL3   #4, R9, R0                               ;
                              01    A0  9F 0004A         PUSHAB  1(R0)                                    ;
                        6B          01  FB 0004D         CALLS   #1, DBG$GET_TEMPMEM                      ;
                        6A          50  D0 00050         MOVL    R0, NUMBER_BUFFER                        ;
                        50          66  9A 00053         MOVZBL  (TEMP), R0                               ;
                        57          6A  D0 00056         MOVL    NUMBER_BUFFER, R7                        ;
    01    A7      01    A6          50  28 00059         MOVC3   R0, 1(TEMP), 1(R7)                       ;
                        50          66  9A 0005F         MOVZBL  (TEMP), R0                               ;
    01  A047      04    BE          58  28 00062         MOVC3   R8, @NUMBER_DESC+4, 1(R0)[R7]            ;
                        67          59  90 00069         MOVB    R9, (R7)                                 ;
                                    1D  11 0006C         BRB     5$                                       ;
                  04    AA          20  88 0006E 4$:     BISB2   #32, AUGMENTATIONS                       ;
          50            58          04  C7 00072         DIVL3   #4, R8, R0                               ;
                              01    A0  9F 00076         PUSHAB  1(R0)                                    ;
                        6B          01  FB 00079         CALLS   #1, DBG$GET_TEMPMEM                      ;
                        6A          50  D0 0007C         MOVL    R0, NUMBER_BUFFER                        ;
                        56          6A  D0 0007F         MOVL    NUMBER_BUFFER, R6                        ;
    01    A6      04    BE          58  28 00082         MOVC3   R8, @NUMBER_DESC+4, 1(R6)                ;
                        66          58  90 00088         MOVB    R8, (R6)                                 ;
    D8    AA      10    AA          08  28 0008B 5$:     MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC     ; 1883
                        51    EC    AA  D0 00091         MOVL    INPUT_DESC, R1                           ;
          50      14    AA    04    A1  C3 00095         SUBL3   4(R1), LEX_STRING_DESC+4, R0             ;
                        52    10    AA  3C 0009B         MOVZWL  LEX_STRING_DESC, R2                      ;
                        50          52  C0 0009F         ADDL2   R2, R0                                   ;
                        61          50  A2 000A2         SUBW2   R0, (R1)                                 ;
                  04    A1    14  BA42  9E 000A5         MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)            ;
                  E4    AA    08    AA  D0 000AB         MOVL    TOKEN, LAST_TOKEN                        ;
                              08    AA  9F 000B0         PUSHAB  TOKEN                                    ; 1884
                              10    AA  9F 000B3         PUSHAB  LEX_STRING_DESC                          ;
                                    51  DD 000B6         PUSHL   R1                                       ;
                  0C    BA          03  FB 000B8         CALLS   #3, @TOKEN_SCANNER_ADDR                  ;
                        06    08    AA  D1 000BC         CMPL    TOKEN, #6                                ;
                                    0A  12 000C0         BNEQ    6$                                       ;
                        09    10    AA  B1 000C2         CMPW    LEX_STRING_DESC, #9                      ;
                                    04  1B 000C6         BLEQU   6$                                       ;
                  08    AA          01  D0 000C8         MOVL    #1, TOKEN                                ;
                        07    08    AA  D1 0C0CC 6$:     CMPL    TOKEN, #7                                ; 1890
                                    03  13 000D0         BEQL    7$                                       ;
                                  0163  31 000D2         BRW     18$                                      ;
                  04    AE    14    AA  D0 000D5 7$:     MOVL    LEX_STRING_DESC+4, NUMBER_DESC+4         ; 1892
                        6E    10    AA  B0 000DA         MOVW    LEX_STRING_DESC, NUMBER_DESC             ;
                        01          6E  B1 000DE 8$:     CMPW    NUMBER_DESC, #1                          ;
                                    0D  1B 000E1         BLEQU   9$                                       ;
                        30    04    BE  91 000E3         CMPB    @NUMBER_DESC+4, #48                      ;
                                    07  12 000E7         BNEQ    9$                                       ;
                                    6E  B7 000E9         DECW    NUMBER_DESC                              ;
                              04    AE  D6 000EB         INCL    NUMBER_DESC+4                            ;
                                    EE  11 000EE         BRB     8$                                       ;
                        58          6E  3C 000F0 9$:     MOVZWL  NUMBER_DESC, R8                          ;
          31      04    AA          05  E1 000F3         BBC     #5, AUGMENTATIONS, 10$                   ;
                        56          6A  D0 000F8         MOVL    NUMBER_BUFFER, TEMP                      ;
                        59          66  9A 000FB         MOVZBL  (TEMP), R9                               ;
```

```
                    59         58  C0 000FE         ADDL2   R8, R9                                          ;
          50        59         04  C7 00101         DIVL3   #4, R9, R0                                      ;
                          01   A0  9F 00105         PUSHAB  1(R0)                                           ;
                    6B         01  FB 00108         CALLS   #1, DBG$GET_TEMPMEM                             ;
                    6A         50  D0 0010B         MOVL    R0, NUMBER_BUFFER                               ;
                    50         66  9A 0010E         MOVZBL  (TEMP), R0                                      ;
                    57         6A  D0 00111         MOVL    NUMBER_BUFFER, R7                               ;
    01    A7    01  A6         50  28 00114         MOVC3   R0, 1(TEMP), 1(R7)                              ;
                    50         66  9A 0011A         MOVZBL  (TEMP), R0                                      ;
    01 A047     04  BE         58  28 0011D         MOVC3   R8, @NUMBER_DESC+4, 1(R0)[R7]                   ;
                    67         59  90 00124         MOVB    R9, (R7)                                        ;
                               1D  11 00127         BRB     11$                                             ;
                04  AA         20  88 00129 10$:    BISB2   #32, AUGMENTATIONS                              ;
          50        58         04  C7 0012D         DIVL3   #4, R8, R0                                      ;
                          01   A0  9F 00131         PUSHAB  1(R0)                                           ;
                    6B         01  FB 00134         CALLS   #1, DBG$GET_TEMPMEM                             ;
                    6A         50  D0 00137         MOVL    R0, NUMBER_BUFFER                               ;
                    56         6A  D0 0013A         MOVL    NUMBER_BUFFER, R6                               ;
    01    A6    04  BE         58  28 0013D         MOVC3   R8, @NUMBER_DESC+4, 1(R6)                       ;
                    66         58  90 00143         MOVB    R8, (R6)                                        ;
    D8    AA    10  AA         08  28 00146 11$:    MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC            ; 1893
                    51    EC   AA  D0 0014C         MOVL    INPUT_DESC, R1                                  ;
          50    14  AA    04   A1  C3 00150         SUBL3   4(R1), LEX_STRING_DESC+4, R0                    ;
                    52    10   AA  3C 00156         MOVZWL  LEX_STRING_DESC, R2                             ;
                    50         52  C0 0015A         ADDL2   R2, R0                                          ;
                    61         50  A2 0015D         SUBW2   R0, (R1)                                        ;
                04  A1    14 BA42  9E 00160         MOVAB   @LEX_STRING_DESC+4[R2], 4(R1)                   ;
                E4  AA    08   AA  D0 00166         MOVL    TOKEN, LAST_TOKEN                               ;
                          08   AA  9F 0016B         PUSHAB  TOKEN                                           ; 1894
                          10   AA  9F 0016E         PUSHAB  LEX_STRING_DESC                                 ;
                               51  DD 00171         PUSHL   R1                                              ;
                0C  BA         03  FB 00173         CALLS   #3, @TOKEN_SCANNER_ADDR                         ;
                    06    08   AA  D1 00177         CMPL    TOKEN, #6                                       ;
                               0A  12 0017B         BNEQ    12$                                             ;
                    09    10   AA  B1 0017D         CMPW    LEX_STRING_DESC, #9                             ;
                               04  1B 00181         BLEQU   12$                                             ;
                08  AA         01  D0 00183         MOVL    #1, TOKEN                                       ;
                    06    08   AA  D1 00187 12$:    CMPL    TOKEN, #6                                       ; 1897
                               03  13 0018B         BEQL    13$                                             ;
                             0207  31 0018D         BRW     30$                                             ;
                04  AE    14   AA  D0 00190 13$:    MOVL    LEX_STRING_DESC+4, NUMBER_DESC+4                ;
                    6E    10   AA  B0 00195         MOVW    LEX_STRING_DESC, NUMBER_DESC                    ;
                    01         6E  B1 00199 14$:    CMPW    NUMBER_DESC, #1                                 ;
                               0D  1B 0019C         BLEQU   15$                                             ;
                    30    04   BE  91 0019E         CMPB    @NUMBER_DESC+4, #48                             ;
                               07  12 001A2         BNEQ    15$                                             ;
                               6E  B7 001A4         DECW    NUMBER_DESC                                     ;
                          04   AE  D6 001A6         INCL    NUMBER_DESC+4                                   ;
                               EE  11 001A9         BRB     14$                                             ;
                    58         6E  3C 001AB 15$:    MOVZWL  NUMBER_DESC, R8                                 ;
          31    04  AA         05  E1 001AE         BBC     #5, AUGMENTATIONS, 16$                          ;
                    56         6A  D0 001B3         MOVL    NUMBER_BUFFER, TEMP                             ;
                    59         66  9A 001B6         MOVZBL  (TEMP), R9                                      ;
                    59         58  C0 001B9         ADDL2   R8, R9                                          ;
          50        59         04  C7 001BC         DIVL3   #4, R9, R0                                      ;
                          01   A0  9F 001C0         PUSHAB  1(R0)                                           ;
                    6B         01  FB 001C3         CALLS   #1, DBG$GET_TEMPMEM                             ;
```

```
                       6A              50  D0 001C6        MOVL    R0, NUMBER_BUFFER                          :
                       50              66  9A 001C9        MOVZBL  (TEMP), R0                                 :
                       57              6A  D0 001CC        MOVL    NUMBER_BUFFER, R7                          :
01    A7         01    A6              50  28 001CF        MOVC3   R0, 1(TEMP), 1(R7)                         :
                       50              66  9A 001D5        MOVZBL  (TEMP), R0                                 :
01  A047         04    BE              58  28 001D8        MOVC3   R8, @NUMBER_DESC+4, 1(R0)[R7]              :
                       67              59  90 001DF        MOVB    R9, (R7)                                   :
                                       1D  11 001E2        BRB     17$                                        :
                 04    AA              20  88 001E4 16$:   BISB2   #32, AUGMENTATIONS                         :
      50               58              04  C7 001E8        DIVL3   #4, R8, R0                                 :
                               01      A0  9F 001EC        PUSHAB  1(R0)                                      :
                       6B              01  FB 001EF        CALLS   #1, DBG$GET_TEMPMEM                        :
                       6A              50  D0 001F2        MOVL    R0, NUMBER_BUFFER                          :
                       56              6A  D0 001F5        MOVL    NUMBER_BUFFER, R6                          :
01    A6         04    BE              58  28 001F8        MOVC3   R8, @NUMBER_DESC+4, 1(R6)                  :
                       66              58  90 001FE        MOVB    R8, (R6)                                   :
                 04    AA              02  88 00201 17$:   BISB2   #2, AUGMENTATIONS                          : 1899
                 04    AA              01  8A 00205        BICB2   #1, AUGMENTATIONS                          :
                       50      00      BA  9A 00209        MOVZBL  @NUMBER_BUFFER, R0                         :
                       50              06  C0 0020D        ADDL2   #6, R0                                     :
                       50              04  C6 00210        DIVL2   #4, R0                                     :
                               01      A0  9F 00213        PUSHAB  1(R0)                                      :
                       6B              01  FB 00216        CALLS   #1, DBG$GET_TEMPMEM                        :
                       57              50  D0 00219        MOVL    R0, LINE_ITEM                              :
01    A7  00000000'    EF              06  28 0021C        MOVC3   #6, P.AAF, 1(LINE_ITEM)                    :
                       56              6A  D0 00225        MOVL    NUMBER_BUFFER, R6                          :
                       50              66  9A 00228        MOVZBL  (R6), R0                                   :
07    A7         01    A6              50  28 0022B        MOVC3   R0, 1(R6), 7(LINE_ITEM)                    :
      67               66              06  81 00231        ADDB3   #6, (R6), (LINE_ITEM)                      :
                                     010E  31 00235        BRW     27$                                        :
                 04    AA              02  88 00238 18$:   BISB2   #2, AUGMENTATIONS                          : 1903
                 04    AA              01  8A 0023C        BICB2   #1, AUGMENTATIONS                          :
                       50      00      BA  9A 00240        MOVZBL  @NUMBER_BUFFER, R0                         :
                       50              06  C0 00244        ADDL2   #6, R0                                     :
                       50              04  C6 00247        DIVL2   #4, R0                                     :
                               01      A0  9F 0024A        PUSHAB  1(R0)                                      :
                       6B              01  FB 0024D        CALLS   #1, DBG$GET_TEMPMEM                        :
                       57              50  D0 00250        MOVL    R0, LINE_ITEM                              :
01    A7  00000000'    EF              06  28 00253        MOVC3   #6, P.AAG, 1(LINE_ITEM)                    :
                       56              6A  D0 0025C        MOVL    NUMBER_BUFFER, R6                          :
                       50              66  9A 0025F        MOVZBL  (R6), R0                                   :
07    A7         01    A6              50  28 00262        MOVC3   R0, 1(R6), 7(LINE_ITEM)                    :
      67               66              06  81 00268        ADDB3   #6, (R6), (LINE_ITEM)                      :
                       52      F8      AA  D0 0026C        MOVL    NAME_INDEX, R2                             :
                       32              52  D1 00270        CMPL    R2, #50                                    :
                                       0F  19 00273        BLSS    19$                                        :
                         00028200      8F  DD 00275        PUSHL   #164352                                    :
          00000000G    00              01  FB 0027B        CALLS   #1, LIB$SIGNAL                             :
                                       08  11 00282        BRB     20$                                        :
                 F4  BA42              57  D0 00284 19$:   MOVL    LINE_ITEM, @NAME_VECT[R2]                  :
                               F8      AA  D6 00289        INCL    NAME_INDEX                                 :
                       50      F0      AA  D0 0028C 20$:   MOVL    PATHNAME_DESC, R0                          :
                                       60  96 00290        INCB    (R0)                                       :
                 01    A0              60  90 00292        MOVB    (R0), 1(R0)                                :
                                     0102  31 00296        BRW     31$                                        : 1877
      03         04    AA              02  E0 00299 21$:   BBS     #2, AUGMENTATIONS, 22$                     : 1907
                                     00F6  31 0029E        BRW     30$                                        :
```

