```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
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 EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      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      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
```

FILEID**DBGNSEARC

```
DDDDDDDD    BBBBBBBB      GGGCGGGG  NN      NN    SSSSSSSS  EEEEEEEEEE    AAAAAA    RRRRRRRR      CCCCCCCC
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN    SSSSSSSS  EEEEEEEEEE    AAAAAA    RRRRRRRR      CCCCCCCC
DD      DD  BB      BB  GG          NN      NN  SS          EE          AA      AA  RR      RR  CC
DD      DD  BB      BB  GG          NN      NN  SS          EE          AA      AA  RR      RR  CC
DD      DD  BB      BB  GG          NNNN    NN  SS          EE          AA      AA  RR      RR  CC
DD      DD  BB      BB  GG          NNNN    NN  SS          EE          AA      AA  RR      RR  CC
DD      DD  BBBBBBBB    GG          NN  NN  NN    SSSSSS    EEEEEEEE    AA      AA  RRRRRRRR    CC
DD      DD  BBBBBBBB    GG          NN  NN  NN    SSSSSS    EEEEEEEE    AA      AA  RRRRRRRR    CC
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN          SS  EE          AAAAAAAAAA  RR  RR      CC
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN          SS  EE          AAAAAAAAAA  RR  RR      CC
DD      DD  BB      BB  GG      GG  NN      NN          SS  EE          AA      AA  RR    RR    CC                ....
DD      DD  BB      BB  GG      GG  NN      NN          SS  EE          AA      AA  RR    RR    CC                ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  SSSSSSSS    EEEEEEEEEE  AA      AA  RR      RR    CCCCCCCC      ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  SSSSSSSS    EEEEEEEEEE  AA      AA  RR      RR    CCCCCCCC      ....

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

```
;     1          0001  0 MODULE DBGNSEARC (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                  *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
;    28          0028  1 !
;    29          0029  1 !
;    30          0030  1 ! FACILITY:
;    31          0031  1 !
;    32          0032  1 !       DEBUG
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !
;    36          0036  1 !       This module contains the parse and execution networks for the SEARCH
;    37          0037  1 !       verb. The parsing method employed is that of ATN's.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !
;    41          0041  1 !       VAX/VMS
;    42          0042  1 !
;    43          0043  1 ! AUTHOR:
;    44          0044  1 !
;    45          0045  1 !       Richard Title
;    46          0046  1 !
;    47          0047  1 ! CREATION DATE:
;    48          0048  1 !
;    49          0049  1 !       10-22-81
;    50          0050  1 !
;    51          0051  1 ! VERSION:
;    52          0052  1 !
;    53          0053  1 !       V03.0-001
;    54          0054  1 !
;    55          0055  1 ! MODIFIED BY:
;    56          0056  1 !       V. Holt, 27-May-1982
;    57          0057  1 !
```

```
;    58          0058  1 ! REVISION HISTORY:
;    59          0059  1 !   27-May-82   VJH     Removed all references to DBG$FAO_PUT and DBG$OUT_PUT,
;    60          0060  1 !                       which are now obsolete.
;    61          0061  1 !
;    62          0062  1
;    63          0063  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    64          0197  1
;    65          0198  1 LIBRARY 'LIB$:DBGGEN.L32';
;    66          0199  1
;    67          0200  1 FORWARD ROUTINE
;    68          0201  1         DBG$NPARSE_SEARCH,              ! Parse network
;    69          0202  1         DBG$NEXECUTE_SEARCH,            ! Execution network
;    70          0203  1         DBG$NACCEPT_STRING,             ! Subroutine of parsing routine
;    71          0204  1                                         !       that reads the search
;    72          0205  1                                         !       string from the input
;    73          0206  1                                         !       stream.
;    74          0207  1         DBG$PARSE_SEARCH;               ! Provides an interface to
;    75          0208  1                                         !       DBG$NPARSE_SEARCH from
;    76          0209  1                                         !       the old debugger.
```

```
;    78          0210  1 EXTERNAL ROUTINE
;    79          0211  1     DBG$GET_MEMORY,                 ! Get a memory block
;    80          0212  1     DBG$GET_TEMPMEM,                ! Get a temporary memory block
;    81          0213  1     DBG$NEWLINE: NOVALUE,           ! Flush the current print line
;    82          0214  1     dbg$nmake_arg_vect,             ! Constructs error messages
;    83          0215  1     dbg$nmatch,                     ! Tries to match the next token
;    84          0216  1     dbg$nnext_word,                 ! Gets next word from input
;    85          0217  1     dbg$nout_arg_vect: NOVALUE,     ! Outputs an error message
;    86          0218  1     dbg$nsave_decimal_integer,      ! Reads an integer from the
;    87          0219  1                                     !       input string
;    88          0220  1     dbg$nsave_string,               ! Reads a character string from the
;    89          0221  1                                     !       input string
;    90          0222  1     dbg$nsyntax_error,              ! Reports a syntax error
;    91          0223  1     DBG$PRINT: NOVALUE,             ! Print some ASCII text
;    92          0224  1     dbg$set_search_lvl: NOVALUE,    ! Sets level of search data structure
;    93          0225  1                                     !       overrid (see DBGMOD)
;    94          0226  1     dbg$src_search_cmd: NOVALUE,    ! The routine in DBGSOURCE that
;    95          0227  1                                     !       performs the search and outputs
;    96          0228  1                                     !       the result to the terminal
;    97          0229  1     dbg$sta_getsourcemod,           ! Gets module rst pointer
;    98          0230  1     dbg$sta_symname;                ! Turns module rst pointer back
;    99          0231  1                                     !       into a string
;   100          0232  1
;   101          0233  1 EXTERNAL
;   102          0234  1     dbg$gb_search_ptr:REF VECTOR[,BYTE],! Pointer to SEARCH data structure
;   103          0235  1     dbg$src_next_lnum,              ! Contains the default starting line number
;   104          0236  1     dbg$src_next_modrstptr,         ! Contains the default module rst pointer
;   105          0237  1     dbg$src_search_string : VECTOR [, BYTE];
;   106          0238  1                                     ! The global in DBGSOURCE that is used
;   107          0239  1                                     !      to pass the search string.
;   108          0240  1
;   109          0241  1 LITERAL
;   110          0242  1     adverb_literal_all = 0,
;   111          0243  1     adverb_literal_ident = 1;
;   112          0244  1
```

```
;  114          0245  1 GLOBAL ROUTINE dbg$nparse_search (
;  115          0246  1     input_desc,
;  116          0247  1     verb_node,
;  117          0248  1     message_vect) =
;  118          0249  1 !++
;  119          0250  1 ! Functional Description
;  120          0251  1 !
;  121          0252  1 !       ATN parse network for the SEARCH verb.
;  122          0253  1 !       This routine takes a verb node for the SEARCH verb, and a string
;  123          0254  1 !       descriptor for the rema ning (unparsed) input.
;  124          0255  1 !       A command execution tree is built. The form of the tree is:
;  125          0256  1 !
;  126          0257  1 !        -----------       -----------       -----------
;  127          0258  1 !       | verb node |-->--| noun node |-->--| noun node |
;  128          0259  1 !        -----------       -----------       -----------
;  129          0260  1 !            |
;  130          0261  1 !            V
;  131          0262  1 !        -------------     -------------
;  132          0263  1 !       | adverb node |-->--| adverb node |
;  133          0264  1 !        -------------     -------------
;  134          0265  1 !
;  135          0266  1 !       The adverb nodes contain the command switches (STRING, IDENTIFIER,
;  136          0267  1 !       NEXT, ALL) if any are present.
;  137          0268  1 !       The first noun node contains the starting line number,
;  138          0269  1 !       ending line number, and module rst pointer. The second noun node
;  139          0270  1 !       contains a pointer to the search string.
;  140          0271  1 !
;  141          0272  1 ! Formal Parameters
;  142          0273  1 !
;  143          0274  1 !       input_desc      - A longword containing the address of the
;  144          0275  1 !                         command input descriptor.
;  145          0276  1 !       verb_node       - A longword containing the address of the verb node.
;  146          0277  1 !       message_vect    - The address of a longword to contain the address
;  147          0278  1 !                         of a standard message argument vector.
;  148          0279  1 !
;  149          0280  1 ! Implicit Inputs
;  150          0281  1 !
;  151          0282  1 !       none
;  152          0283  1 !
;  153          0284  1 ! Implicit Outputs
;  154          0285  1 !
;  155          0286  1 !       On success, the command execution tree is constructed.
;  156          0287  1 !       On failure, a message argument vector is constructed or obtained.
;  157          0288  1 !
;  158          0289  1 ! Routine value
;  159          0290  1 !
;  160          0291  1 !       sts$k_success (1)       - Success. Command execution tree constructed.
;  161          0292  1 !       sts$k_severe  (4)       - Failure. Error encountered. Message argument
;  162          0293  1 !                                 constructed and returned.
;  163          0294  1 !
;  164          0295  1 ! Side Effects
;  165          0296  1 !
;  166          0297  1 !       none
;  167          0298  1 !--
;  168          0299  2 BEGIN
;  169          0300  2
;  170          0301  2 MAP
```