```
                    04   AE           14   AA  D0 002A1 22$:   MOVL    LEX_STRING_DESC+4, NUMBER_DESC+4        ; 1908
                         6E           10   AA  B0 002A6        MOVW    LEX_STRING_DESC, NUMBER_DESC            ;
                         01                6E  B1 002AA 23$:   CMPW    NUMBER_DESC, #1                         ;
                                           0D  1B 002AD        BLEQU   24$                                     ;
                         30           04   BE  91 002AF        CMPB    @NUMBER_DESC+4, #48                     ;
                                           07  12 002B3        BNEQ    24$                                     ;
                                           6E  B7 002B5        DECW    NUMBER_DESC                             ;
                                      04   AE  D6 002B7        INCL    NUMBER_DESC+4                           ;
                                           EE  11 002BA        BRB     23$                                     ;
                         58                6E  3C 002BC 24$:   MOVZWL  NUMBER_DESC, R8                         ;
               31   04   AA                05  E1 002BF        BBC     #5, AUGMENTATIONS, 25$                  ;
                         56                6A  D0 002C4        MOVL    NUMBER_BUFFER, TEMP                     ;
                         59                66  9A 002C7        MOVZBL  (TEMP), R9                              ;
                         59                58  C0 002CA        ADDL2   R8, R9                                  ;
               50        59                04  C7 002CD        DIVL3   #4, R9, R0                              ;
                                      01   A0  9F 002D1        PUSHAB  1(R0)                                   ;
                         6B                01  FB 002D4        CALLS   #1, DBG$GET_TEMPMEM                     ;
                         6A                50  D0 002D7        MOVL    R0, NUMBER_BUFFER                       ;
                         50                66  9A 002DA        MOVZBL  (TEMP), R0                              ;
                         57                6A  D0 002DD        MOVL    NUMBER_BUFFER, R7                       ;
          01   A7        01   A6                50  28 002E0        MOVC3   R0, 1(TEMP), 1(R7)                      ;
                         50                66  9A 002E6        MOVZBL  (TEMP), R0                              ;
          01 A047        04   BE                58  28 002E9        MOVC3   R8, @NUMBER_DESC+4, 1(R0)[R7]           ;
                         67                59  90 002F0        MOVB    R9, (R7)                                ;
                                           1D  11 002F3        BRB     26$                                     ;
                    04   AA                20  88 002F5 25$:   BISB2   #32, AUGMENTATIONS                      ;
               50        58                04  C7 002F9        DIVL3   #4, R8, R0                              ;
                                      01   A0  9F 002FD        PUSHAB  1(R0)                                   ;
                         6B                01  FB 00300        CALLS   #1, DBG$GET_TEMPMEM                     ;
                         6A                50  D0 00303        MOVL    R0, NUMBER_BUFFER                       ;
                         57                6A  D0 00306        MOVL    NUMBER_BUFFER, R7                       ;
          01   A7        04   BE                58  28 00309        MOVC3   R8, @NUMBER_DESC+4, 1(R7)               ;
                         67                58  90 0030F        MOVB    R8, (R7)                                ;
                    04   AA                08  88 00312 26$:   BISB2   #8, AUGMENTATIONS                       ; 1909
                    04   AA                04  8A 00316        BICB2   #4, AUGMENTATIONS                       ;
                         50           00   BA  9A 0031A        MOVZBL  @NUMBER_BUFFER, R0                      ;
                         50                07  C0 0031E        ADDL2   #7, R0                                  ;
                         50                04  C6 00321        DIVL2   #4, R0                                  ;
                                      01   A0  9F 00324        PUSHAB  1(R0)                                   ;
                         6B                01  FB 00327        CALLS   #1, DBG$GET_TEMPMEM                     ;
                         57                50  D0 0032A        MOVL    R0, LABEL_ITEM                          ;
          01   A7 00000000'  EF                07  28 0032D        MOVC3   #7, P.AAH, 1(LABEL_ITEM)                ;
                         56                6A  D0 00336        MOVL    NUMBER_BUFFER, R6                       ;
                         50                66  9A 00339        MOVZBL  (R6), R0                                ;
          08   A7        01   A6                50  28 0033C        MOVC3   R0, 1(R6), 8(LABEL_ITEM)                ;
               67        66                07  81 00342        ADDB3   #7, (R6), (LABEL_ITEM)                  ;
                         52           F8   AA  D0 00346 27$:   MOVL    NAME_INDEX, R2                          ;
                         32                52  D1 0034A        CMPL    R2, #50                                 ;
                                           0F  19 0034D        BLSS    28$                                     ;
                              00028200     8F  DD 0034F        PUSHL   #164352                                 ;
                00000000G   00                01  FB 00355        CALLS   #1, LIB$SIGNAL                          ;
                                           08  11 0035C        BRB     29$                                     ;
                    F4 BA42                57  D0 0035E 28$:   MOVL    LABEL_ITEM, @NAME_VECT[R2]              ;
                                      F8   AA  D6 00363        INCL    NAME_INDEX                              ;
                         50           F0   AA  D0 00366 29$:   MOVL    PATHNAME_DESC, R0                       ;
                                           60  96 0036A        INCB    (R0)                                    ;
                    01   A0                60  90 0036C        MOVB    (R0), 1(R0)                             ;
```

```
                  D8    AA       10    AA              08  28 00370          MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC          ; 1910
                                       50        EC    AA  D0 00376          MOVL    INPUT_DESC, R0                                ;
                        51       14    AA        04    A0  C3 0037A          SUBL3   4(R0), LEX_STRING_DESC+4, R1                  ;
                                       52        10    AA  3C 00380          MOVZWL  LEX_STRING_DESC, R2                           ;
                                       51              52  C0 00384          ADDL2   R2, R1                                        ;
                                       60              51  A2 00387          SUBW2   R1, (R0)                                      ;
                                 04    A0        14  BA42  9E 0038A          MOVAB   @LEX_STRING_DESC+4[R2], 4(R0)                 ;
                                 E4    AA        08    AA  D0 00390          MOVL    TOKEN, LAST_TOKEN                             ;
                                                       04  11 00395          BRB     31$                                           ; 1877
                                       50              04  D0 00397 30$:     MOVL    #4, R0                                        ; 1915
                                                           04 0039A          RET                                                   ;
                                 04    AA        40    8F  88 0039B 31$:     BISB2   #64, AUGMENTATIONS                            ; 1919
                                       50              01  D0 003A0          MOVL    #1, R0                                        ; 1921
                                                           04 003A3          RET                                                   ; 1923

; Routine Size:  932 bytes,    Routine Base:  DBG$CODE + 0B9E


; 1810          1924  1
```

```
1 ROUTINE SHORT_SCOPE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Parses global or numeric scopes. On failure, resets input to original state.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS           - Success. Global or numeric scope accepted.
1 !
1 !       STS$K_SEVERE            - Failure. Input not a numeric or global scope
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If successful, produces a complete pathname descriptor for global
1 !       or numeric scope.
1 !
1 !--
2     BEGIN
2
2     LOCAL
2         LENGTH,                 ! Original input length
2         POINTER;                ! Original input pointer
2
2     ! Save the original input
2     !
2     save (length, pointer);
2
2
2     ! Obtain the first token and check for integer or backslash
2     !
2     get_token;
2
2     CASE .token FROM dbg$k_tok_lowest TO dbg$k_tok_highest
2         OF
2         SET
2
2         [dbg$k_tok_bs] : ! Global scope ?
3             BEGIN
3             advance;
```

```
 3                  get_token;
 3
 3                  IF .token EQL dbg$k_tok_null OR .token EQL dbg$k_tok_inval
 3                  THEN
 4                      BEGIN
 4
 4                      ! Yes, global scope.
 4                      !
 4                      add_null_id;
 4                      END
 3                  ELSE
 4                      BEGIN
 4
 4                      ! No. Restore input.
 4                      !
 4                      restore (.length, .pointer);
 4                      RETURN sts$k_severe;
 3                      END;
 2                  END;
 2
 2              [dbg$k_tok_int] :       ! Numeric scope ?
 3                  BEGIN
 3                  advance;
 3                  get_token;
 3
 3                  IF .token EQL dbg$k_tok_inval OR .token EQL dbg$k_tok_null
 3                  THEN
 4                      BEGIN
 4
 4                      ! Yes, numeric scope
 4                      !
 4                      restore (.length, .pointer);
 4                      get_token;
 4                      add_numeric_scope;
 4                      advance;
 4                      END
 3                  ELSE
 4                      BEGIN
 4
 4                      ! No, restore and fail
 4                      !
 4                      restore (.length, .pointer);
 4                      RETURN sts$k_severe;
 3                      END;
 2                  END;
 2
 2              [INRANGE, OUTRANGE] :
 3                  BEGIN
 3                  RETURN sts$k_severe;
 2                  END;
 2
 2              TES;
 2
 2          RETURN sts$k_success;
 2
 1          END;        ! End of short_scope
```

```
                                                              .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                            0D  00018 P.AAI:   .BYTE   13                                          ;