```
;  171          0302  2     input_desc  : REF dbg$stg_desc,
;  172          0303  2     verb_node   : REF dbg$verb_node;
;  173          0304  2
;  174          0305  2 BIND
;  175          0306  2     dbg$cs_all          = UPLIT BYTE (3, 'ALL'),
;  176          0307  2     dbg$cs_ident        = UPLIT BYTE (10,'IDENTIFIER'),
;  177          0308  2     dbg$cs_next         = UPLIT BYTE (4, 'NEXT'),
;  178          0309  2     dbg$cs_string       = UPLIT BYTE (6, 'STRING'),
;  179          0310  2     dbg$cs_backslash    = UPLIT BYTE (1, dbg$k_backslash),
;  180          0311  2     dbg$cs_colon        = UPLIT BYTE (1, dbg$k_colon),
;  181          0312  2     dbg$cs_cr           = UPLIT BYTE (1, dbg$k_car_return),
;  182          0313  2     dbg$cs_quote        = UPLIT BYTE (1, dbg$k_quote),
;  183          0314  2     dbg$cs_dblquote     = UPLIT BYTE (1, dbg$k_dblquote),
;  184          0315  2     dbg$cs_slash        = UPLIT BYTE (1, dbg$k_slash);
;  185          0316  2
;  186          0317  2 ! NAME_BUF must be an OWN variable since its address may get placed in
;  187          0318  2 ! an error message vector and used later during output of an error
;  188          0319  2 ! message.
;  189          0320  2 !
;  190          0321  2 OWN
;  191          0322  2     name_buf : VECTOR [81, BYTE];       ! Holds counted string with module name
;  192          0323  2
;  193          0324  2 LOCAL
;  194          0325  2     adverb_node : REF dbg$adverb_node,  ! points to an adverb node
;  195          0326  2     all_flag,                           ! TRUE if /ALL or /NEXT is present
;  196          0327  2     all_value,                          ! TRUE for /ALL, FALSE for /NEXT
;  197          0328  2     char,                               ! Used during parsing of module name
;  198          0329  2     dblquote_flag,                      ! Indicates a double quote
;  199          0330  2     delimiter,                          ! either " or '
;  200          0331  2     eol_flag,                           ! Indicates end of command line
;  201          0332  2     high_lnum,                          ! High line number in the search range
;  202          0333  2     I,                                  ! Temporary string index for scan-ahead
;  203          0334  2     ident_flag,                         ! TRUE if /IDENT or /STRING is present
;  204          0335  2     ident_value,                        ! TRUE for /IDENT, FALSE for /STRING
;  205          0336  2     is_it_name,                         ! flag indicating we read a module name
;  206          0337  2     length,                             ! Used during parsing of module name
;  207          0338  2     link,                               ! Used for building linked lists.
;  208          0339  2     low_lnum,                           ! Low line number in the search range
;  209          0340  2     modrstptr,                          ! RST pointer for the module being searched
;  210          0341  2     noun_node : REF dbg$noun_node,      ! A node in the command execution tree
;  211          0342  2     quote_flag,                         ! Indicates a '
;  212          0343  2     string_ptr,                         ! Used during parsing of module name
;  213          0344  2     switch_flag,                        ! Indicates a switch is present
;  214          0345  2     TPTR: REF VECTOR[,BYTE];            ! Temporary string pointer for scanning
;  215          0346  2
;  216          0347  2
;  217          0348  2     ! Initialize the switch variables.
;  218          0349  2     !
;  219          0350  2     switch_flag = FALSE;
;  220          0351  2     all_flag = FALSE;
;  221          0352  2     ident_flag = FALSE;
;  222          0353  2
;  223          0354  2
;  224          0355  2     ! Accept any override switches that may be present
;  225          0356  2     !
;  226          0357  2     WHILE dbg$nmatch (.input_desc, dbg$cs_slash, 1) DO
;  227          0358  3         BEGIN
```

```
;  228     0359  3          SELECTONE TRUE OF
;  229     0360  3              SET
;  230     0361  3
;  231     0362  3              [dbg$nmatch (.input_desc, dbg$cs_all, 1)] : ! /ALL
;  232     0363  4                  BEGIN
;  233     0364  4                  switch_flag = TRUE;
;  234     0365  4                  all_flag = TRUE;
;  235     0366  4                  all_value = TRUE;
;  236     0367  3                  END;
;  237     0368  3
;  238     0369  3              [dbg$nmatch (.input_desc, dbg$cs_ident, 1)] : ! /IDENT
;  239     0370  4                  BEGIN
;  240     0371  4                  switch_flag = TRUE;
;  241     0372  4                  ident_flag = TRUE;
;  242     0373  4                  ident_value = TRUE;
;  243     0374  3                  END;
;  244     0375  3
;  245     0376  3              [dbg$nmatch (.input_desc, dbg$cs_next, 1)] : ! ! /NEXT
;  246     0377  4                  BEGIN
;  247     0378  4                  switch_flag = TRUE;
;  248     0379  4                  all_flag = TRUE;
;  249     0380  4                  all_value = FALSE;
;  250     0381  3                  END;
;  251     0382  3
;  252     0383  3              [dbg$nmatch (.input_desc, dbg$cs_string, 1)] : ! /STRING
;  253     0384  4                  BEGIN
;  254     0385  4                  switch_flag = TRUE;
;  255     0386  4                  ident_flag = TRUE;
;  256     0387  4                  ident_value = FALSE;
;  257     0388  3                  END;
;  258     0389  3
;  259     0390  3              [ OTHERWISE ] :     ! Syntax error
;  260     0391  4                  BEGIN
;  261     0392  4                  .message_vect =
;  262     0393  4                      dbg$nsyntax_error (dbg$nnext_word (.input_desc));
;  263     0394  4                  RETURN sts$k_severe;
;  264     0395  3                  END;
;  265     0396  3
;  266     0397  3              TES;
;  267     0398  2          END;
;  268     0399  2
;  269     0400  2      ! Construct any adverb nodes, if needed.
;  270     0401  2      !
;  271     0402  2      IF .switch_flag
;  272     0403  2      THEN
;  273     0404  3          BEGIN
;  274     0405  3
;  275     0406  3          link = verb_node [dbg$l_verb_adverb_ptr];
;  276     0407  3
;  277     0408  3          SELECT TRUE OF
;  278     0409  3              SET
;  279     0410  3
;  280     0411  3              [.all_flag] : ! We have either /ALL or /NEXT
;  281     0412  4                  BEGIN
;  282     0413  4                  ! Construct an adverb node and link.
;  283     0414  4                  !
;  284     0415  4                  ADVERB_NODE = DBG$GET_TEMPMEM(DBG$K_ADVERB_NODE_SIZE);
```

```
;  285          0416  4                .link = .adverb_node;
;  286          0417  4                link = adverb_node [dbg$l_adverb_link];
;  287          0418  4                adverb_node [dbg$b_adverb_litera[] = adverb_literal_all;
;  288          0419  4                adverb_node [dbg$l_adverb_value] = .all_value;
;  289          0420  3                END;
;  290          0421  3
;  291          0422  3            [.ident_flag] : ! We have either /IDENT or /STRING
;  292          0423  4                BEGIN
;  293          0424  4                ! Construct an adverb node and link.
;  294          0425  4                !
;  295          0426  4                ADVERB_NODE = DBG$GET_TEMPMEM(DBG$K_ADVERB_NODE_SIZE);
;  296          0427  4                .link = .adverb_node;
;  297          0428  4                link = adverb_node [dbg$l_adverb_link];
;  298          0429  4                adverb_node [dbg$b_adverb_litera[] = adverb_literal_ident;
;  299          0430  4                adverb_node [dbg$l_adverb_value] = .ident_value;
;  300          0431  3                END;
;  301          0432  3
;  302          0433  3            TES;
;  303          0434  3
;  304          0435  3        ! Now put a zero in the last link field
;  305          0436  3        !
;  306          0437  3        .link = 0;
;  307          0438  3
;  308          0439  2        END;
;  309          0440  2
;  310          0441  2
;  311          0442  2    ! Create and link a noun node
;  312          0443  2    !
;  313          0444  2    NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE);
;  314          0445  2    verb_node[dbg$l_verb_object_ptr] = .noun_node;
;  315          0446  2
;  316          0447  2
;  317          0448  2    ! Check for SEARCH <cr>
;  318          0449  2    !
;  319          0450  2    IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  320          0451  2    THEN
;  321          0452  3        BEGIN
;  322          0453  3        eol_flag = TRUE;
;  323          0454  3        dblquote_flag = FALSE;
;  324          0455  3        quote_flag = FALSE;
;  325          0456  3        END
;  326          0457  2    ELSE
;  327          0458  3        BEGIN
;  328          0459  3        eol_flag = FALSE;
;  329          0460  3
;  330          0461  3        ! Check for SEARCH "string"
;  331          0462  3        !
;  332          0463  3        IF dbg$nmatch (.input_desc, dbg$cs_dblquote, 1)
;  333          0464  3        THEN
;  334          0465  4            BEGIN
;  335          0466  4            dblquote_flag = TRUE;
;  336          0467  4            quote_flag = FALSE;
;  337          0468  4            delimiter = dbg$k_dblquote;
;  338          0469  4            END
;  339          0470  3        ELSE
;  340          0471  4            BEGIN
;  341          0472  4            dblquote_flag = FALSE;
```