                                                              .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                           07FC 00000 SHORT_SCOPE:
                                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10             ; 1925
                      5A 00000000'  EF  9E 00002              MOVAB   NULL_STRING, R10                             ;
                      59 00000000'  EF  9E 00009              MOVAB   TOKEN, R9                                    ;
                      5E            10  C2 00010              SUBL2   #16, SP                                      ;
                      50        E4  A9  D0 00013              MOVL    INPUT_DESC, R0                               ; 1968
                      58            60  3C 00017              MOVZWL  (R0), LENGTH                                 ;
                      57        04  A0  D0 0001A              MOVL    4(R0), POINTER                               ;
                                    59  DD 0001E              PUSHL   R9                                           ;
                                08  A9  9F 00020              PUSHAB  LEX_STRING_DESC                              ;
                                    50  DD 00023              PUSHL   R0                                           ;
                  04  B9            03  FB 00025              CALLS   #3, @TOKEN_SCANNER_ADDR                      ;
                      06            69  D1 00029              CMPL    TOKEN, #6                                    ;
                                    09  12 0002C              BNEQ    1$                                           ;
                      09        08  A9  B1 0002E              CMPW    LEX_STRING_DESC, #9                          ;
                                    03  1B 00032              BLEQU   1$                                           ;
                      69            01  D0 00034              MOVL    #1, TOKEN                                    ;
                      56            69  D0 00037 1$:          MOVL    TOKEN, R6                                    ; 1975
            09        00            56  CF 0003A              CASEL   R6, #0, #9                                   ;
016F        016F     016F         016F     0003E 2$:          .WORD   13$-2$,-                                     ;
016F        007A     016F         0017     00046                      13$-2$,-                                     ;
                     016F         016F     0004E                      13$-2$,-                                     ;
                                                                      13$-2$,-                                     ;
                                                                      3$-2$,-                                      ;
                                                                      13$-2$,-                                     ;
                                                                      8$-2$,-                                      ;
                                                                      13$-2$,-                                     ;
                                                                      13$-2$,-                                     ;
                                                                      13$-2$                                       ;
                                  0158  31 00052              BRW     13$                                          ; 2030
       D0     A9      08  A9            08  28 00055 3$:      MOVC3   #8, LEX_STRING_DESC, LAST_TOKEN_DESC         ; 1980
                      50        E4  A9  D0 0005B              MOVL    INPUT_DESC, R0                               ;
              51      0C  A9    04  A0  C3 0005F              SUBL3   4(R0), LEX_STRING_DESC+4, R1                 ;
                      52        08  A9  3C 00065              MOVZWL  LEX_STRING_DESC, R2                          ;
                      51            52  C0 00069              ADDL2   R2, R1                                       ;
                      60            51  A2 0006C              SUBW2   R1, (R0)                                     ;
                  04  A0     0C B942  9E 0006F              MOVAB   @LEX_STRING_DESC+4[R2], 4(R0)                ;
                  DC  A9            56  D0 00075              MOVL    R6, LAST_TOKEN                               ;
                                    59  DD 00079              PUSHL   R9                                           ; 1981
                                08  A9  9F 0007B              PUSHAB  LEX_STRING_DESC                              ;
                                    50  DD 0007E              PUSHL   R0                                           ;
                  04  B9            03  FB 00080              CALLS   #3, @TOKEN_SCANNER_ADDR                      ;
                      06            69  D1 00084              CMPL    TOKEN, #6                                    ;
                                    09  12 00087              BNEQ    4$                                           ;
                      09        08  A9  B1 00089              CMPW    LEX_STRING_DESC, #9                          ;
                                    03  1B 0008D              BLEQU   4$                                           ;
                      69            01  D0 0008F              MOVL    #1, TOKEN                                    ;
```

```
                       50             69  D0 00092 4$:     MOVL     TOKEN, R0                                          ; 1984
                                      05  13 00095         BEQL     5$                                                 ;
                       01             50  D1 00097         CMPL     R0, #1                                             ;
                                      15  12 0009A         BNEQ     6$                                                 ;
                  EC   B9             6A  9E 0009C 5$:     MOVAB    NULL_STRING, @NAME_VECT                            ; 1986
                       50        E8   A9  D0 000A0         MOVL     PATHNAME_DESC, R0                                  ;
                                      60  96 000A4         INCB     (R0)                                               ;
                  01   A0             60  90 000A6         MOVB     (R0), 1(R0)                                        ;
                  F0   A9             01  D0 000AA         MOVL     #1, NAME_INDEX                                     ;
                                    0100  31 000AE         BRW      14$                                                ; 1984
                       50        E4   A9  D0 000B1 6$:     MOVL     INPUT_DESC, R0                                     ; 1997
                                    00EE  31 000B5 7$:     BRW      12$                                                ;
D0     A9         08   A9             08  28 000B8 8$:     MOVC3    #8, LEX_STRING_DESC, LAST_TOKEN_DESC               ; 2003
                       50        E4   A9  D0 000BE         MOVL     INPUT_DESC, R0                                     ;
       51         0C   A9        04   A0  C3 000C2         SUBL3    4(R0), LEX_STRING_DESC+4, R1                       ;
                       52        08   A9  3C 000C8         MOVZWL   LEX_STRING_DESC, R2                                ;
                       51             52  C0 000CC         ADDL2    R2, R1                                             ;
                       60             51  A2 000CF         SUBW2    R1, (R0)                                           ;
                  04   A0        0C B942  9E 000D2         MOVAB    @LEX_STRING_DESC+4[R2], 4(R0)                      ;
                  DC   A9             56  D0 000D8         MOVL     R6, LAST_TOKEN                                     ;
                                      59  DD 000DC         PUSHL    R9                                                 ; 2004
                                 08   A9  9F 000DE         PUSHAB   LEX_STRING_DESC                                    ;
                                      50  DD 000E1         PUSHL    R0                                                 ;
                  04   B9             03  FB 000E3         CALLS    #3, @TOKEN_SCANNER_ADDR                            ;
                       06             69  D1 000E7         CMPL     TOKEN, #6                                          ;
                                      09  12 000EA         BNEQ     9$                                                 ;
                       09        08   A9  B1 000EC         CMPW     LEX_STRING_DESC, #9                                ;
                                      03  1B 000F0         BLEQU    9$                                                 ;
                       69             01  D0 000F2         MOVL     #1, TOKEN                                          ;
                       50        E4   A9  D0 000F5 9$:     MOVL     INPUT_DESC, R0                                     ; 2013
                       01             69  D1 000F9         CMPL     TOKEN, #1                                          ; 2007
                                      04  13 000FC         BEQL     10$                                                ;
                                      69  D5 000FE         TSTL     TOKEN                                              ;
                                      B3  12 00100         BNEQ     7$                                                 ;
                       60             58  B0 00102 10$:    MOVW     LENGTH, (R0)                                       ; 2013
                  04   A0             57  D0 00105         MOVL     POINTER, 4(R0)                                     ;
                                      59  DD 00109         PUSHL    R9                                                 ;
                                 08   A9  9F 0010B         PUSHAB   LEX_STRING_DESC                                    ;
                                      50  DD 0010E         PUSHL    R0                                                 ;
                  04   B9             03  FB 00110         CALLS    #3, @TOKEN_SCANNER_ADDR                            ;
                       06             69  D1 00114         CMPL     TOKEN, #6                                          ;
                                      09  12 00117         BNEQ     11$                                                ;
                       09        08   A9  B1 00119         CMPW     LEX_STRING_DESC, #9                                ;
                                      03  1B 0011D         BLEQU    11$                                                ;
                       69             01  D0 0011F         MOVL     #1, TOKEN                                          ;
                  EC   B9             6A  9E 00122 11$:    MOVAB    NULL_STRING, @NAME_VECT                            ; 2014
                       50        E8   A9  D0 00126         MOVL     PATHNAME_DESC, R0                                  ;
                                      60  96 0012A         INCB     (R0)                                               ;
                  01   A0             60  90 0012C         MOVB     (R0), 1(R0)                                        ;
                  F0   A9             01  D0 00130         MOVL     #1, NAME_INDEX                                     ;
                  FC   A9             10  88 00134         BISB2    #16, AUGMENTATIONS                                 ;
04     AE         08   A9             01  A1 00138         ADDW3    #1, LEX_STRING_DESC, NUMBER_DESC                   ;
                       50        04   AE  3C 0013E         MOVZWL   NUMBER_DESC, R0                                    ;
                       50             04  C6 00142         DIVL2    #4, R0                                             ;
                                 01   A0  9F 00145         PUSHAB   1(R0)                                              ;
          00000000G    00             01  FB 00148         CALLS    #1, DBG$GET_TEMPMEM                                ;
                       56             50  D0 0014F         MOVL     R0, NUM_BUF                                        ;
                  66        0C   B9         08    A9   28 00152            MOVC3    LEX_STRING_DESC, @LEX_STRING_DESC+4, -  ;
                                                                                    (NUM_BUF)                                ;
                                 63         18    AA   90 00158            MOVB     P.AAI, (POINTER)                          ;
                            08   AE               56   D0 0015C            MOVL     NUM_BUF, NUMBER_DESC+4                    ;
                                            E0    A9   9F 00160            PUSHAB   DUMMY                                     ;
                                            04    AE   9F 00163            PUSHAB   NUMBER                                    ;
                                            0C    AE   9F 00166            PUSHAB   NUMBER_DESC                               ;
                     00000000G   00               03   FB 00169            CALLS    #3, DBG$NSAVE_DECIMAL_INTEGER             ;
                                 3A               50   E9 00170            BLBC     R0, 13$                                   ;
                                 50         E8    A9   D0 00173            MOVL     PATHNAME_DESC, R0                         ;
                            02   A0         F0    A9   90 00177            MOVB     NAME_INDEX, 2(R0)                         ;
                            04   A0               6E   D0 0017C            MOVL     NUMBER, 4(R0)                             ;
              D0    A9      08   A9               08   28 00180            MOVC3    #8, LEX_STRING_DESC, LAST_TOKEN_DESC      ; 2015
                                 50         E4    A9   D0 00186            MOVL     INPUT_DESC, R0                            ;
                  51        0C   A9         04    A0   C3 0018A            SUBL3    4(R0), LEX_STRING_DESC+4, R1              ;
                                 52         08    A9   3C 00190            MOVZWL   LEX_STRING_DESC, R2                       ;
                                 51               52   C0 00194            ADDL2    R2, R1                                    ;
                                 60               51   A2 00197            SUBW2    R1, (R0)                                  ;
                            04   A0         0C  B942   9E 0019A            MOVAB    @LEX_STRING_DESC+4[R2], 4(R0)             ;
                            DC   A9               69   D0 001A0            MOVL     TOKEN, LAST_TOKEN                         ;
                                                  0B   11 001A4            BRB      14$                                       ; 2007
                                 60               58   B0 001A6 12$:       MOVW     LENGTH, (R0)                              ; 2023
                            04   A0               57   D0 001A9            MOVL     POINTER, 4(R0)                            ;
                                 50               04   D0 001AD 13$:       MOVL     #4, R0                                    ; 2024
                                                       04 001B0            RET                                                ;
                                 50               01   D0 001B1 14$:       MOVL     #1, R0                                    ; 2035
                                                       04 001B4            RET                                                ; 2037

; Routine Size:  437 bytes,    Routine Base:  DBG$CODE + 0F42


; 1925          2038  1
```

```
GLOBAL ROUTINE CHECK_PATHNAME  : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine examines a completed pathname descriptor and classifies its
!       type by setting the value state to:
!
!       dbg$k_reg       :       register reference (item count is 0)
!
!       dbg$k_line      :       line number reference (not a data item)
!
!       dbg$k_label     :       numeric label reference (not a data item)
!
!       dbg$k_pn        :       data or lexical item reference
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       The pathname descriptor constructed by parse_pathname.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       NONE
!
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       The value state is set according to the type of pathname descriptor examined.
!
!--
    BEGIN

    BIND
        LINE_STG        = UPLIT BYTE ('%LINE'),
        LABEL_STG       = UPLIT BYTE ('%LABEL');

    LOCAL
        NEW_STRING      : REF VECTOR [,BYTE],
        STRING          : REF VECTOR [,BYTE];   ! String vector

    string = .name_vect [.pathname_desc [pth$b_totcnt] - 1];

    ! If language is C, then copy and upcase the string.
    !
```

```
; 1984          2096  2     IF .dbg$gb_language EQL dbg$k_c
; 1985          2097  2     THEN
; 1986          2098  3         BEGIN
; 1987          2099  3         new_string = dbg$get_tempmem((.string[0]/4)+1);
; 1988          2100  3         ch$move(.string[0]+1, .string, .new_string);
; 1989          2101  3         INCR i FROM 1 TO .new_string[0] DO
; 1990          2102  3             IF .new_string[.i] GEQ 'a' AND .new_string[.i] LEQ 'z'
; 1991          2103  3             THEN
; 1992          2104  3                 new_string[.i] = .new_string[.i] - ('a' - 'A');
; 1993          2105  3         string = .new_string;
; 1994          2106  2         END;
; 1995          2107  2
; 1996          2108  2
; 1997          2109  2     ! Set the value state by examining the completed pathname descriptor
; 1998          2110  2     !
; 1999          2111  2     SELECTONE true
; 2000          2112  2         OF
; 2001          2113  2         SET
; 2002          2114  2
; 2003          2115  2         [.pathname_desc [pth$b_totcnt] EQL 0] : value_state = dbg$k_reg;
; 2004          2116  2
; 2005          2117  2         [ch$find_sub (.string [0], string [1], 5, line_stg) NEQA 0] : value_state = dbg$k_line;
; 2006          2118  2
; 2007          2119  2         [ch$find_sub (.string [0], string [1], 6, label_stg) NEQA 0] : value_state = dbg$k_label;
; 2008          2120  2
; 2009          2121  2         [OTHERWISE] : value_state = dbg$k_pn;
; 2010          2122  2
; 2011          2123  2         TES;
; 2012          2124  2
; 2013          2125  2     RETURN;
; 2014          2126  2
; 2015          2127  1     END;                ! End of check_pathname


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                  45  4E  49  4C  25  00019 P.AAJ:   .ASCII  \%LINE\                                    ;
              4C  45  42  41  4C  25  0001E P.AAK:   .ASCII  \%LABEL\                                   ;

                                                LINE_STG=            P.AAJ
                                                LABEL_STG=           P.AAK


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                              01FC  00000              .ENTRY  CHECK_PATHNAME, Save R2,R3,R4,R5,R6,R7,R8    ; 2039
                58 00000000'  EF  9E  00002            MOVAB   VALUE_STATE, R8                              ;
                50         F4  B8  9A  00009            MOVZBL  @PATHNAME_DESC, R0                           ; 2091
                50         F8 B840  DE  0000D            MOVAL   @NAME_VECT[R0], R0                           ;
                56         FC  A0  D0  00012            MOVL    -4(R0), STRING                               ;
                07 00000000G  00  91  00016            CMPB    DBG$GB_LANGUAGE, #7                          ; 2096
                              3C  12  0001D            BNEQ    3$                                           ;
                50             66  9A  0001F            MOVZBL  (STRING), R0                                 ; 2099
                50             04  C6  00022            DIVL2   #4, R0                                       ;
                         01    A0  9F  00025            PUSHAB  1(R0)                                        ;
       00000000G  00          01  FB  00028            CALLS   #1, DBG$GET_TEMPMEM                          ;
```

```
                                57        50  D0 0002F        MOVL     R0, NEW_STRING                          ;
                                50        66  9A 00032        MOVZBL   (STRING), R0                            ; 2100
                                          50  D6 00035        INCL     R0                                      ;
                    67          66        50  28 00037        MOVC3    R0, (STRING), (NEW_STRING)              ;
                                51        67  9A 0003B        MOVZBL   (NEW_STRING), R1                        ; 2101
                                          50  D4 0003E        CLRL     I                                       ;
                                          12  11 00040        BRB      2$                                      ;
                         61     8F      6047  91 00042  1$:   CMPB     (I)[NEW_STRING], #97                    ; 2102
                                          0B  1F 00047        BLSSU    2$                                      ;
                         7A     8F      6047  91 00049        CMPB     (I)[NEW_STRING], #122                   ;
                                          04  1A 0004E        BGTRU    2$                                      ;
                              6047        20  82 00050        SUBB2    #32, (I)[NEW_STRING]                    ; 2104
                    EA          50        51  F3 00054  2$:   AOBLEQ   R1, I, 1$                               ; 2102
                                56        57  D0 00058        MOVL     NEW_STRING, STRING                      ; 2105
                                   F4     B8  95 0005B  3$:   TSTB     @PATHNAME_DESC                          ; 2115
                                          04  12 0005E        BNEQ     4$                                      ;
                                68        01  D0 00060        MOVL     #1, VALUE_STATE                         ;
                                          04     00063        RET                                              ;
                                50        66  9A 00064  4$:   MOVZBL   (STRING), R0                            ; 2117
     01    A6          50 00000000'  EF   05  39 00067        MATCHC   #5, LINE_STG, R0, 1(STRING)             ;
                                          03  13 00071        BEQL     5$                                      ;
                                53        05  D0 00073        MOVL     #5, R3                                  ;
                                53        05  C2 00076  5$:   SUBL2    #5, R3                                  ;
                                          04  13 00079        BEQL     6$                                      ;
                                68        02  D0 0007B        MOVL     #2, VALUE_STATE                         ;
                                          04     0007E        RET                                              ;
                                50        66  9A 0007F  6$:   MOVZBL   (STRING), R0                            ; 2119
     01    A6          50 00000000'  EF   06  39 00082        MATCHC   #6, LABEL_STG, R0, 1(STRING)            ;
                                          03  13 0008C        BEQL     7$                                      ;
                                53        06  D0 0008E        MOVL     #6, R3                                  ;
                                53        06  C2 00091  7$:   SUBL2    #6, R3                                  ;
                                          04  13 00094        BEQL     8$                                      ;
                                68        03  D0 00096        MOVL     #3, VALUE_STATE                         ;
                                          04     00099        RET                                              ;
                                          68  D4 0009A  8$:   CLRL     VALUE_STATE                             ; 2121
                                          04     0009C        RET                                              ; 2127

; Routine Size:  157 bytes,    Routine Base:  DBG$CODE + 10F7


; 2016          2128  1
```

```
1 GLOBAL ROUTINE DBG$NPATHDESC_TO_CS (PATHNAME_DESC, COUNTED_STRING) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine accepts a pathname descriptor and translates the contents ot
1 !       the descriptor into a printable form. That is, the names and optional
1 !       invocation number contained within the pathname descriptor are formatted
1 !       into one long counted string.
1 !
1 !       This routine will oroduce the translation for any pathname descriptor which
1 !       describes a legal scope including '\' and numeric scopes.
1 !
1 !       Pathnames in which the first two names are the same are modified to
1 !       output the name only once (situations where routine and module names
1 !       are the same).
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PN_DESC                 - A longword containing the address of a pathnaem
1 !                                 descriptor
1 !
1 !       COUNTED_STRING          - The address of a longword to contain the address
1 !                                 of a counted string representing the translation
1 !                                 of the contents of the pathname descriptor
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       The translated pathname string
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       This routine will produce a SIGNAL for certain circumstances.
1 !
1 !--
2     BEGIN
2
2     MAP
2         PATHNAME_DESC    : REF pth$pathname;
2
2     LOCAL
2         SAVE_STRING,                        ! Pointer to original string
2         PATH_STRING      : REF VECTOR [,BYTE],  ! Result buffer
2         NAME_VECT        : REF VECTOR,          ! Vector of pointers to name strings
2         NAME             : REF VECTOR [,BYTE],  ! Name counted string
```

```
        INDEX,                                  ! Index into name_vect
        SOURCE_DESC     : dbg$stg_desc,         ! Source descriptor
        TARGET_DESC     : dbg$stg_desc,         ! Target descriptor
        RESULT_LENGTH   : WORD,                 ! Length of string after FAOing
        NEXT_CHAR,                              ! Pointer into result string
        SIZE;                                   ! Number of bytes needed for result buffer

    save_string = 0;

    ! Line up the name vector
    !
    name_vect = pathname_desc [pth$a_pathvector];


    ! Look for an invocation number. If there is one, go ahead and add the number
    ! to the correct name string. We save the original name string so that we may restore it.
    !
    IF .pathname_desc [pth$b_locinvoc] NEQ 0
    THEN
        BEGIN

        ! Recover the name string
        !
        name = .name_vect [.pathname_desc [pth$b_locinvoc] - 1];
        save_string = .name;


        ! Allocate enough storage to concatenate the number to the string
        !
        path_string = dbg$get_tempmem((.name [0] + 24) / %UPVAL);


        ! Copy the name string
        !
        IF .name [0] NEQ 0
        THEN
            BEGIN
            next_char = ch$move (.name [0], name [1], path_string [1]);
            source_desc [dsc$a_pointer] = UPLIT BYTE (' !UL');
            source_desc [dsc$w_length] = 4;
            END
        ELSE
            BEGIN
            next_char = path_string [1];
            source_desc [dsc$a_pointer] = UPLIT BYTE ('!UL');
            source_desc [dsc$w_length] = 3;
            END;


        ! Append the invocation number
        !
        target_desc [dsc$a_pointer] = .next_char;
        target_desc [dsc$w_length] = 23;

        sys$fao (source_desc, result_length, target_desc, .pathname_desc [pth$l_invocnum]);

```

```
            ! Update the copie's length
            !
            path_string [0] = .name [0] + .result_length;


            ! Point to the copy
            !
            name_vect [.pathname_desc [pth$b_locinvoc] - 1] = .path_string;
            END;


        ! Figure out how much space will be needed to hold the entire string
        !
        size = 0;
        INCR index FROM 0 TO .pathname_desc [pth$b_totcnt] - 1
        DO
            BEGIN
            name = .name_vect [.index];
            size = .size + .name [0] + 1;   ! One for '\'
            END;


        ! Allocate enough storage to hold the string plus one byte for the length
        !
        path_string = dbg$get_tempmem((.size / %UPVAL) + 2);


        ! Now we're ready to append all the name strings into one string. First
        ! check for the special case of the global scope, '\'.
        !
        name = .name_vect [0];
        IF .name [0] EQL 0 AND .pathname_desc [pth$b_locinvoc] EQL 0
        THEN
            BEGIN

            ! Global scope or global reference
            !
            ch$move (1, UPLIT BYTE ('\'), path_string [1]);
            result_length = 1;
            IF .pathname_desc [pth$b_totcnt] GTR 1
            THEN
                BEGIN
                name = .name_vect [1];
                ch$move (.name [0], name [1], path_string [2]);
                result_length = .result_length + .name [0];
                END;
            END
        ELSE
            BEGIN
            LOCAL
                I,
                NAME_1      : REF VECTOR [,BYTE],
                NAME_2      : REF VECTOR [,BYTE];

            ! Loop, adding all the name strings.
            !
            result_length = 0;
```

```
3          next_char = path_string [1];
3
3
3          ! We do not want to ouput the same name twice. Check to see if the
3          ! first name and the second name are the same. If they are, skip over
3          ! the first name.
3          !
3          i = 0;
3          IF .pathname_desc [pth$b_totcnt] GEQ 2
3          THEN
4              BEGIN
4              name_1 = .name_vect [0];
4              name_2 = .name_vect [1];
4              IF ch$eql (.name_1 [0], name_1 [1], .name_2 [0], name_2 [1])
4              THEN
4                  i = 1;
3              END;
3
3          INCR index FROM .i TO .pathname_desc [pth$b_totcnt] - 1
3          DO
4              BEGIN
4              name = .name_vect [.index];
4              next_char = ch$move (.name [0], name [1], .next_char);
4              result_length = .result_length + .name [0];
4
4
4              ! If there is another name string, we add a '\'
4              !
4              IF .index LSS .pathname_desc [pth$b_totcnt] - 1
4              THEN
5                  BEGIN
5                  IF .index LSS .pathname_desc[pth$b_pathcnt] - 1
5                  THEN
5                      next_char = ch$move (1, UPLIT BYTE ('\'), .next_char)
5                  ELSE
5                      next_char = ch$move (1, UPLIT BYTE ('.'), .next_char);
5                  result_length = .result_length + 1;
4                  END;
3              END;
2          END;
2
2
2      ! Fill in the count byte. Check for overflow.
2      !
2      path_string [0] = (IF .result_length GTR 255 THEN 255 ELSE .result_length);
2
2
2      ! Restore the saved string if there is one.
2      !
2      IF .save_string NEQA 0
2      THEN
2          name_vect [.pathname_desc [pth$b_locinvoc] - 1] = .save_string;
2
2
2      ! Return the counted string
2      !
2      .counted_string = .path_string;
```

```
; 2246          2357  2
; 2247          2358  2       RETURN;
; 2248          2359  2
; 2249          2360  1       END;                    ! End of DBG$NPATHDESC_TO_CS