```
;  342          0473  4            IF dbg$nmatch (.input_desc, dbg$cs_quote, 1)
;  343          0474  4            THEN
;  344          0475  5                BEGIN
;  345          0476  5                quote_flag = TRUE;
;  346          0477  5                delimiter = dbg$k_quote;
;  347          0478  5                END
;  348          0479  4            ELSE
;  349          0480  5                BEGIN
;  350          0481  5                quote_flag = FALSE;
;  351          0482  5                delimiter = dbg$k_car_return;
;  352          0483  4                END;
;  353          0484  3            END;
;  354          0485  2        END;
;  355          0486  2
;  356          0487  2    IF .dblquote_flag OR .quote_flag OR .eol_flag
;  357          0488  2    THEN
;  358          0489  3        BEGIN ! no location argument supplied
;  359          0490  3
;  360          0491  3        ! Try to get default module.
;  361          0492  3        !
;  362          0493  3        IF .dbg$src_next_modrstptr EQL 0
;  363          0494  3        THEN
;  364          0495  4            BEGIN
;  365          0496  4            ! report an error
;  366          0497  4            .message_vect = dbg$nmake_arg_vect(dbg$_nonxtlin);
;  367          0498  4            RETURN sts$k_severe;
;  368          0499  3            END;
;  369          0500  3
;  370          0501  3        ! The module rst pointer is placed in the adjective field of
;  371          0502  3        ! the noun node.
;  372          0503  3        !
;  373          0504  3        noun_node [dbg$l_adjective_ptr] = .dbg$src_next_modrstptr;
;  374          0505  3
;  375          0506  3        ! Fill in the starting line number based on global info
;  376          0507  3        !
;  377          0508  3        noun_node [dbg$l_noun_value] = .dbg$src_next_lnum; ! Start of search
;  378          0509  3
;  379          0510  3        ! The search should go to the end of the module. Fill in a line
;  380          0511  3        ! number that is definitely beyond the end of the module.
;  381          0512  3        !
;  382          0513  3        noun_node [dbg$l_noun_value2] = 2000000000;
;  383          0514  3
;  384          0515  2        END;
;  385          0516  2
;  386          0517  2    IF .eol_flag
;  387          0518  2    THEN
;  388          0519  3        BEGIN
;  389          0520  3        ! The link field is zero
;  390          0521  3        !
;  391          0522  3        noun_node [dbg$l_noun_link] = 0;
;  392          0523  3        RETURN sts$k_success;
;  393          0524  2        END;
;  394          0525  2
;  395          0526  3    IF NOT (.dblquote_flag OR .quote_flag)
;  396          0527  2    THEN
;  397          0528  3        BEGIN
;  398          0529  3
```

```
;  399          0530  3          ! We now attempt to read a module name
;  400          0531  3          !
;  401          0532  3          name_buf[0] = 0;
;  402          0533  3          string_ptr = .input_desc[dsc$a_pointer];
;  403          0534  3          length = .input_desc[dsc$w_length];
;  404          0535  3
;  405          0536  3          ! read past leading blanks
;  406          0537  3          !
;  407          0538  3          WHILE .length GTR 0 DO
;  408          0539  4              BEGIN
;  409          0540  4              char = ch$rchar_a(string_ptr);
;  410          0541  4              length = .length - 1;
;  411          0542  4              IF .char NEQ dbg$k_blank
;  412          0543  4              THEN
;  413          0544  4                  EXITLOOP;
;  414          0545  3              END;
;  415          0546  3
;  416          0547  3          ! If the length reaches zero then it is an error
;  417          0548  3          ! This should not happen.
;  418          0549  3          !
;  419          0550  3          IF .length EQL 0 AND .char EQL dbg$k_blank
;  420          0551  3          THEN
;  421          0552  4              BEGIN
;  422          0553  4              $DBG_ERROR('DBGNSEARC\DBG$NPARSE_SEARCH');
;  423          0554  3              END;
;  424          0555  3
;  425          0556  3          ! Read until we reach a separating character.
;  426          0557  3          ! Place the characters into name_buf as we read them.
;  427          0558  3          !
;  428          0559  3          WHILE .length GTR 0 DO
;  429          0560  4              BEGIN
;  430          0561  4              IF .char EQL '\'
;  431          0562  4              OR .char EQL ':' OR .char EQL ' '
;  432          0563  4              THEN
;  433          0564  5                  BEGIN
;  434          0565  5                  ! Correct for loop going one too far.
;  435          0566  5                  string_ptr = ch$plus (.string_ptr, -1);
;  436          0567  5                  length = .length + 1;
;  437          0568  5                  EXITLOOP;
;  438          0569  4                  END;
;  439          0570  4              name_buf[0] = .name_buf[0] + 1;
;  440          0571  4              name_buf[.name_buf[0]] = .char;
;  441          0572  4              char = ch$rchar_a(string_ptr);
;  442          0573  4              length = .length - 1;
;  443          0574  3              END;
;  444          0575  3
;  445          0576  3          ! Decide whether what the user entered seems to be a name or a number.
;  446          0577  3          !
;  447          0578  3          IS_IT_NAME = FALSE;
;  448          0579  3          INCR J FROM 1 TO .NAME_BUF[0] DO
;  449          0580  3              IF .NAME_BUF[.J] GTR '9' OR .NAME_BUF[.J] LSS '0'
;  450          0581  3              THEN
;  451          0582  3                  IS_IT_NAME = TRUE;
;  452          0583  3
;  453          0584  3          ! Now decide whether we are looking at a module name.
;  454          0585  3          ! Convert the name to an rst pointer.
;  455          0586  3          !
```

```
;  456          0587  3          noun_node[dbg$l_adjective_ptr] =
;  457          0588  3              dbg$sta_getsourcemod(name_buf);
;  458          0589  3
;  459          0590  3          ! If the above routine returned a non-zero value then the user entered
;  460          0591  3          ! an valid module.
;  461          0592  3          !
;  462          0593  3          IF .noun_node[dbg$l_adjective_ptr] NEQ 0
;  463          0594  3          THEN
;  464          0595  4              BEGIN
;  465          0596  4              ! Update the input descriptor
;  466          0597  4              !
;  467          0598  4              input_desc[dsc$a_pointer] = .string_ptr;
;  468          0599  4              input_desc[dsc$w_length] = .length;
;  469          0600  4
;  470          0601  4              ! Eat the backslash which may follow the module name.
;  471          0602  4              ! If it is not there, don't worry about it.
;  472          0603  4              !
;  473          0604  4              dbg$nmatch (.input_desc, dbg$cs_backslash, 1);
;  474          0605  4
;  475          0606  4              ! fill in new value of modrstptr
;  476          0607  4              modrstptr = .noun_node[dbg$l_adjective_ptr];
;  477          0608  4
;  478          0609  4              END ! pick up module name
;  479          0610  4
;  480          0611  3          ELSE
;  481          0612  3
;  482          0613  4              BEGIN ! decide whether to put out an error message.
;  483          0614  4
;  484          0615  4              ! If the user seems to have entered a name but it is
;  485          0616  4              ! not a valid module name then issue an error message.
;  486          0617  4              !
;  487          0618  4              IF .is_it_name
;  488          0619  4              THEN
;  489          0620  5                  BEGIN
;  490          0621  5                  .message_vect = dbg$nmake_arg_vect (
;  491          0622  5                          dbg$_nosuchmodu, 1, name_buf);
;  492          0623  5                  RETURN sts$k_severe;
;  493          0624  4                  END;
;  494          0625  4
;  495          0626  4              ! Fill in a module based on current scope
;  496          0627  4              !
;  497          0628  4              modrstptr = dbg$sta_getsourcemod(0);
;  498          0629  4
;  499          0630  4              IF .modrstptr EQL 0
;  500          0631  4              THEN
;  501          0632  5                  BEGIN
;  502          0633  5
;  503          0634  5                  ! If this is zero, then we have no scope with
;  504          0635  5                  ! which to supply a module. Report an error.
;  505          0636  5                  !
;  506          0637  5                  .message_vect = dbg$nmake_arg_vect(
;  507          0638  5                          dbg$_noscope, 1, .dbg$src_next_lnum);
;  508          0639  5                  RETURN sts$k_severe;
;  509          0640  5                  END
;  510          0641  5
;  511          0642  4              ELSE
;  512          0643  4                  ! we have found a module.
```

```
;  513          0644  4              ! fill in the adjective field.
;  514          0645  4              noun_node[dbg$l_adjective_ptr] = .modrstptr;
;  515          0646  4
;  516          0647  3          END;                          ! Fill in default module
;  517          0648  3
;  518          0649  3
;  519          0650  3          ! Read in the low line number.  First we scan ahead to see if such a
;  520          0651  3          ! number was specified.  If it was, we pick it up and make that the
;  521          0652  3          ! low line number of the range.  If no number is specified, we make
;  522          0653  3          ! the range large enough to include the entire module.
;  523          0654  3          !
;  524          0655  3          LOW_LNUM = 1;
;  525          0656  3          HIGH_LNUM = 2000000000;
;  526          0657  3          TPTR = .INPUT_DESC[DSC$A_POINTER];
;  527          0658  3          I = 0;
;  528          0659  3          WHILE (.TPTR[.I] EQL DBG$K_BLANK) OR (.TPTR[.I] EQL DBG$K_TAB) DO
;  529          0660  3              I = .I + 1;
;  530          0661  3
;  531          0662  4          IF (.TPTR[.I] GEQ '0') AND (.TPTR[.I] LEQ '9')
;  532          0663  3          THEN
;  533          0664  3              DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, LOW_LNUM);
;  534          0665  3
;  535          0666  3
;  536          0667  3          ! Now look for colon which signifies that the user has also specified a
;  537          0668  3          ! high line number.  If the colon is present, we scan ahead to see if a
;  538          0669  3          ! high line number was specified.  If it was, we pick it up and make
;  539          0670  3          ! that the high line number of the range.  If no number was specified,
;  540          0671  3          ! we leave the range large enough to include the entire module.
;  541          0672  3          !
;  542          0673  3          IF DBG$NMATCH(.INPUT_DESC, DBG$CS_COLON, 1)
;  543          0674  3          THEN
;  544          0675  4              BEGIN
;  545          0676  4              TPTR = .INPUT_DESC[DSC$A_POINTER];
;  546          0677  4              I = 0;
;  547          0678  4              WHILE (.TPTR[.I] EQL DBG$K_BLANK) OR (.TPTR[.I] EQL DBG$K_TAB) DO
;  548          0679  4                  I = .I + 1;
;  549          0680  4
;  550          0681  5              IF (.TPTR[.I] GEQ '0') AND (.TPTR[.I] LEQ '9')
;  551          0682  4              THEN
;  552          0683  4                  DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, HIGH_LNUM);
;  553          0684  4
;  554          0685  3              END;
;  555          0686  3
;  556          0687  3
;  557          0688  3          ! Fill in the fields of noun_node
;  558          0689  3          !
;  559          0690  3          NOUN_NODE[DBG$L_NOUN_VALUE] = .LOW_LNUM;
;  560          0691  3          NOUN_NODE[DBG$L_NOUN_VALUE2] = .HIGH_LNUM;
;  561          0692  3
;  562          0693  3
;  563          0694  3          ! Now that we have read the module and/or line range information,
;  564          0695  3          ! look for quote or double quote once again.
;  565          0696  3          !
;  566          0697  3          IF dbg$nmatch (.input_desc, dbg$cs_dblquote, 1)
;  567          0698  3          THEN
;  568          0699  4              BEGIN
;  569          0700  4              dblquote_flag = TRUE;
```