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                         4C  55  21  20  00024 P.AAL:   .ASCII  \ !UL\                                      ;
                             4C  55  21  00028 P.AAM:   .ASCII  \!UL\                                       ;
                                     5C  0002B P.AAN:   .ASCII  <92>                                        ;
                                     5C  0002C P.AAO:   .ASCII  <92>                                        ;
                                     2E  0002D P.AAP:   .ASCII  \.\                                         ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                   0FFC  00000          .ENTRY  DBG$NPATHDESC_TO_CS, Save R2,R3,R4,R5,R6,-  ; 2129
                                                                R7,R8,R9,R10,R11                            ;
                         5E          20  C2 00002       SUBL2   #32, SP                                     ;
                                     7E  D4 00005       CLRL    SAVE_STRING                                 ; 2193
                         5A      04  AC  D0 00007       MOVL    PATHNAME_DESC, R10                          ; 2197
                         57      08  AA  9E 0000B       MOVAB   8(R10), NAME_VECT                           ;
                         02          AA  95 0000F       TSTB    2(R10)                                      ; 2203
                                     76  13 00012       BEQL    3$                                          ;
                         56      02  AA  9A 00014       MOVZBL  2(R10), R6                                  ; 2209
                         58   FC A746    D0 00018       MOVL    -4(NAME_VECT)[R6], NAME                     ;
                         6E          58  D0 0001D       MOVL    NAME, SAVE_STRING                           ; 2210
                         50          68  9A 00020       MOVZBL  (NAME), R0                                  ; 2215
                         50          18  C0 00023       ADDL2   #24, R0                                     ;
                 7E      50          04  C7 00026       DIVL3   #4, R0, -(SP)                               ;
                  00000000G  00      01  FB 0002A       CALLS   #1, DBG$GET_TEMPMEM                         ;
                         59          50  D0 00031       MOVL    R0, PATH_STRING                             ;
                                     68  95 00034       TSTB    (NAME)                                      ; 2220
                                     1B  13 00036       BEQL    1$                                          ;
                         50          68  9A 00038       MOVZBL  (NAME), R0                                  ; 2223
          01     A9      01  A8      50  28 0003B       MOVC3   R0, 1(NAME), 1(PATH_STRING)                 ;
                         04  AE      53  D0 00041       MOVL    R3, NEXT_CHAR                               ;
                         1C  AE 00000000'  EF  9E 00045 MOVAB   P.AAL, SOURCE_DESC+4                        ; 2224
                         18  AE      04  B0 0004D       MOVW    #4, SOURCE_DESC                             ; 2225
                                     11  11 00051       BRB     2$                                          ; 2220
                         04  AE      01  A9  9E 00053 1$:  MOVAB   1(PATH_STRING), NEXT_CHAR                 ; 2229
                         1C  AE 00000000'  EF  9E 00058 MOVAB   P.AAM, SOURCE_DESC+4                        ; 2230
                         18  AE      03  B0 00060       MOVW    #3, SOURCE_DESC                             ; 2231
                         10  AE      04  AE  D0 00064 2$:  MOVL    NEXT_CHAR, TARGET_DESC+4                  ; 2237
                         0C  AE      17  B0 00069       MOVW    #23, TARGET_DESC                            ; 2238
                             04  AA  DD 0006D           PUSHL   4(R10)                                      ; 2240
                             10  AE  9F 00070           PUSHAB  TARGET_DESC                                 ;
                             10  AE  9F 00073           PUSHAB  RESULT_LENGTH                               ;
                             24  AE  9F 00076           PUSHAB  SOURCE_DESC                                 ;
                  00000000G  9F      04  FB 00079       CALLS   #4, @#SYS$FAO                               ;
                 69      68      08  AE  81 00080       ADDB3   RESULT_LENGTH, (NAME), (PATH_STRING)        ; 2245
                      FC A746        59  D0 00085       MOVL    PATH_STRING, -4(NAME_VECT)[R6]              ; 2250
                                 50  D4 0008A 3$:       CLRL    SIZE                                        ; 2256
                         5B          6A  9A 0008C       MOVZBL  (R10), R11                                  ; 2257
```

```
                                     51                 01  CE 0008F        MNEGL   #1, INDEX                                        ; 2260
                                                        0C  11 00092        BRB     5$
                                     58               6741  D0 00094 4$:    MOVL    (NAME_VECT)[INDEX], NAME
                                     52                 68  9A 00098        MOVZBL  (NAME), R2                                       ; 2261
                                     50        01     A240  9E 0009B        MOVAB   1(R2)[SIZE], SIZE
                        F0           51                 5B  F2 000A0 5$:    AOBLSS  R11, INDEX, 4$                                   ; 2257
                                     50                 04  C6 000A4        DIVL2   #4, R0                                           ; 2267
                                               02       A0  9F 000A7        PUSHAB  2(R0)
                           00000000G 00                 01  FB 000AA        CALLS   #1, DBG$GET_TEMPMEM
                                     59                 50  D0 000B1        MOVL    R0, PATH_STRING
                                     58                 67  D0 000B4        MOVL    (NAME_VECT), NAME                                ; 2273
                                                        68  95 000B7        TSTB    (NAME)                                           ; 2274
                                                        2C  12 000B9        BNEQ    7$
                                               02       AA  95 000BB        TSTB    2(R10)
                                                        27  12 000BE        BNEQ    7$
                                 01  A9 00000000'       EF  90 000C0        MOVB    P.AAN, 1(PATH_STRING)                            ; 2280
                                 08  AE                 01  B0 000C8        MOVW    #1, RESULT_LENGTH                                ; 2281
                                 01                     6A  91 000CC        CMPB    (R10), #1                                        ; 2282
                                                        14  1B 000CF        BLEQU   6$
                                     58        04       A7  D0 000D1        MOVL    4(NAME_VECT), NAME                               ; 2285
                                     50                 68  9A 000D5        MOVZBL  (NAME), R0                                       ; 2286
                  02    A9       01  A8                 50  28 000D8        MOVC3   R0, 1(NAME), 2(PATH_STRING)
                                     50                 68  9A 000DE        MOVZBL  (NAME), R0                                       ; 2287
                                 08  AE                 50  A0 000E1        ADDW2   R0, RESULT_LENGTH
                                                        75  11 000E5 6$:    BRB     13$                                              ; 2274
                                               08       AE  B4 000E7 7$:    CLRW    RESULT_LENGTH                                    ; 2299
                                 04  AE        01       A9  9E 000EA        MOVAB   1(PATH_   RING), NEXT_CHAR                       ; 2300
                                                        56  D4 000EF        CLRL    I                                                ; 2307
                                 02                     6A  91 000F1        CMPB    (R10), #2                                        ; 2308
                                                        1A  1F 000F4        BLSSU   8$
                                     51                 67  D0 000F6        MOVL    (NAME_VECT), NAME_1                              ; 2311
                                     50        04       A7  D0 000F9        MOVL    4(NAME_VECT), NAME_2                             ; 2312
                                     53                 61  9A 000FD        MOVZBL  (NAME_1), R3                                     ; 2313
                                     52                 60  9A 00100        MOVZBL  (NAME_2), R2
        52              00       01  A1                 53  2D 00103        CMPC5   R3, 1(NAME_1), #0, R2, 1(NAME_2)
                                               01       A0     00109
                                                        03  12 0010B        BNEQ    8$
                                     56                 01  D0 0010D        MOVL    #1, I                                            ; 2315
                                                        56  D7 00110 8$:    DECL    INDEX                                            ; 2318
                                                        44  11 00112        BRB     12$
                                     58               6746  D0 00114 9$:    MOVL    (NAME_VECT)[INDEX], NAME                         ; 2321
                                     50                 68  9A 00118        MOVZBL  (NAME), R0                                       ; 2322
                  04    BE       01  A8                 50  28 0011B        MOVC3   R0, 1(NAME), @NEXT_CHAR
                                 04  AE                 53  D0 00121        MOVL    R3, NEXT_CHAR
                                     50                 68  9A 00125        MOVZBL  (NAME), R0                                       ; 2323
                                 08  AE                 50  A0 00128        ADDW2   R0, RESULT_LENGTH
                                     50        FF       AB  9E 0012C        MOVAB   -1(R11), R0                                      ; 2328
                                     50                 56  D1 00130        CMPL    INDEX, R0
                                                        23  18 00133        BGEQ    12$
                                     50        01       AA  9A 00135        MOVZBL  1(R10), R0                                       ; 2331
                                                        50  D7 00139        DECL    R0
                                     50                 56  D1 0013B        CMPL    INDEX, R0
                                                        0A  18 0013E        BGEQ    10$
                                 04  BE 00000000'       EF  90 00140        MOVB    P.AAO, @NEXT_CHAR                                ; 2333
                                                        08  11 00148        BRB     11$
                                 04  BE 00000000'       EF  90 0014A 10$:   MOVB    P.AAP, @NEXT_CHAR                                ; 2335
                                               04       AE  D6 00152 11$:   INCL    NEXT_CHAR
```

```
                                         08   AE  B6 00155        INCW    RESULT_LENGTH                                ; 2336
                     B8          56           5B  F2 00158 12$:   AOBLSS  R11, INDEX, 9$                               ; 2318
                          00FF   8F      08   AE  B1 0015C 13$:   CMPW    RESULT_LENGTH, #255                          ; 2344
                                              06  1B 00162        BLEQU   14$                                          ;
                                 50      FF   8F  9A 00164        MOVZBL  #255, R0                                     ;
                                              04  11 00168        BRB     15$                                          ;
                                 50      08   AE  3C 0016A 14$:   MOVZWL  RESULT_LENGTH, R0                            ;
                                 69           50  90 0016E 15$:   MOVB    R0, (PATH_STRING)                            ;
                                              6E  D5 00171        TSTL    SAVE_STRING                                  ; 2349
                                              09  13 00173        BEQL    16$                                          ;
                                 50      02   AA  9A 00175        MOVZBL  2(R10), R0                                   ; 2351
                            FC A740           6E  D0 00179        MOVL    SAVE_STRING, -4(NAME_VECT)[R0]               ;
                            08   BC           59  D0 0017E 16$:   MOVL    PATH_STRING, @COUNTED_STRING                 ; 2356
                                                  04 00182        RET                                                  ; 2360

; Routine Size:  387 bytes,    Routine Base:  DBG$CODE + 1194


; 2250          2361  1
```

```
1 ROUTINE SCOPE_SCANNER (INPUT_DESC, LEX_DESC, TOKEN) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Lexical scanner for the parsing of scopes. This routine supplies
1 !       tokens to the pathname parser when a scope is to be parsed. It plays
1 !       the part of a language specific lexical scanner and its address is
1 !       supplied to the pathname parser by dbg$nparse_scope_list.
1 !
1 !       The tokens returned by this routine are limited to:
1 !
1 !       dbg$k_tok_null, dbg$k_tok_inval, dbg$k_tok_line, dbg$k_tok_label,
1 !       dbg$k_tok_int, dbg$k_tok_id, dbg$k_tok_dot, and dbg$k_tok_bs.
1 !
1 !       Note that unlike the acutual language specific scanners, this routine does
1 !       not return a token for %register since these are invalid in a scope.
1 !
1 !       The input line is NOT updated after a token is recognized. The caller
1 !       of this routine is responsible for updating the input line by
1 !       using the information in the lexical string descriptor.
1 !
1 !       The input line is assumed to be terminated with a <CR>.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       INPUT_DESC              - A longword containing the address of a standard
1 !                                 ascii string descriptor representing the input line
1 !
1 !       LEX_DESC                - A longword containing the address of a standard
1 !                                 ascii string descriptor. The length and a_pointer
1 !                                 fields of this descriptor are filled in to reflect
1 !                                 the portion of the input which represents the token
1 !                                 recognized.
1 !
1 !       TOKEN                   - The address of a longword to contain the value
1 !                                 of the token recognized
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       Token value is returned and the lexical string descriptor is updated.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NOVALUE
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
```

```
1 !
1 !--
2       BEGIN
2
2       MAP
2           INPUT_DESC      : REF dbg$stg_desc,
2           LEX_DESC        : REF dbg$stg_desc;
2
2       LOCAL
2           CHAR            : BYTE,                 ! Input character
2           POINTER,                                ! Pointer to input char
2           TOKEN_START,                            ! Pointer to start of lexical string
2           TOKEN_END,                              ! Pointer to one char beyond lex string
2           LENGTH,
2           NEW_STRING      : REF VECTOR [,BYTE],   ! String vector
2           STRING          : REF VECTOR [,BYTE];   ! String vector
2
2
2       pointer = ch$ptr (.input_desc [dsc$a_pointer]);
2
2
2       ! Skip over leading white space
2       !
2       char = ch$rchar (.pointer);
2       WHILE .char EQL ' ' DO char = ch$a_rchar (pointer);
2
2
2       ! Pointer now points to the first character of the token string
2       !
2       token_start = .pointer;
2
2
2       ! Case off of the character to begin acceptance of the token
2       !
2       SELECTONE true
2           OF
2           SET
2
2           [.char EQL dbg$k_car_return] :  ! Null input line, <CR>
3               BEGIN
3               token_end = .token_start;
3               .token = dbg$k_tok_null;
2               END;
2
2           [.char EQL '\'] :
3               BEGIN
3               token_end = ch$plus (.token_start, 1);
3               .token = dbg$k_tok_bs;
2               END;
2
2           [.char EQL '.'] :
3               BEGIN
3               token_end = ch$plus (.token_start, 1);
3               .token = dbg$k_tok_dot;
2               END;
2
2           [.char EQL '%'] :       ! '%LINE' or '%LABEL'
```

```
3                       BEGIN
3                       LOCAL
3                           STRING_DESC : dbg$stg_desc;
3
3
3                       IF .dbg$gb_language EQL dbg$k_c
3                       THEN
4                           BEGIN
4
4                           ! Copy and upcase the string.
4                           !
4                           string = ch$plus (.pointer, 1);
4                           length = .input_desc [dsc$w_length] -
4                                   (.pointer + 1 - .input_desc [dsc$a_pointer]);
4                           new_string = dbg$get_tempmem((.length+3)/4);
4                           ch$move(.length, .string, .new_string);
4                           INCR i FROM 0 TO .length - 1 DO
4                               IF .new_string[.i] GEQ 'a' AND .new_string[.i] LEQ 'z'
4                               THEN
4                                   new_string[.i] = .new_string[.i] - ('a' - 'A');
4                           string_desc [dsc$w_length] = .length;
4                           string_desc [dsc$a_pointer] = .new_string;
4                           END
4
4                       ! All other languages.
4                       !
3                       ELSE
4                           BEGIN
4                           string_desc [dsc$a_pointer] = ch$plus (.pointer, 1);
4                           string_desc [dsc$w_length] = .input_desc [dsc$w_length] -
4                                                       (.pointer + 1 - .input_desc [dsc$a_pointer]);
3                           END;
3
3                       SELECTONE true
3                           OF
3                           SET
3
3                           [dbg$nmatch (string_desc, UPLIT BYTE (%ASCIC 'LINE'), 2)] :
3                               .token = dbg$k_tok_line;
3
3                           [dbg$nmatch (string_desc , UPLIT BYTE (%ASCIC 'LABEL'), 2)] :
3                               .token = dbg$k_tok_label;
3
3                           [dbg$nmatch (string_desc , UPLIT BYTE (%ASCIC 'NAME'), 1)] :
3                               .token = dbg$k_tok_qname;
3
3                           [OTHERWISE] :
3                               .token = dbg$k_tok_inval;
3
3                           TES;
3
3                       IF .dbg$gb_language EQL dbg$k_c
3                       THEN
3                           token_end = .pointer +
4                                   (.string_desc[dsc$a_pointer] - .new_string)
4
3                       ELSE
```

```
3                       token_end = .string_desc [dsc$a_pointer];
2                   END;
2
2               [.char GEQ '0' AND .char LEQ '9'] :      ! Integer
3                   BEGIN
3                   WHILE .char GEQ '0' AND .char LEQ '9' DO char = ch$a_rchar (pointer);
3
3
3                   token_end = .pointer;
3                   .token = dbg$k_tok_int;
2                   END;
2
2               [(.char GEQ 'A' AND .char LEQ 'Z') OR
2                (.char GEQ 'a' AND .char LEQ 'z')] :    ! ID
3                   BEGIN
3                   WHILE .char NEQ ','
3                           AND
3                         .char NEQ '\'
3                           AND
3                         .char NEQ ' '
3                           AND
3                         .char NEQ dbg$k_car_return
3                   DO
4                       BEGIN
4                       pointer = ch$plus (.pointer, 1);
4                       char = ch$rchar (.pointer);
3                       END;
3
3                   token_end = .pointer;
3                   .token = dbg$k_tok_id;
2                   END;
2
2               [OTHERWISE] :
3                   BEGIN
3                   .token = dbg$k_tok_inval;
2                   END;
2
2               TES;
2
2
2           ! Now fill in the lexical string descriptor
2           !
2           lex_string_desc [dsc$a_pointer] = .token_start;
2           lex_string_desc [dsc$w_length] = .token_end - .token_start;
2
2           RETURN;
2
1           END;        ! End of SCOPE_SCANNER


                                                    .PSECT  DBG$PLIT.NOWRT,  SHR,  PIC,0

                    45  4E  49  4C  04  0002E P.AAQ:  .ASCII  <4>\LINE\                               ;
                4C  45  42  41  4C  05  00033 P.AAR:  .ASCII  <5>\LABEL\                              ;
                    45  4D  41  4E  04  00039 P.AAS:  .ASCII  <4>\NAME\                               ;
```

```
                                                         .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                      0FFC 00000 SCOPE_SCANNER:
                                                         .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 2362
                  5E            0C  C2 00002             SUBL2   #12, SP                                        ;
                  50        04  AC  D0 00005             MOVL    INPUT_DESC, R0                                 ; 2437
                  56        04  A0  D0 00009             MOVL    4(R0), POINTER                                 ;
                  59            66  90 0000D 1$:         MOVB    (POINTER), CHAR                                ; 2442
                  20            59  91 00010             CMPB    CHAR, #32                                      ; 2443
                                04  12 00013             BNEQ    2$                                             ;
                                56  D6 00015             INCL    POINTER                                        ;
                                F4  11 00017             BRB     1$                                             ;
                  5B            56  D0 00019 2$:         MOVL    POINTER, TOKEN_START                           ; 2448
                  0D            59  91 0001C             CMPB    CHAR, #13                                      ; 2457
                                08  12 0001F             BNEQ    3$                                             ;
                  5A            5B  D0 00021             MOVL    TOKEN_START, TOKEN_END                         ; 2459
                            0C  BC  D4 00024             CLRL    @TOKEN                                         ; 2460
                                1D  11 00027             BRB     5$                                             ; 2453
            5C    8F            59  91 00029 3$:         CMPB    CHAR, #92                                      ; 2463
                                0A  12 0002D             BNEQ    4$                                             ;
                  5A        01  AB  9E 0002F             MOVAB   1(R11), TOKEN_END                              ; 2465
            0C    BC            04  D0 00033             MOVL    #4, @TOKEN                                     ; 2466
                                0D  11 00037             BRB     5$                                             ; 2453
                  2E            59  91 00039 4$:         CMPB    CHAR, #46                                      ; 2469
                                0B  12 0003C             BNEQ    6$                                             ;
                  5A        01  AB  9E 0003E             MOVAB   1(R11), TOKEN_END                              ; 2471
            0C    BC            07  D0 00042             MOVL    #7, @TOKEN                                     ; 2472
                              017B  31 00046 5$:         BRW     31$                                            ; 2453
                  25            59  91 00049 6$:         CMPB    CHAR, #37                                      ; 2475
                                03  13 0004C             BEQL    7$                                             ;
                              00D7  31 0004E             BRW     17$                                            ;
                  51        01  A6  9E 00051 7$:         MOVAB   1(R6), R1                                      ; 2487
                  07 00000000G  00  91 00055             CMPB    DBG$GB_LANGUAGE, #7                            ; 2481
                                48  12 0005C             BNEQ    10$                                            ;
                  53            51  D0 0005E             MOVL    R1, STRING                                     ; 2487
      52        04  A0          51  C3 00061             SUBL3   R1, 4(R0), R2                                  ; 2489
                  57            60  3C 00066             MOVZWL  (R0), LENGTH                                   ;
                  57            52  C0 00069             ADDL2   R2, LENGTH                                     ;
                  50        03  A7  9E 0006C             MOVAB   3(R7), R0                                      ; 2490
      7E          50            04  C7 00070             DIVL3   #4, R0, -(SP)                                  ;
        00000000G 00            01  FB 00074             CALLS   #1, DBG$GET_TEMPMEM                            ;
                  58            50  D0 0007B             MOVL    R0, NEW_STRING                                 ;
      68          63            57  28 0007E             MOVC3   LENGTH, (STRING), (NEW_STRING)                 ; 2491
                  50            01  CE 00082             MNEGL   #1, I                                          ; 2492
                                12  11 00085             BRB     9$                                             ;
            61    8F          6048  91 00087 8$:         CMPB    (I)[NEW_STRING], #97                           ; 2493
                                0B  1F 0008C             BLSSU   9$                                             ;
            7A    8F          6048  91 0008E             CMPB    (I)[NEW_STRING], #122                          ;
                                04  1A 00093             BGTRU   9$                                             ;
                6048            20  82 00095             SUBB2   #32, (I)[NEW_STRING]                           ; 2495
      EA          50            57  F2 00099 9$:         AOBLSS  LENGTH, I, 8$                                  ; 2493
                  6E            57  B0 0009D             MOVW    LENGTH, STRING_DESC                            ; 2496
            04    AE            58  D0 000A0             MOVL    NEW_STRING, STRING_DESC+4                      ; 2497
                                0D  11 000A4             BRB     11$                                            ; 2481
            04    AE            51  D0 000A6 10$:        MOVL    R1, STRING_DESC+4                              ; 2504
```

```
51          04   A0              51  C3 000AA          SUBL3   R1, 4(R0), R1                          ; 2506
6E               51              60  A1 000AF          ADDW3   (R0), R1, STRING_DESC                  ;
                                 02  DD 000B3  11$:    PUSHL   #2                                     ; 2513
                       00000000' EF  9F 000B5          PUSHAB  P.AAQ                                  ;
                              08 AE  9F 000BB          PUSHAB  STRING_DESC                            ;
     00000000G   00              03  FB 000BE          CALLS   #3, DBG$NMATCH                         ;
                 01              50  D1 000C5          CMPL    R0, #1                                 ;
                                 06  12 000C8          BNEQ    12$                                    ;
          0C     BC              02  D0 000CA          MOVL    #2, @TOKEN                             ; 2514
                                 3E  11 000CE          BRB     15$                                    ;
                                 02  DD 000D0  12$:    PUSHL   #2                                     ; 2516
                       00000000' EF  9F 000D2          PUSHAB  P.AAR                                  ;
                              08 AE  9F 000D8          PUSHAB  STRING_DESC                            ;
     00000000G   00              03  FB 000DB          CALLS   #3, DBG$NMATCH                         ;
                 01              50  D1 000E2          CMPL    R0, #1                                 ;
                                 06  12 000E5          BNEQ    13$                                    ;
          0C     BC              03  D0 000E7          MOVL    #3, @TOKEN                             ; 2517
                                 21  11 000EB          BRB     15$                                    ;
                                 01  DD 000ED  13$:    PUSHL   #1                                     ; 2519
                       00000000' EF  9F 000EF          PUSHAB  P.AAS                                  ;
                              08 AE  9F 000F5          PUSHAB  STRING_DESC                            ;
     00000000G   00              03  FB 000F8          CALLS   #3, DBG$NMATCH                         ;
                 01              50  D1 000FF          CMPL    R0, #1                                 ;
                                 06  12 00102          BNEQ    14$                                    ;
          0C     BC              09  D0 00104          MOVL    #9, @TOKEN                             ; 2520
                                 04  11 00108          BRB     15$                                    ;
          0C     BC              01  D0 0010A  14$:    MOVL    #1, @TOKEN                             ; 2523
                 07  00000000G   00  91 0010E  15$:    CMPB    DBG$GB_LANGUAGE, #7                    ; 2527
                                 08  12 00115          BNEQ    16$                                    ;
50          04   AE              58  C3 00117          SUBL3   NEW_STRING, STRING_DESC+4, R0          ; 2530
5A               50              56  C1 0011C          ADDL3   POINTER, R0, TOKEN_END                 ;
                                 3B  11 00120          BRB     22$                                    ; 2529
                 5A           04 AE  D0 00122  16$:    MOVL    STRING_DESC+4, TOKEN_END               ; 2533
                                 35  11 00126          BRB     22$                                    ; 2453
                                 51  D4 00128  17$:    CLRL    R1                                     ; 2536
                 30              59  91 0012A          CMPB    CHAR, #48                              ;
                                 02  1F 0012D          BLSSU   18$                                    ;
                                 51  D6 0012F          INCL    R1                                     ;
                                 50  D4 00131  18$:    CLRL    R0                                     ;
                 39              59  91 00133          CMPB    CHAR, #57                              ;
                                 02  1A 00136          BGTRU   19$                                    ;
                                 50  D6 00138          INCL    R0                                     ;
                 52              51  D2 0013A  19$:    MCOML   R1, R2                                 ;
                 50              52  CA 0013D          BICL2   R2, R0                                 ;
                 01              50  D1 00140          CMPL    R0, #1                                 ;
                                 1A  12 00143          BNEQ    23$                                    ;
                 30              59  91 00145  20$:    CMPB    CHAR, #48                              ; 2538
                                 0C  1F 00148          BLSSU   21$                                    ;
                 39              59  91 0014A          CMPB    CHAR, #57                              ;
                                 07  1A 0014D          BGTRU   21$                                    ;
                                 56  D6 0014F          INCL    POINTER                                ;
                 59              66  90 00151          MOVB    (POINTER), CHAR                        ;
                                 EF  11 00154          BRB     20$                                    ;
                 5A              56  D0 00156  21$:    MOVL    POINTER, TOKEN_END                     ; 2541
          0C     BC              06  D0 00159          MOVL    #6, @TOKEN                             ; 2542
                                 65  11 0015D  22$:    BRB     31$                                    ; 2453
                                 50  D4 0015F  23$:    CLRL    R0                                     ; 2545
```