```
;  570          0701  4          delimiter = dbg$k_dblquote;
;  571          0702  4          END
;  572          0703  3      ELSE
;  573          0704  3          IF dbg$nmatch (.input_desc, dbg$cs_quote, 1)
;  574          0705  3          THEN
;  575          0706  4              BEGIN
;  576          0707  4              quote_flag = TRUE;
;  577          0708  4              delimiter = dbg$k_quote;
;  578          0709  4              END
;  579          0710  3          ELSE
;  580          0711  4              BEGIN
;  581          0712  4              ! No quotes specified. We want to strip off leading
;  582          0713  4              ! white space before reading the string
;  583          0714  4              !
;  584          0715  4              LOCAL
;  585          0716  4                  char;
;  586          0717  4              WHILE  .input_desc [dsc$w_length] GTR 0 DO
;  587          0718  5                  BEGIN
;  588          0719  5                  char = ch$rchar_a(input_desc[dsc$a_pointer]);
;  589          0720  5                  input_desc[dsc$w_length]  = .input_desc[dsc$w_length] - 1;
;  590          0721  5                  IF .char NEQ dbg$k_blank THEN EXITLOOP;
;  591          0722  4                  END;
;  592          0723  4
;  593          0724  4              ! Back up since the above read one too far.
;  594          0725  4              !
;  595          0726  4              input_desc[dsc$a_pointer] =
;  596          0727  4                      ch$plus(.input_desc[dsc$a_pointer],-1);
;  597          0728  4              input_desc [dsc$w_length] = .input_desc [dsc$w_length] + 1;
;  598          0729  3              END;
;  599          0730  3
;  600          0731  2          END;
;  601          0732  2
;  602          0733  2      ! At this point we have accepted the module and line number range
;  603          0734  2      ! (or used defaults). We now expect to see the search string.
;  604          0735  2      !
;  605          0736  2      IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  606          0737  2      THEN
;  607          0738  3          BEGIN
;  608          0739  3          ! First check for the case of SEARCH ''  or SEARCH ' ,
;  609          0740  3          ! both of which we shall treat as errors.
;  610          0741  3          IF .dblquote_flag OR .quote_flag
;  611          0742  3          THEN
;  612          0743  3              SIGNAL(dbg$_noend,3,.input_desc,1,delimiter);
;  613          0744  3
;  614          0745  3          ! No search string specified, so we just fill a zero into the
;  615          0746  3          ! noun link field and return.
;  616          0747  3          !
;  617          0748  3          noun_node[dbg$l_noun_link] = 0;
;  618          0749  3          RETURN sts$k_success;
;  619          0750  2          END;
;  620          0751  2
;  621          0752  2      ! If we reach this point then the user has specified a search string.
;  622          0753  2      ! So we create and link new noun node for the search string.
;  623          0754  2      !
;  624          0755  2      link = noun_node[dbg$l_noun_link];
;  625          0756  2      NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE);
;  626          0757  2      .link = .noun_node;
```

```
;  627          0758  2
;  628          0759  2
;  629          0760  2     ! Read the string and leave a pointer in the second noun node
;  630          0761  2     !
;  631          0762  2     IF NOT dbg$naccept_string (.input_desc, noun_node[dbg$l_noun_value],
;  632          0763  2                                .delimiter, FALSE, .message_vect, TRUE)
;  633          0764  2     THEN
;  634          0765  2         RETURN sts$k_severe;
;  635          0766  2
;  636          0767  2     RETURN sts$k_success;
;  637          0768  2
;  638          0769  1     END; ! dbg$nparse_search


                                                                .TITLE  DBGNSEARC
                                                                .IDENT  \V04-000\

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

                                                        03  00000 P.AAA:   .BYTE   3                                       ;
                                                    4C  4C  41  00001       .ASCII  \ALL\                                   ;
                                                        0A  00004 P.AAB:    .BYTE   10                                      ;
                                52  45  49  46  49  54  4E  45  44  49  00005       .ASCII  \IDENTIFIER\                    ;
                                                        04  0000F P.AAC:    .BYTE   4                                       ;
                                                    54  58  45  4E  00010       .ASCII  \NEXT\                          ;
                                                        06  00014 P.AAD:    .BYTE   6                                       ;
                                            47  4E  49  52  54  53  00015       .ASCII  \STRING\                        ;
                                                    5C  01  0001B P.AAE:    .BYTE   1, 92                                   ;
                                                    3A  01  0001D P.AAF:    .BYTE   1, 58                                   ;
                                                    0D  01  0001F P.AAG:    .BYTE   1, 13                                   ;
                                                    27  01  00021 P.AAH:    .BYTE   1, 39                                   ;
                                                    22  01  00023 P.AAI:    .BYTE   1, 34                                   ;
                                                    2F  01  00025 P.AAJ:    .BYTE   1, 47                                   ;
24  47  42  44  5C  43  52  41  45  53  4E  47  42  44  1B  00027 P.AAK:    .ASCII  <27>\DBGNSEARC\<92>\DBG$NPARSE_SEARCH\   ;
        48  43  52  41  45  53  5F  45  53  52  41  50  4E  00036                                                           ;

                                                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                                                        00000 NAME_BUF:
                                                                .BLKB   81

                                                        DBG$CS_ALL=         P.AAA
                                                        DBG$CS_IDENT=       P.AAB
                                                        DBG$CS_NEXT=        P.AAC
                                                        DBG$CS_STRING=      P.AAD
                                                        DBG$CS_BACKSLASH=   P.AAE
                                                        DBG$CS_COLON=       P.AAF
                                                        DBG$CS_CR=          P.AAG
                                                        DBG$CS_QUOTE=       P.AAH
                                                        DBG$CS_DBLQUOTE=    P.AAI
                                                        DBG$CS_SLASH=       P.AAJ
                                                                .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
                                                                .EXTRN  DBG$NEWLINE, DBG$NMAKE_ARG_VECT
                                                                .EXTRN  DBG$NMATCH, DBG$NNEXT_WORD
                                                                .EXTRN  DBG$NOUT_ARG_VECT
                                                                .EXTRN  DBG$NSAVE_DECIMAL_INTEGER
                                                                .EXTRN  DBG$NSAVE_STRING
```

```
                                        .EXTRN  DBG$NSYNTAX_ERROR
                                        .EXTRN  DBG$PRINT, DBG$SET_SEARCH_LVL
                                        .EXTRN  DBG$SRC_SEARCH_CMD
                                        .EXTRN  DBG$STA_GETSOURCEMOD
                                        .EXTRN  DBG$STA_SYMNAME
                                        .EXTRN  DBG$GB_SEARCH_PTR
                                        .EXTRN  DBG$SRC_NEXT_LNUM
                                        .EXTRN  DBG$SRC_NEXT_MODRSTPTR
                                        .EXTRN  DBG$SRC_SEARCH_STRING

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

                            0FFC 00000  .ENTRY  DBG$NPARSE_SEARCH, Save R2,R3,R4,R5,R6,R7,- ; 0245
                                                R8,R9,R10,R11
                    5B 00000000'  EF  9E 00002      MOVAB   DBG$CS_CR, R11                  ;
                    5E            0C  C2 00009      SUBL2   #12, SP                         ;
                                  54  D4 0000C      CLRL    ALL_FLAG                        ; 0351
                                  52  7C 0000E      CLRQ    SWITCH_FLAG                     ; 0350
                    55        04  AC  D0 00010      MOVL    INPUT_DESC, R5                  ; 0357
                                  01  DD 00014 1$:   PUSHL   #1                             ;
                              06  AB  9F 00016      PUSHAB  DBG$CS_SLASH                    ;
                                  55  DD 00019      PUSHL   R5                              ;
                00000000G     00    03  FB 0001B      CALLS   #3, DBG$NMATCH                  ;
                              03    50  E8 00022      BLBS    R0, 2$                          ;
                                008B  31 00025      BRW     8$                              ;
                                  01  DD 00028 2$:   PUSHL   #1                             ; 0362
                              E1  AB  9F 0002A      PUSHAB  DBG$CS_ALL                      ;
                                  55  DD 0002D      PUSHL   R5                              ;
                00000000G     00    03  FB 0002F      CALLS   #3, DBG$NMATCH                  ;
                              01    50  D1 00036      CMPL    R0, #1                          ;
                                    0B  12 00039      BNEQ    4$                              ;
                          52        01  D0 0003B      MOVL    #1, SWITCH_FLAG                 ; 0364
                          54        01  D0 0003E      MOVL    #1, ALL_FLAG                    ; 0365
                          56        01  D0 00041      MOVL    #1, ALL_VALUE                   ; 0366
                                    CE  11 00044 3$:   BRB     1$                              ; 0359
                                  01  DD 00046 4$:   PUSHL   #1                             ; 0369
                              E5  AB  9F 00048      PUSHAB  DBG$CS_IDENT                    ;
                                  55  DD 0004B      PUSHL   R5                              ;
                00000000G     00    03  FB 0004D      CALLS   #3, DBG$NMATCH                  ;
                              01    50  D1 00054      CMPL    R0, #1                          ;
                                    0B  12 00057      BNEQ    5$                              ;
                          52        01  D0 00059      MOVL    #1, SWITCH_FLAG                 ; 0371
                          53        01  D0 0005C      MOVL    #1, IDENT_FLAG                  ; 0372
                          57        01  D0 0005F      MOVL    #1, IDENT_VALUE                 ; 0373
                                    B0  11 00062      BRB     1$                              ; 0359
                                  01  DD 00064 5$:   PUSHL   #1                             ; 0376
                              F0  AB  9F 00066      PUSHAB  DBG$CS_NEXT                     ;
                                  55  DD 00069      PUSHL   R5                              ;
                00000000G     00    03  FB 0006B      CALLS   #3, DBG$NMATCH                  ;
                              01    50  D1 00072      CMPL    R0, #1                          ;
                                    0A  12 00075      BNEQ    6$                              ;
                          52        01  D0 00077      MOVL    #1, SWITCH_FLAG                 ; 0378
                          54        01  D0 0007A      MOVL    #1, ALL_FLAG                    ; 0379
                                    56  D4 0007D      CLRL    ALL_VALUE                       ; 0380
                                    93  11 0007F      BRB     1$                              ; 0359
                                  01  DD 00081 6$:   PUSHL   #1                             ; 0383
                              F5  AB  9F 00083      PUSHAB  DBG$CS_STRING                   ;
```

```
                                   55  DD 00086          PUSHL   R5                                ;
          00000000G  00            03  FB 00088          CALLS   #3, DBG$NMATCH                    ;
                     01            50  D1 0008F          CMPL    R0, #1                            ;
                                   0A  12 00092          BNEQ    7$                                ;
                     52            01  D0 00094          MOVL    #1, SWITCH_FLAG                   ; 0385
                     53            01  D0 00097          MOVL    #1, IDENT_FLAG                    ; 0386
                                   57  D4 0009A          CLRL    IDENT_VALUE                       ; 0387
                                   A6  11 0009C          BRB     3$                                ; 0359
                                   55  DD 0009E  7$:     PUSHL   R5                                ; 0393
          00000000G  00            01  FB 000A0          CALLS   #1, DBG$NNEXT_WORD                ;
                                   50  DD 000A7          PUSHL   R0                                ;
          00000000G  00            01  FB 000A9          CALLS   #1, DBG$NSYNTAX_ERROR             ;
                                 01F1  31 000B0          BRW     33$                               ;
                     3E            52  E9 000B3  8$:     BLBC    SWITCH_FLAG, 11$                  ; 0402
     59         08   AC            04  C1 000B6          ADDL3   #4, VERB_NODE, LINK               ; 0406
                     01            54  D1 000BB          CMPL    ALL_FLAG, #1                      ; 0411
                                   16  12 000BE          BNEQ    9$                                ;
                                   03  DD 000C0          PUSHL   #3                                ; 0415
          00000000G  00            01  FB 000C2          CALLS   #1, DBG$GET_TEMPMEM               ;
                     69            50  D0 000C9          MOVL    ADVERB_NODE, (LINK)               ; 0416
                     59       08   A0  9E 000CC          MOVAB   8(R0), LINK                       ; 0417
                                   60  94 000D0          CLRB    (ADVERB_NODE)                     ; 0418
                04   A0            56  D0 000D2          MOVL    ALL_VALUE, 4(ADVERB_NODE)         ; 0419
                     01            53  D1 000D6  9$:     CMPL    IDENT_FLAG, #1                    ; 0422
                                   17  12 000D9          BNEQ    10$                               ;
                                   03  DD 000DB          PUSHL   #3                                ; 0426
          00000000G  00            01  FB 000DD          CALLS   #1, DBG$GET_TEMPMEM               ;
                     69            50  D0 000E4          MOVL    ADVERB_NODE, (LINK)               ; 0427
                     59       08   A0  9E 000E7          MOVAB   8(R0), LINK                       ; 0428
                     60            01  90 000EB          MOVB    #1, (ADVERB_NODE)                 ; 0429
                04   A0            57  D0 000EE          MOVL    IDENT_VALUE, 4(ADVERB_NODE)       ; 0430
                                   69  D4 000F2  10$:    CLRL    (LINK)                            ; 0437
                                   04  DD 000F4  11$:    PUSHL   #4                                ; 0444
          00000000G  00            01  FB 000F6          CALLS   #1, DBG$GET_TEMPMEM               ;
                     54            50  D0 000FD          MOVL    R0, NOUN_NODE                     ;
                     50       08   AC  D0 00100          MOVL    VERB_NODE, R0                     ; 0445
                08   A0            54  D0 00104          MOVL    NOUN_NODE, 8(R0)                  ;
                                   01  DD 00108          PUSHL   #1                                ; 0450
                             0820  8F  BB 0010A          PUSHR   #^M<R5,R11>                       ;
          00000000G  00            03  FB 0010E          CALLS   #3, DBG$NMATCH                    ;
                     09            50  E9 00115          BLBC    R0, 12$                           ;
                     52            01  D0 00118          MOVL    #1, EOL_FLAG                      ; 0453
                                   5A  D4 0011B          CLRL    DBLQUOTE_FLAG                     ; 0454
                                   58  D4 0011D          CLRL    QUOTE_FLAG                        ; 0455
                                   40  11 0011F          BRB     15$                               ; 0450
                                   52  D4 00121  12$:    CLRL    EOL_FLAG                          ; 0459
                                   01  DD 00123          PUSHL   #1                                ; 0463
                              04   AB  9F 00125          PUSHAB  DBG$CS_DBLQUOTE                   ;
                                   55  DD 00128          PUSHL   R5                                ;
          00000000G  00            03  FB 0012A          CALLS   #3, DBG$NMATCH                    ;
                     0B            50  E9 00131          BLBC    R0, 13$                           ;
                     5A            01  D0 00134          MOVL    #1, DBLQUOTE_FLAG                 ; 0466
                                   58  D4 00137          CLRL    QUOTE_FLAG                        ; 0467
                08   AE            22  D0 00139          MOVL    #34, DELIMITER                    ; 0468
                                   22  11 0013D          BRB     15$                               ; 0463
                                   5A  D4 0013F  13$:    CLRL    DBLQUOTE_FLAG                     ; 0472
                                   01  DD 00141          PUSHL   #1                                ; 0473
```

```
                                02    AB  9F 00143          PUSHAB  DBG$CS_QUOTE                                    ;
                                      55  DD 00146          PUSHL   R5                                              ;
                00000000G       00    03  FB 00148          CALLS   #3, DBG$NMATCH                                  ;
                                09    50  E9 0014F          BLBC    R0, 14$                                         ;
                                58    01  D0 00152          MOVL    #1, QUOTE_FLAG                                  ; 0476
                          08    AE    27  D0 00155          MOVL    #39, DELIMITER                                  ; 0477
                                      06  11 00159          BRB     15$                                             ; 0473
                                      58  D4 0015B 14$:      CLRL    QUOTE_FLAG                                      ; 0481
                          08    AE    0D  D0 0015D          MOVL    #13, DELIMITER                                  ; 0482
                                06    5A  E8 00161 15$:      BLBS    DBLQUOTE_FLAG, 16$                              ; 0487
                                03    58  E8 00164          BLBS    QUOTE_FLAG, 16$                                 ;
                                35    52  E9 00167          BLBC    EOL_FLAG, 18$                                   ;
                                   00000000G  00  D5 0016A 16$:      TSTL    DBG$SRC_NEXT_MODRSTPTR                  ; 0493
                                      10  12 00170          BNEQ    17$                                             ;
                                00028CE0    8F  DD 00172          PUSHL   #167136                                 ; 0497
                00000000G       00    01  FB 00178          CALLS   #1, DBG$NMAKE_ARG_VECT                          ;
                                    0122  31 0017F          BRW     33$                                             ;
                          04    A4 00000000G  00  D0 00182 17$:      MOVL    DBG$SRC_NEXT_MODRSTPTR, 4(NOUN_NODE)    ; 0504
                          64    00000000G  00  D0 0018A          MOVL    DBG$SRC_NEXT_LNUM, (NOUN_NODE)          ; 0508
                          0C    A4 77359400    8F  D0 00191          MOVL    #2000000000, 12(NOUN_NODE)              ; 0513
                                03    52  E9 00199          BLBC    EOL_FLAG, 18$                                   ; 0517
                                    0202  31 0019C          BRW     49$                                             ;
                                03    5A  E9 0019F 18$:      BLBC    DBLQUOTE_FLAG, 20$                              ; 0526
                                    01D0  31 001A2 19$:      BRW     47$                                             ;
                                FA    58  E8 001A5 20$:      BLBS    QUOTE_FLAG, 19$                                 ;
                                00000000'   EF  94 001A8          CLRB    NAME_BUF                                ; 0532
                                56    04    A5  9E 001AE          MOVAB   4(R5), R6                               ; 0533
                                57          66  D0 001B2          MOVL    (R6), STRING_PTR                        ;
                                52          65  3C 001B5          MOVZWL  (R5), LENGTH                            ; 0534
                                            52  D5 001B8 21$:      TSTL    LENGTH                                  ; 0538
                                            0A  15 001BA          BLEQ    22$                                     ;
                                53          87  9A 001BC          MOVZBL  (STRING_PTR)+, CHAR                     ; 0540
                                            52  D7 001BF          DECL    LENGTH                                  ; 0541
                                20          53  D1 001C1          CMPL    CHAR, #32                               ; 0542
                                            F2  13 001C4          BEQL    21$                                     ;
                                            52  D5 001C6 22$:      TSTL    LENGTH                                  ; 0550
                                            17  12 001C8          BNEQ    23$                                     ;
                                <0          53  D1 001CA          CMPL    CHAR, #32                               ;
                                            12  12 001CD          BNEQ    23$                                     ;
                                      08    AB  9F 001CF          PUSHAB  P.AAK                                   ; 0553
                                            01  DD 001D2          PUSHL   #1                                      ;
                                00028362    8F  DD 001D4          PUSHL   #164706                                 ;
                00000000G       00          03  FB 001DA          CALLS   #3, LIB$SIGNAL                          ;
                                            52  D5 001E1 23$:      TSTL    LENGTH                                  ; 0559
                                            35  15 001E3          BLEQ    26$                                     ;
                0000005C        8F          53  D1 001E5          CMPL    CHAR, #92                               ; 0561
                                            0A  13 001EC          BEQL    24$                                     ;
                                3A          53  D1 001EE          CMPL    CHAR, #58                               ; 0562
                                            05  13 001F1          BEQL    24$                                     ;
                                20          53  D1 001F3          CMPL    CHAR, #32                               ;
                                            06  12 001F6          BNEQ    25$                                     ;
                                            57  D7 001F8 24$:      DECL    STRING_PTR                              ; 0566
                                            52  D6 001FA          INCL    LENGTH                                  ; 0567
                                            1C  11 001FC          BRB     26$                                     ; 0564
                                00000000'   EF  96 001FE 25$:      INCB    NAME_BUF                                ; 0570
                                50 00000000'   EF  9A 00204          MOVZBL  NAME_BUF, R0                            ; 0571
                00000000'EF40               53  90 0020B          MOVB    CHAR, NAME_BUF[R0]                      ;
```

```
                    53              87  9A 00213          MOVZBL  (STRING_PTR)+, CHAR                     ; 0572
                                    52  D7 00216          DECL    LENGTH                                  ; 0573
                                    C7  11 00218          BRB     23$                                     ; 0559
                                    53  D4 0021A 26$:     CLRL    IS_IT_NAME                              ; 0578
                    51 00000000'    EF  9A 0021C          MOVZBL  NAME_BUF, R1                            ; 0579
                                    50  D4 00223          CLRL    J                                       ;
                                    17  11 00225          BRB     29$                                     ;
                    39 00000000'EF  40  91 00227 27$:     CMPB    NAME_BUF[J], #57                        ; 0580
                                    0A  1A 0022F          BGTRU   28$                                     ;
                    30 00000000'EF  40  91 00231          CMPB    NAME_BUF[J], #48                        ;
                                    03  1E 00239          BGEQU   29$                                     ;
                    53              01  D0 0023B 28$:     MOVL    #1, IS_IT_NAME                          ; 0582
                E5  50              51  F3 0023E 29$:     AOBLEQ  R1, J, 27$                              ; 0580
                    00000000'       EF  9F 00242          PUSHAB  NAME_BUF                                ; 0588
            00000000G   00          01  FB 00248          CALLS   #1, DBG$STA_GETSOURCEMOD                ;
                    04  A4          50  D0 0024F          MOVL    R0, 4(NOUN_NODE)                        ;
                                    1A  13 00253          BEQL    30$                                     ; 0593
                    66              57  D0 00255          MOVL    STRING_PTR, (R6)                        ; 0598
                    65              52  B0 00258          MOVW    LENGTH, (R5)                            ; 0599
                                    01  DD 0025B          PUSHL   #1                                      ; 0604
                            FC      AB  9F 0025D          PUSHAB  DBG$CS_BACKSLASH                        ;
                                    55  DD 00260          PUSHL   R5                                      ;
            00000000G   00          03  FB 00262          CALLS   #3, DBG$NMATCH                          ;
                    50          04  A4  D0 00269          MOVL    4(NOUN_NODE), MODRSTPTR                 ; 0607
                                    40  11 0026D          BRB     35$                                     ; 0593
                    10              53  E9 0026F 30$:     BLBC    IS_IT_NAME, 31$                         ; 0618
                        00000000'   EF  9F 00272          PUSHAB  NAME_BUF                                ; 0621
                                    01  DD 00278          PUSHL   #1                                      ;
                        000281E8    8F  DD 0027A          PUSHL   #164328                                 ;
                                    1B  11 00280          BRB     32$                                     ;
                                    7E  D4 00282 31$:     CLRL    -(SP)                                   ; 0628
            00000000G   00          01  FB 00284          CALLS   #1, DBG$STA_GETSOURCEMOD                ;
                                    50  D5 0028B          TSTL    MODRSTPTR                               ; 0630
                                    1C  12 0028D          BNEQ    34$                                     ;
                        00000000G   00  DD 0028F          PUSHL   DBG$SRC_NEXT_LNUM                       ; 0638
                                    01  DD 00295          PUSHL   #1                                      ; 0637
                        00028972    8F  DD 00297          PUSHL   #166258                                 ;
            00000000G   00          03  FB 0029D 32$:     CALLS   #3, DBG$NMAKE_ARG_VECT                  ;
                    0C  BC          50  D0 002A4 33$:     MOVL    R0, @MESSAGE_VECT                       ;
                                0124    31 002A8          BRW     51$                                     ; 0639
                    04  A4          50  D0 002AB 34$:     MOVL    MODRSTPTR, 4(NOUN_NODE)                 ; 0645
                        6E          01  D0 002AF 35$:     MOVL    #1, LOW_LNUM                            ; 0655
                    04  AE 77359400   8F  D0 002B2          MOVL    #2000000000, HIGH_LNUM                  ; 0656
                        57          66  D0 002BA          MOVL    (R6), TPTR                              ; 0657
                                    52  D4 002BD          CLRL    I                                       ; 0658
                            20      6247 91 002BF 36$:     CMPB    (I)[TPTR], #32                          ; 0659
                                    06  13 002C3          BEQL    37$                                     ;
                            09      6247 91 002C5          CMPB    (I)[TPTR], #9                           ;
                                    04  12 002C9          BNEQ    38$                                     ;
                                    52  D6 002CB 37$:     INCL    I                                       ; 0660
                                    F0  11 002CD          BRB     36$                                     ;
                            30      6247 91 002CF 38$:     CMPB    (I)[TPTR], #48                          ; 0662
                                    11  1F 002D3          BLSSU   39$                                     ;
                            39      6247 91 002D5          CMPB    (I)[TPTR], #57                          ;
                                    0B  1A 002D9          BGTRU   39$                                     ;
                                4020    8F  BB 002DB          PUSHR   #^M<R5,SP>                          ; 0664
            00000000G   00          02  FB 002DF          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER           ;
```

```
                         01 DD 002E6 39$:   PUSHL    #1                                ; 0673
                   FE    AB 9F 002E8        PUSHAB   DBG$CS_COLON                      ;
                         55 DD 002EB        PUSHL    R5                                ;
00000000G  00            03 FB 002ED        CALLS    #3, DBG$NMATCH                    ;
           2D            50 E9 002F4        BLBC     R0, 43$                           ;
           57            66 D0 002F7        MOVL     (R6), TPTR                        ; 0676
                         52 D4 002FA        CLRL     I                                 ; 0677
           20          6247 91 002FC 40$:   CMPB     (I)[TPTR], #32                    ; 0678
                         06 13 00300        BEQL     41$                               ;
           09          6247 91 00302        CMPB     (I)[TPTR], #9                     ;
                         04 12 00306        BNEQ     42$                               ;
                         52 D6 00308 41$:   INCL     I                                 ; 0679
                         F0 11 0030A        BRB      40$                               ;
           30          6247 91 0030C 42$:   CMPB     (I)[TPTR], #48                    ; 0681
                         12 1F 00310        BLSSU    43$                               ;
           39          6247 91 00312        CMPB     (I)[TPTR], #57                    ;
                         0C 1A 00316        BGTRU    43$                               ;
                   04    AE 9F 00318        PUSHAB   HIGH_LNUM                         ; 0683
                         55 DD 0031B        PUSHL    R5                                ;
00000000G  00            02 FB 0031D        CALLS    #2, DBG$NSAVE_DECIMAL_INTEGER     ;
           64            6E D0 00324 43$:   MOVL     LOW_LNUM, (NOUN_NODE)             ; 0690
     0C    A4      04    AE D0 00327        MOVL     HIGH_LNUM, 12(NOUN_NODE)          ; 0691
                         01 DD 0032C        PUSHL    #1                                ; 0697
                   04    AB 9F 0032E        PUSHAB   DBG$CS_DBLQUOTE                   ;
                         55 DD 00331        PUSHL    R5                                ;
00000000G  00            03 FB 00333        CALLS    #3, DBG$NMATCH                    ;
           09            50 E9 0033A        BLBC     R0, 44$                           ;
           5A            01 D0 0033D        MOVL     #1, DBLQUOTE_FLAG                 ; 0700
     08    AE            22 D0 00340        MOVL     #34, DELIMITER                    ; 0701
                         2F 11 00344        BRB      47$                               ; 0697
                         01 DD 00346 44$:   PUSHL    #1                                ; 0704
                   02    AB 9F 00348        PUSHAB   DBG$CS_QUOTE                      ;
                         55 DD 0034B        PUSHL    R5                                ;
00000000G  00            03 FB 0034D        CALLS    #3, DBG$NMATCH                    ;
           09            50 E9 00354        BLBC     R0, 45$                           ;
           58            01 D0 00357        MOVL     #1, QUOTE_FLAG                    ; 0707
     08    AE            27 D0 0035A        MOVL     #39, DELIMITER                    ; 0708
                         15 11 0035E        BRB      47$                               ; 0704
                         65 B5 00360 45$:   TSTW     (R5)                              ; 0717
                         0D 13 00362        BEQL     46$                               ;
           50      00    B6 9A 00364        MOVZBL   @0(R6), CHAR                      ; 0719
                         66 D6 00368        INCL     (R6)                              ;
                         65 B7 0036A        DECW     (R5)                              ; 0720
           20            50 D1 0036C        CMPL     CHAR, #32                         ; 0721
                         EF 13 0036F        BEQL     45$                               ;
                         66 D7 00371 46$:   DECL     (R6)                              ; 0727
                         65 B6 00373        INCW     (R5)                              ; 0728
                         01 DD 00375 47$:   PUSHL    #1                                ; 0736
                 0820    8F BB 00377        PUSHR    #^M<R5,R11>                       ;
00000000G  00            03 FB 0037B        CALLS    #3, DBG$NMATCH                    ;
           21            50 E9 00382        BLBC     R0, 50$                           ;
           03            5A E8 00385        BLBS     DBLQUOTE_FLAG, 48$                ; 0741
           16            58 E9 00388        BLBC     QUOTE_FLAG, 49$                   ;
                   08    AE 9F 0038B 48$:   PUSHAB   DELIMITER                         ; 0743
                         01 DD 0038E        PUSHL    #1                                ;
                         55 DD 00390        PUSHL    R5                                ;
                         03 DD 00392        PUSHL    #3                                ;
```

```
                              000281D0   8F   DD 00394          PUSHL   #164304                          ;
          00000000G  00                  05   FB 0039A          CALLS   #5, LIB$SIGNAL                   ;
                               08        A4   D4 003A1  49$:    CLRL    8(NOUN_NODE)                     ; 0748
                                         2D   11 003A4          BRB     52$                              ; 0749
                     59        08        A4   9E 003A6  50$:    MOVAB   8(R4), LINK                      ; 0755
                                         04   DD 003AA          PUSHL   #4                               ; 0756
          00000000G  00                  01   FB 003AC          CALLS   #1, DBG$GET_TEMPMEM              ;
                     54                  50   D0 003B3          MOVL    R0, NOUN_NODE                    ;
                     69                  54   D0 003B6          MOVL    NOUN_NODE, (LINK)                ; 0757
                                         01   DD 003B9          PUSHL   #1                               ; 0762
                               0C        AC   DD 003BB          PUSHL   MESSAGE_VECT                     ; 0763
                                         7E   D4 003BE          CLRL    -(SP)                            ; 0762
                               14        AE   DD 003C0          PUSHL   DELIMITER                        ; 0763
                                         54   DD 003C3          PUSHL   NOUN_NODE                        ; 0762
                                         55   DD 003C5          PUSHL   R5                               ;
                0000V  CF                06   FB 003C7          CALLS   #6, DBG$NACCEPT_STRING           ;
                       04                50   E8 003CC          BLBS    R0, 52$                          ;
                       50                04   D0 003CF  51$:    MOVL    #4, R0                           ; 0765
                                              04 003D2          RET                                      ;
                       50                01   D0 003D3  52$:    MOVL    #1, R0                           ; 0767
                                              04 003D6          RET                                      ; 0769

; Routine Size:  983 bytes,    Routine Base:  DBG$CODE + 0000


;  639          0770  1
;  640          0771  1
```

```
;  642          0772  1 GLOBAL ROUTINE dbg$nexecute_search (verb_node,message_vect) =
;  643          0773  1 !++
;  644          0774  1 ! Functional Description
;  645          0775  1 !
;  646          0776  1 !       This routine performs the action associated with the SEARCH
;  647          0777  1 !       command.
;  648          0778  1 !
;  649          0779  1 ! Formal Parameters
;  650          0780  1 !
;  651          0781  1 !       verb_node       - A longword containing the address of the
;  652          0782  1 !                         head (verb) node.
;  653          0783  1 !       message_vect    - The address of a longword to contain the
;  654          0784  1 !                         address of an error message vector
;  655          0785  1 !
;  656          0786  1 ! Implicit Inputs
;  657          0787  1 !
;  658          0788  1 !       The command tree contains a verb node, a linked list
;  659          0789  1 !       of one or two noun nodes, and possibly a linked list of
;  660          0790  1 !       one or two adverb nodes. (See the diagram in the header for
;  661          0791  1 !       dbg$nparse_search).
;  662          0792  1 !
;  663          0793  1 ! Implicit Outputs
;  664          0794  1 !
;  665          0795  1 !       This routine calls a  outine in DBGSOURCE which displays the
;  666          0796  1 !       source lines to the  er.
;  667          0797  1 !
;  668          0798  1 ! Routine Value
;  669          0799  1 !
;  670          0800  1 !       A completion code.
;  671          0801  1 !
;  672          0802  1 ! Completion Codes
;  673          0803  1 !
;  674          0804  1 !       sts$k_success (1)       - Success. Command executed
;  675          0805  1 !       sts$k_severe (4)        - Failure. The command could not be
;  676          0806  1 !                                 executed. An error message is constructed.
;  677          0807  1 !
;  678          0808  1 ! Side Effects
;  679          0809  1 !
;  680          0810  1 !       none
;  681          0811  1 !--
;  682          0812  2     BEGIN
;  683          0813  2
;  684          0814  2     MAP
;  685          0815  2         verb_node : REF dbg$verb_node;
;  686          0816  2
;  687          0817  2     LOCAL
;  688          0818  2         adverb_node: REF dbg$adverb_node,       ! Pointer to an adverb node
;  689          0819  2         cs_ptr: REF VECTOR[,BYTE],              ! Points to the search string
;  690          0820  2         link,                                   ! Points to a node in the
;  691          0821  2                                                 !       command execution tree.
;  692          0822  2         modnameptr,                             ! Pointer to module name
;  693          0823  2         next_flag,                              ! TRUE if /NEXT was specified
;  694          0824  2         noun_node : REF dbg$noun_node,          ! address of first noun node
;  695          0825  2         second_noun_node : REF dbg$noun_node,   ! address of second noun node
;  696          0826  2         string_flag;                            ! TRUE if /STRING was specified
;  697          0827  2
;  698          0828  2
```

```
;  699          0829  2
;  700          0830  2      noun_node = .verb_node[dbg$l_verb_object_ptr];
;  701          0831  2
;  702          0832  2
;  703          0833  2      ! Get the module name and print it.
;  704          0834  2      !
;  705          0835  2      dbg$sta_symname(.noun_node[dbg$l_adjective_ptr], modnameptr);
;  706          0836  2      DBG$PRINT(UPLIT BYTE(%ASCIC 'module !AC'),.modnameptr);
;  707          0837  2      DBG$NEWLINE();
;  708          0838  2
;  709          0839  2
;  710          0840  2      ! If the user supplied a string, copy it into DBG$SRC_SEARCH_STRING
;  711          0841  2      ! which is where the search routine expects to find it.
;  712          0842  2      !
;  713          0843  2      IF .noun_node[dbg$l_noun_link] NEQ 0
;  714          0844  2      THEN
;  715          0845  3          BEGIN
;  716          0846  3          second_noun_node = .noun_node [dbg$l_noun_link];
;  717          0847  3          cs_ptr = .second_noun_node [dbg$l_noun_value];
;  718          0848  3          dbg$src_search_string[0] = .cs_ptr[0];
;  719          0849  3          ch$move (.cs_ptr[0], cs_ptr[1], dbg$src_search_string[1]);
;  720          0850  2          END;
;  721          0851  2
;  722          0852  2
;  723          0853  2      ! Process any command overrides that may be present.
;  724          0854  2      !
;  725          0855  2      link = .verb_node [dbg$l_verb_adverb_ptr];
;  726          0856  2      IF .link NEQA 0
;  727          0857  2      THEN
;  728          0858  2          dbg$set_search_lvl (override_search);
;  729          0859  2
;  730          0860  2      WHILE .link NEQA 0 DO
;  731          0861  3          BEGIN
;  732          0862  3          adverb_node = .link;
;  733          0863  3          CASE .adverb_node [dbg$b_adverb_literal] FROM adverb_literal_all
;  734          0864  3                                                  TO adverb_literal_ident OF
;  735          0865  3              SET
;  736          0866  3
;  737          0867  3              [adverb_literal_all] : ! /ALL or /NEXT
;  738          0868  4                  BEGIN
;  739          0869  4                  dbg$gb_search_ptr[search_all] =
;  740          0870  4                      .adverb_node [dbg$l_adverb_value];
;  741          0871  4                  link = .adverb_node [dbg$l_adverb_link];
;  742          0872  3                  END;
;  743          0873  3
;  744          0874  3              [adverb_literal_ident] : ! /IDENT or /STRING
;  745          0875  4                  BEGIN
;  746          08 6  4                  dbg$gb_search_ptr [search_ident] =
;  747          0877  4                      .adverb_node [dbg$l_adverb_value];
;  748          0878  4                  link = .adverb_node [dbg$l_adverb_link];
;  749          0879  3                  END;
;  750          0880  3
;  751          0881  3              TES;
;  752          0882  3
;  753          0883  2          END;
;  754          0884  2
;  755          0885  2      IF .dbg$gb_search_ptr [search_ident]
```

```
;  756          0886  2     THEN
;  757          0887  2         string_flag = 0
;  758          0888  2     ELSE
;  759          0889  2         string_flag = 1;
;  760          0890  2     IF .dbg$gb_search_ptr [search_all]
;  761          0891  2     THEN
;  762          0892  2         next_flag = 0
;  763          0893  2     ELSE
;  764          0894  2         next_flag = 1;
;  765          0895  2
;  766          0896  2
;  767          0897  2     ! Call the routine which actually searches for the string.
;  768          0898  2     !
;  769          0899  2     dbg$src_search_cmd (
;  770          0900  2         .noun_node[dbg$l_adjective_ptr],
;  771          0901  2         .noun_node[dbg$l_noun_value],
;  772          0902  2         0,
;  773          0903  2         .noun_node[dbg$l_noun_value2],
;  774          0904  2         0,
;  775          0905  2         .string_flag,
;  776          0906  2         .next_flag);
;  777          0907  2
;  778          0908  2     RETURN sts$k_success
;  779          0909  2
;  780          0910  1     END; ! dbg$nexecute_search


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

                00043 P.AAL:    .ASCII  <10>\module !AC\                                        ;

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

                00000 DBG$NEXECUTE_SEARCH::
                00000           .ENTRY  DBG$NEXECUTE_SEARCH, Save R2,R3,R4,R5,R6,R7 ; 0772
                00002           SUBL2   #4, SP                                      ;
                00005           MOVL    VERB_NODE, R7                               ; 0830
                00009           MOVL    8(R7), NOUN_NODE                            ;
                0000D           PUSHL   SP                                          ; 0835
                0000F           PUSHL   4(NOUN_NODE)                                ;
                00012           CALLS   #2, DBG$STA_SYMNAME                         ;
                00019           PUSHL   MODNAMEPTR                                  ; 0836
                0001B           PUSHAB  P.AAL                                       ;
                00021           CALLS   #2, DBG$PRINT                               ;
                00028           CALLS   #0, DBG$NEWLINE                             ; 0837
                0002F           TSTL    8(NOUN_NODE)                                ; 0843
                00032           BEQL    1$                                          ;
                00034           MOVL    8(NOUN_NODE), SECOND_NOUN_NODE              ; 0846
                00038           MOVL    (SECOND_NOUN_NODE), CS_PTR                  ; 0847
                0003B           MOVB    (CS_PTR), DBG$SRC_SEARCH_STRING             ; 0848
                00042           MOVZBL  (CS_PTR), R1                                ; 0849
                00045           MOVC3   R1, 1(CS_PTR), DBG$SRC_SEARCH_STRING+1      ;
                0004E 1$:       MOVL    4(R7), LINK                                 ; 0855
                00052           BEQL    2$                                          ; 0856
                00054           PUSHL   #2                                          ; 0858
```

```
          00000000G  00             01  FB 00056         CALLS   #1, DBG$SET_SEARCH_LVL                    ;
                     51 00000000G   00  D0 0005D 2$:     MOVL    DBG$GB_SEARCH_PTR, R1                     ; 0869
                                    52  D5 00064 3$:     TSTL    LINK                                      ; 0860
                                    1C  13 00066         BEQL    8$                                        ;
                     50             52  D0 00068         MOVL    LINK, ADVERB_NODE                         ; 0862
       01            00             60  8F 0006B         CASEB   (ADVERB_NODE), #0, #1                     ; 0863
                   000A         0004       0006F 4$:     .WORD   5$-4$,-                                   ;
                                                                 6$-4$                                     ;
                     61         04  A0  90 00073 5$:     MOVB    4(ADVERB_NODE), (R1)                      ; 0870
                                    05  11 00077         BRB     7$                                        ; 0871
              01     A1         04  A0  90 00079 6$:     MOVB    4(ADVERB_NODE), 1(R1)                     ; 0877
                     52         08  A0  D0 0007E 7$:     MOVL    8(ADVERB_NODE), LINK                      ; 0878
                                    E0  11 00082         BRB     3$                                        ; 0860
                     04         01  A1  E9 00084 8$:     BLBC    1(R1), 9$                                 ; 0885
                                    52  D4 00088         CLRL    STRING_FLAG                               ; 0887
                                    03  11 0008A         BRB     10$                                       ;
                     52             01  D0 0008C 9$:     MOVL    #1, STRING_FLAG                           ; 0889
                     04             61  E9 0008F 10$:    BLBC    (R1), 11$                                 ; 0890
                                    50  D4 00092         CLRL    NEXT_FLAG                                 ; 0892
                                    03  11 00094         BRB     12$                                       ;
                     50             01  D0 00096 11$:    MOVL    #1, NEXT_FLAG                             ; 0894
                                    50  DD 00099 12$:    PUSHL   NEXT_FLAG                                 ; 0906
                                    52  DD 0009B         PUSHL   STRING_FLAG                               ; 0905
                                    7E  D4 0009D         CLRL    -(SP)                                     ; 0899
                                0C  A6  DD 0009F         PUSHL   12(NOUN_NODE)                             ; 0903
                                    7E  D4 000A2         CLRL    -(SP)                                     ; 0899
                                    66  DD 000A4         PUSHL   (NOUN_NODE)                               ; 0901
                                04  A6  DD 000A6         PUSHL   4(NOUN_NODE)                              ; 0900
          00000000G  00             07  FB 000A9         CALLS   #7, DBG$SRC_SEARCH_CMD                    ;
                     50             01  D0 000B0         MOVL    #1, R0                                    ; 0908
                                        04 000B3         RET                                               ; 0910

; Routine Size:  180 bytes,    Routine Base:  DBG$CODE + 03D7
```

```
;  782          0911  1 GLOBAL ROUTINE dbg$naccept_string (input_desc, result_addr,
;  783          0912  1                                    delimiter, perm_flag,
;  784          0913  1                                    message_vect, uppercase_flag) =
;  785          0914  1 !++
;  786          0915  1 ! Functional Description
;  787          0916  1 !
;  788          0917  1 !       This routine is called at the point when DBG$NPARSE_SEARCH expects to
;  789          0918  1 !       see a search string. Some possible forms of the search command are:
;  790          0919  1 !
;  791          0920  1 !                       SEARCH low:high search-string
;  792          0921  1 !                       SEARCH "search-string"
;  793          0922  1 !                       SEARCH 'search-string'
;  794          0923  1 !
;  795          0924  1 !       At the point this routine is called the leading quote, if there was one,
;  796          0925  1 !       has already been eaten. The delimiter will indicate what kind of leading
;  797          0926  1 !       quote was present, or will be <cr> if no leading quote was present.
;  798          0927  1 !
;  799          0928  1 !       This routine allocates space for the search string. It then reads
;  800          0929  1 !       characters from the input stream until it hits the delimiter,
;  801          0930  1 !       copying them to the allocated area as it reads. It also translates
;  802          0931  1 !       lower case to upper case and undoubles quotes as it is doing this.
;  803          0932  1 !       It returns the address of the counted string in RESULT_ADDR.
;  804          0933  1 !       It updates INPUT_DESC to reflect the characters read.
;  805          0934  1 !
;  806          0935  1 ! Inputs
;  807          0936  1 !
;  808          0937  1 !       input_desc      - A string descriptor for the remaining input.
;  809          0938  1 !       result_addr     - The address at which to leave the string that
;  810          0939  1 !                         is read.
;  811          0940  1 !       delimiter       - The character that terminates the string. This
;  812          0941  1 !                         will be one of:
;  813          0942  1 !                         quote, double quote, or carriage return.
;  814          0943  1 !       perm_flag       - Says whether the string will be allocated from
;  815          0944  1 !                         permanent or temporary memory.
;  816          0945  1 !       message_vect    - The argument vector for error messages.
;  817          0946  1 !       uppercase_flag  - True if the string is to be uppercased, False if
;  818          0947  1 !                         not.
;  819          0948  1 !
;  820          0949  1 ! Outputs
;  821          0950  1 !
;  822          0951  1 !       A counted string is created and its address is left in RESULT_ADDR.
;  823          0952  1 !
;  824          0953  1 ! Routine Value
;  825          0954  1 !
;  826          0955  1 !       A standard completion code.
;  827          0956  1 !--
;  828          0957  2     BEGIN
;  829          0958  2     MAP
;  830          0959  2         input_desc : REF dbg$stg_desc;          ! String descriptor for the
;  831          0960  2                                                 !       remaining (unparsed)
;  832          0961  2                                                 !       input string.
;  833          0962  2     LOCAL
;  834          0963  2         char,                                   ! Holds a single character thatr
;  835          0964  2                                                 !       is being copied from the
;  836          0965  2                                                 !       input stream to the
;  837          0966  2                                                 !       result string area.
;  838          0967  2         count,                                  ! Count of characters read
```

```
;  839          0968  2          input_ptr,                          ! Pointer to the current position
;  840          0969  2                                                     in the input stream.
;  841          0970  2          lahead_char,                        ! Holds next character in stream.
;  842          0971  2                                                     Used to look ahead one
;  843          0972  2                                                     character to determine
;  844          0973  2                                                     whether it is necessary
;  845          0974  2                                                     to undouble quotes.
;  846          0975  2          length,                             ! Holds the remaining length of the
;  847          0976  2                                                     input stream.
;  848          0977  2          output_ptr,                         ! Pointer to the current position
;  849          0978  2                                                     in the output character
;  850          0979  2                                                     string.
;  851          0980  2          result_str: REF VECTOR [,BYTE];     ! A pointer to the counted string
;  852          0981  2                                                     containing the search string.
;  853          0982  2
;  854          0983  2      ! We first allocate space for the result, so we can copy