```
                          41    8F            59  91 00161       CMPB     CHAR, #65                                ;
                                              02  1F 00165       BLSSU    24$                                      ;
                                              50  D6 00167       INCL     R0                                       ;
                                              52  D4 00169 24$:  CLRL     R2                                       ;
                          5A    8F            59  91 0016B       CMPB     CHAR, #90                                ;
                                              02  1A 0016F       BGTRU    25$                                      ;
                                              52  D6 00171       INCL     R2                                       ;
                                51            50  D2 00173 25$:  MCOML    R0, R1                                   ;
                                52            51  CA 00176       BICL2    R1, R2                                   ;
                                              51  D4 00179       CLRL     R1                                       ; 2546
                          61    8F            59  91 0017B       CMPB     CHAR, #97                                ;
                                              02  1F 0017F       BLSSU    26$                                      ;
                                              51  D6 00181       INCL     R1                                       ;
                                              50  D4 00183 26$:  CLRL     R0                                       ;
                          7A    8F            59  91 00185       CMPB     CHAR, #122                               ;
                                              02  1A 00189       BGTRU    27$                                      ;
                                              50  D6 0018B       INCL     R0                                       ;
                                53            51  D2 0018D 27$:  MCOML    R1, R3                                   ;
                                50            53  CA 00190       BICL2    R3, R0                                   ;
                                50            52  C8 00193       BISL2    R2, R0                                   ;
                                01            50  D1 00196       CMPL     R0, #1                                   ; 2545
                                              25  12 00199       BNEQ     30$                                      ;
                                2C            59  91 0019B 28$:  CMPB     CHAR, #44                                ; 2548
                                              17  13 0019E       BEQL     29$                                      ;
                          5C    8F            59  91 001A0       CMPB     CHAR, #92                                ; 2550
                                              11  13 001A4       BEQL     29$                                      ;
                                20            59  91 001A6       CMPB     CHAR, #32                                ; 2552
                                              0C  13 001A9       BEQL     29$                                      ;
                                0D            59  91 001AB       CMPB     CHAR, #13                                ; 2554
                                              07  13 001AE       BEQL     29$                                      ;
                                              56  D6 001B0       INCL     POINTER                                  ; 2557
                                59            66  90 001B2       MOVB     (POINTER), CHAR                          ; 2558
                                              E4  11 001B5       BRB      28$                                      ; 2548
                                5A            56  D0 001B7 29$:  MOVL     POINTER, TOKEN_END                       ; 2561
                          0C    BC            05  D0 001BA       MOVL     #5, @TOKEN                               ; 2562
                                              04  11 001BE       BRB      31$                                      ; 2453
                          0C    BC            01  D0 001C0 30$:  MOVL     #1, @TOKEN                               ; 2567
                   00000000'    EF            5B  D0 001C4 31$:  MOVL     TOKEN_START, LEX_STRING_DESC+4           ; 2575
      00000000'  EF             5A            5B  A3 001CB       SUBW3    TOKEN_START, TOKEN_END, LEX_STRING_DESC  ; 2576
                                                  04 001D3       RET                                               ; 2580

; Routine Size:  468 bytes,    Routine Base:  DBG$CODE + 1317


; 2471          2581  1
; 2472          2582  1
```

```
1 GLOBAL ROUTINE DBG$NPARSE_SCOPE_LIST (INPUT_DESC, SCOPE_LIST, MESSAGE_VECT) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine parses the objects of a SET SCOPE command. The pathname
1 !       parser is called within a loop to parse each scope item. A longword vector
1 !       is constructed which contains the number of scope items in the first cell
1 !       with the addresses of pathname descriptors in the subsequent cells.
1 !
1 !       A limit of 50 scope items per SET SCOPE command is observed.
1 !
1 !       This routine supplies the address of SCOPE_SCANNER as the lexical
1 !       analyzer for the pathname parser.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       INPUT_DESC              - A longword containing the address of a standard
1 !                                 character string descriptor reflecting the input
1 !
1 !       SCOPE_LIST              - The address of a longword to contain the address
1 !                                 of the pathname descriptor vector
1 !
1 !       MESSAGE_VECT            - The address of a longword to contain the address
1 !                                 of a message argument vector on error
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       On success, the pathname descriptor vector is obtained.
1 !
1 !       On failure, a message argument vector is constructed and returned.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS (1)       - Success. Pathname descriptor vector formed.
1 !
1 !       STS$K_SEVERE  (4)       - Failure. Error detected. Message argument vector
1 !                                 constructed.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If more than 50 scopes are collected, this routine will issue a string
1 !       truncation message.
1 !
1 !--
2     BEGIN
2
2     MAP
2       INPUT_DESC              : REF dbg$stg_desc;
```

```

LITERAL
    SCOPE_VECT_SIZE         = 51,
    MAX_NOM_SCOPES          = 50;

LOCAL
    SCOPE_VECT              : REF VECTOR,   ! Pathname descriptor vector
    INDEX,                                  ! Index into the vector
    DUMMY1,                                 ! Dummy parameter
    DUMMY2,                                 !       "       "
    STATUS;                                 ! Return status from
                                            ! the pathname parser.

! Allocate space for 50 pathname descriptor pointers, plus one for the count.
!
scope_vect = dbg$get_tempmem(scope_vect_size);


! Loop and collect the pathname descriptors
!
index = 1;
WHILE true
DO
    BEGIN

    ! For language C, we do some fancy footwork to
    ! make sure we preserve the original casing of
    ! the identifiers (since casing is significant
    ! in C).
    !
    IF .dbg$gb_language EQL dbg$k_c
    THEN
        BEGIN
        LOCAL
            length,
            new_pointer: REF VECTOR [,BYTE], ! Pointer to orig. command input
            pointer,                         ! Pointer into input string
            stg_desc: dbg$stg_desc,          ! String descriptor
            temp_ptr;

        ! Obtain a pointer into the current command buffer and check
        ! that it is still within the range of the start and end of
        ! the command buffer that we saved away in DBG$NGET_CMD.
        !
        pointer = .input_desc[dsc$a_pointer];
        IF (.pointer LSS .dbg$gl_upcase_command_ptr[0]) OR
           (.pointer GTR .dbg$gl_upcase_command_ptr[1])
        THEN
            $DBG_ERROR('DBGNPNP\DBG$NPARSE_SCOPE_LIST 10');

        ! Obtain a pointer into the original (not up-cased)
        ! command buffer (TEMP_PTR).
        ! Copy from this buffer into a new buffer pointed to
        ! by NEW_POINTER.
        ! We unfortunately have to allocate memory
        ! and copy strings in order to stuff a
        ! trailing carriage return at the end.
```

```
4           !
4           length = .input_desc[dsc$w_length];
4           new_pointer = dbg$get_tempmem((.length+3)/4);
4           temp_ptr = (.pointer - .dbg$gl_upcase_command_ptr[0]) +
4                       .dbg$gl_orig_command_ptr;
4           CH$MOVE (.length, .temp_ptr, .new_pointer);
4           new_pointer[.length-1] = dbg$k_car_return;
4
4           ! Fill in the string descriptor.
4           !
4           stg_desc[dsc$b_class] = dsc$k_class_s;
4           stg_desc[dsc$b_dtype] = dsc$k_dtype_t;
4           stg_desc[dsc$w_length] = .length;
4           stg_desc[dsc$a_pointer] = .new_pointer;
4           stg_desc[dsc$l_pos] = 0;
4
4           ! Pick up the pathname.
4           !
4           status = dbg$npathname_parser ( stg_desc,
4                                           scope_scanner,
4                                           scope_vect [.index],
4                                           dummy1,
4                                           dummy2,
4                                           true);
4
4           ! Update the input descriptor.
4           !
4           input_desc[dsc$w_length] = .input_desc[dsc$w_length] -
4               (.length - .stg_desc[dsc$w_length]);
4           input_desc[dsc$a_pointer] = .input_desc[dsc$a_pointer] +
4               (.length - .stg_desc[dsc$w_length]);
4           END
4
4       ! All other languages besides C ...
4       !
3       ELSE
3
3           status = dbg$npathname_parser (.input_desc,
3                                           scope_scanner,
3                                           scope_vect [.index],
3                                           dummy1,
3                                           dummy2,
3                                           true);
3
3       IF NOT .status
3       THEN
4           BEGIN
4           IF dbg$nmatch (.input_desc, UPLIT BYTE (1, dbg$k_car_return), 1)
4           THEN
5               BEGIN
5               .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
5               RETURN sts$k_severe;
5               END
5
4           ELSE
5               BEGIN
5               .message_vect = dbg$nsyntax_error (dbg$nnext_word (.input_desc));
```

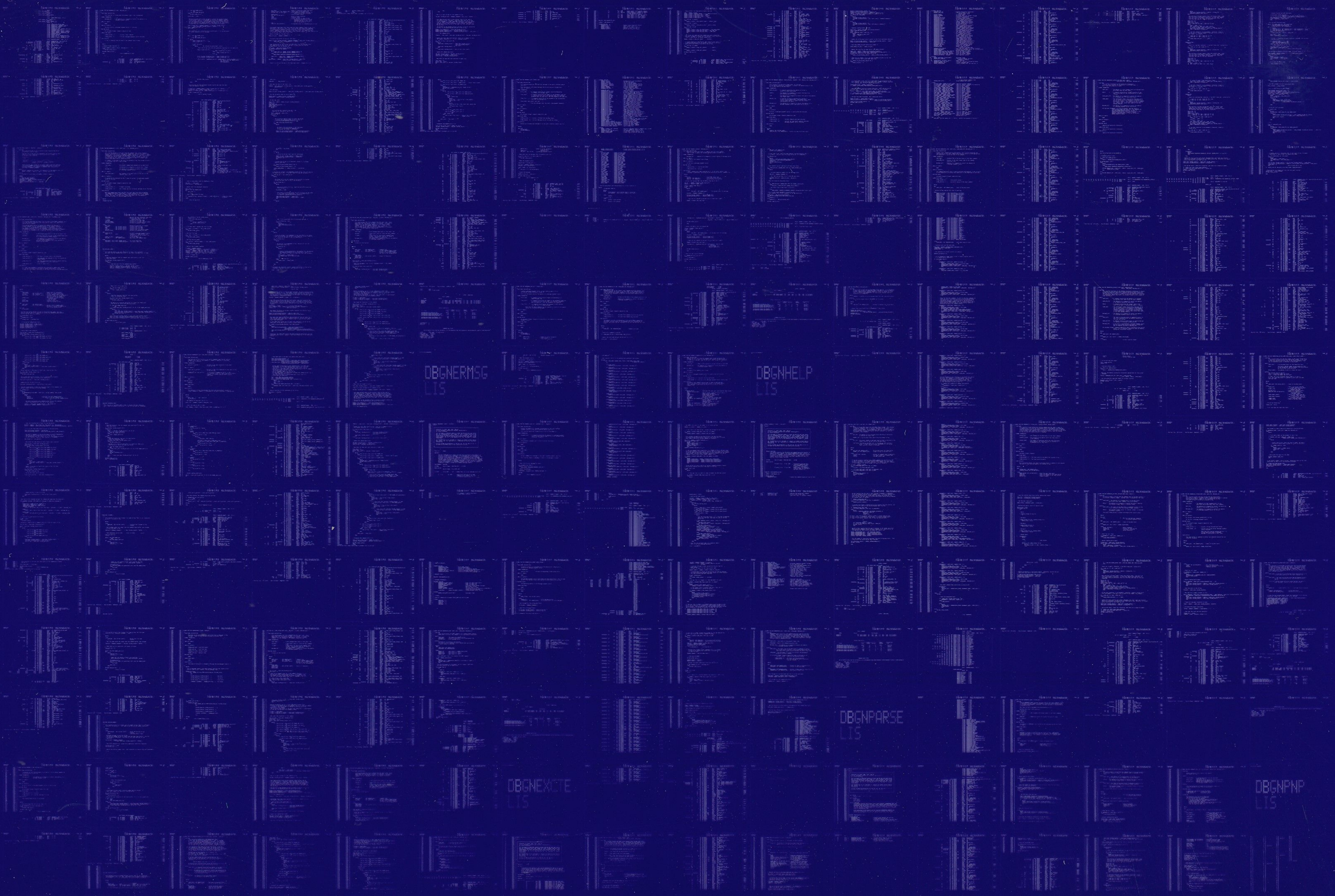
```
5                    RETURN sts$k_severe;
4                    END;
4
3                END;
3
3
3            ! Look for a comma that separates scopes
3            !
3            IF NOT dbg$nmatch (.input_desc, UPLIT BYTE (%ASCIC ','), 1)
3            THEN
3                EXITLOOP;
3
3            ! Check for end of line
3            !
3            IF dbg$nmatch (.input_desc, UPLIT BYTE (1, dbg$k_car_return), 1)
3            THEN
4                BEGIN
4                .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
4                RETURN sts$k_severe;
3                END;
3
3            ! There is atleast one more scope. Check for exceeding the limit.
3            !
3            IF .index GEQ max_num_scopes
3            THEN
4                BEGIN
4
4                ! Issue a truncation message.
4                !
4                dbg$nout_info (dbg$_stgtrunc);
4
4
4                ! Set up a phony input descriptor and exit the loop
4                !
4                input_desc [dsc$a_pointer] = UPLIT BYTE (dbg$k_car_return);
4                input_desc [dsc$w_length] = 1;
4                EXITLOOP;
3                END;
3
3
3            ! Update the index
3            !
3            index = .index + 1;
3
2            END;            ! End of loop
2
2
2        ! The scopes have been collected. Set the count.
2        !
2        scope_vect [0] = .index;
2
2
2        ! Return the scope list and success
2        !
2        .scope_list = .scope_vect;
2
2        RETURN sts$k_success;
```

```
: 2702          2811  2
: 2703          2812  1    END;



                                                                    .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

50 4E 24 47 42 44 5C 50 4E 50 4E 47 42 44 20 0003E P.AAT:  .ASCII  \ DBGNPNP\<92>\DBG$NPARSE_SCOPE_LIST 10\  ;
54 53 49 4C 5F 45 50 4F 43 53 5F 45 53 52 41 0004D                                                          ;
                                         30 31 20 0005C                                                          ;
                                            0D 01 0005F P.AAU:  .BYTE   1, 13                                    ;
                                            2C 01 00061 P.AAV:  .ASCII  <1>\,\                                   ;
                                            0D 01 00063 P.AAW:  .BYTE   1, 13                                    ;
                                               0D 00065 P.AAX:  .BYTE   13                                       ;


                                                                    .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                           0FFC 00000          .ENTRY  DBG$NPARSE_SCOPE_LIST, Save R2,R3,R4,R5,R6,-; 2583
                                                                       R7,R8,R9,R10,R11
                         5E            18  C2 00002            SUBL2   #24, SP                                    ;
                                       33  DD 00005            PUSHL   #51                                        ; 2655
              00000000G  00            01  FB 00007            CALLS   #1, DBG$GET_TEMPMEM                        ;
                         5B            50  D0 0000E            MOVL    R0, SCOPE_VECT                             ;
                         59            01  D0 00011            MOVL    #1, INDEX                                  ; 2660
                         57       04   AC  D0 00014            MOVL    INPUT_DESC, R7                             ; 2684
                         5A          6B49  DE 00018 1$:        MOVAL   (SCOPE_VECT)[INDEX], R10                   ; 2717
                         07 00000000G  00  91 0001C            CMPB    DBG$GB_LANGUAGE, #7                        ; 2670
                                       03  13 00023            BEQL    2$                                         ;
                                     0092  31 00025            BRW     5$                                         ;
                         52       04   A7  D0 00028 2$:        MOVL    4(R7), POINTER                             ; 2684
              00000000G  00            52  D1 0002C            CMPL    POINTER, DBG$GL_UPCASE_COMMAND_PTR         ; 2685
                                       09  19 00033            BLSS    3$                                         ;
              00000000G  00            52  D1 00035            CMPL    POINTER, DBG$GL_UPCASE_COMMAND_PTR+4       ; 2686
                                       15  15 0003C            BLEQ    4$                                         ;
                            00000000'  EF  9F 0003E 3$:        PUSHAB  P.AAT                                      ; 2688
                                       01  DD 00044            PUSHL   #1                                         ;
                            00028362   8F  DD 00046            PUSHL   #164706                                    ;
              00000000G  00            03  FB 0004C            CALLS   #3, LIB$SIGNAL                             ;
                         56            67  3C 00053 4$:        MOVZWL  (R7), LENGTH                               ; 2698
                         50       03   A6  9E 00056            MOVAB   3(R6), R0                                  ; 2699
                 7E      50            04  C7 0005A            DIVL3   #4, R0, -(SP)                              ;
              00000000G  00            01  FB 0005E            CALLS   #1, DBG$GET_TEMPMEM                        ;
                         58            50  D0 00065            MOVL    R0, NEW_POINTER                            ;
                         52 00000000G  00  C2 00068            SUBL2   DBG$GL_UPCASE_COMMAND_PTR, R2              ; 2700
                 50      52 00000000G  00  C1 0006F            ADDL3   DBG$GL_ORIG_COMMAND_PTR, R2, TEMP_PTR      ; 2701
                 68      60            56  28 00077            MOVC3   LENGTH, (TEMP_PTR), (NEW_POINTER)          ; 2702
                   FF A648             0D  90 0007B            MOVB    #13, -1(LENGTH)[NEW_POINTER]               ; 2703
                   0E    AE      010E  8F  B0 00080            MOVW    #270, STG_DESC+2                           ; 2708
                   0C    AE            56  B0 00086            MOVW    LENGTH, STG_DESC                           ; 2709
                   10    AE            58  D0 0008A            MOVL    NEW_POINTER, STG_DESC+4                    ; 2710
                                  14   AE  D4 0008E            CLRL    STG_DESC+8                                 ; 2711
                                       01  DD 00091            PUSHL   #1                                         ; 2717
                                  08   AE  9F 00093            PUSHAB  DUMMY2                                     ;
                                  10   AE  9F 00096            PUSHAB  DUMMY1                                     ;
                                       5A  DD 00099            PUSHL   R10                                        ;
```

```
                     FD8D   CF  9F 0009B         PUSHAB  SCOPE_SCANNER                                        ; 2715
                       20   AE  9F 0009F         PUSHAB  STG_DESC                                             ;
          EA6E   CF         06  FB 000A2         CALLS   #6, DBG$NPATHNAME_PARSER                             ; 2717
                 6E         50  D0 000A7         MOVL    R0, STATUS                                           ;
                 50    0C   AE  3C 000AA         MOVZWL  STG_DESC, R0                                         ; 2725
                 50         56  C2 000AE         SUBL2   LENGTH, R0                                           ;
                 67         50  A0 000B1         ADDW2   R0, (R7)                                             ;
            04   A7         50  C2 000B4         SUBL2   R0, 4(R7)                                            ; 2727
                            18  11 000B8         BRB     6$                                                   ; 2670
                            01  DD 000BA 5$:     PUSHL   #1                                                   ; 2736
                       08   AE  9F 000BC         PUSHAB  DUMMY2                                               ;
                       10   AE  9F 000BF         PUSHAB  DUMMY1                                               ;
                            5A  DD 000C2         PUSHL   R10                                                  ;
                     FD64   CF  9F 000C4         PUSHAB  SCOPE_SCANNER                                        ; 2734
                            57  DD 000C8         PUSHL   R7                                                   ; 2736
          EA46   CF         06  FB 000CA         CALLS   #6, DBG$NPATHNAME_PARSER                             ;
                 6E         50  D0 000CF         MOVL    R0, STATUS                                           ;
                 28         6E  E8 000D2 6$:     BLBS    STATUS, 7$                                           ; 2741
                            01  DD 000D5         PUSHL   #1                                                   ; 2744
                00000000'   EF  9F 000D7         PUSHAB  P.AAU                                                ;
                            57  DD 000DD         PUSHL   R7                                                   ;
     00000000G   00         03  FB 000DF         CALLS   #3, DBG$NMATCH                                       ;
                 3C         50  E8 000E6         BLBS    R0, 8$                                               ;
                            57  DD 000E9         PUSHL   R7                                                   ; 2753
     00000000G   00         01  FB 000EB         CALLS   #1, DBG$NNEXT_WORD                                   ;
                            50  DD 000F2         PUSHL   R0                                                   ;
     00000000G   00         01  FB 000F4         CALLS   #1, DBG$NSYNTAX_ERROR                                ;
                            35  11 000FB         BRB     9$                                                   ;
                            01  DD 000FD 7$:     PUSHL   #1                                                   ; 2762
                00000000'   EF  9F 000FF         PUSHAB  P.AAV                                                ;
                            57  DD 00105         PUSHL   R7                                                   ;
     00000000G   00         03  FB 00107         CALLS   #3, DBG$NMATCH                                       ;
                 4D         50  E9 0010E         BLBC    R0, 12$                                              ;
                            01  DD 00111         PUSHL   #1                                                   ; 2768
                00000000'   EF  9F 00113         PUSHAB  P.AAW                                                ;
                            57  DD 00119         PUSHL   R7                                                   ;
     00000000G   00         03  FB 0011B         CALLS   #3, DBG$NMATCH                                       ;
                 15         50  E9 00122         BLBC    R0, 10$                                              ;
                000280D0    8F  DD 00125 8$:     PUSHL   #164048                                              ; 2771
     00000000G   00         01  FB 0012B         CALLS   #1, DBG$NMAKE_ARG_VECT                               ;
            0C   BC         50  D0 00132 9$:     MOVL    R0, @MESSAGE_VECT                                    ;
                 50         04  D0 00136         MOVL    #4, R0                                               ; 2772
                                04 00139         RET                                                          ;
                 32         59  D1 0013A 10$:    CMPL    INDEX, #50                                           ; 2777
                            1A  19 0013D         BLSS    11$                                                  ;
                0002804B    8F  DD 0013F         PUSHL   #163915                                              ; 2783
     00000000G   00         01  FB 00145         CALLS   #1, DBG$NOUT_INFO                                    ;
            04   A7 00000000'   EF  9E 0014C     MOVAB   P.AAX, 4(R7)                                         ; 2788
                 67         01  B0 00154         MOVW    #1, (R7)                                             ; 2789
                            05  11 00157         BRB     12$                                                  ; 2779
                            59  D6 00159 11$:    INCL    INDEX                                                ; 2796
                      FEBA  31 0015B             BRW     1$                                                   ; 2661
                 6B         59  D0 0015E 12$:    MOVL    INDEX, (SCOPE_VECT)                                  ; 2803
            08   BC         5B  D0 00161         MOVL    SCOPE_VECT, @SCOPE_LIST                              ; 2808
                 50         01  D0 00165         MOVL    #1, R0                                               ; 2810
                                04 00168         RET                                                          ; 2812
```

```
; Routine Size:  361 bytes,    Routine Base:  DBG$CODE + 14EB
; 2705          2813  1 END                                   !End of module
; 2706          2814  0 ELUDOM



                                                                                .EXTRN  LIB$SIGNAL

;                                       PSECT SUMMARY
;
;       Name                      Bytes                         Attributes
;
;    DBG$OWN                           68  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;    DBG$PLIT                         102  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;    DBG$CODE                        5716  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)




;                                Library Statistics
;
;                                                  -------- Symbols --------       Pages        Processing
;       File                                       Total    Loaded   Percent       Mapped       Time
;
;    _$255$DUA28:[SYSLIB]LIB.L32;1                 18619         9         0        1000          00:01.9
;    _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1            32         0         0           7          00:00.1
;    _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1            1545        32         2          97          00:02.1
;    _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                                    418         1         0          31          00:00.3
;    _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1             386         4         1          22          00:00.3





;                                       COMMAND QUALIFIERS

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

; Size:          5716 code + 170 data bytes
; Run Time:         01:38.3
; Elapsed Time:     04:09.9
; Lines/CPU Min:     1718
; Lexemes/CPU-Min: 21243
; Memory Used:  380 pages
; Compilation Complete
```

0087 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY



DBGNERMSG
LIS

DBGNHELP
LIS

DBGNPARSE
LIS

DBGNEXCTE
LIS

DBGNPNP
LIS

0088 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY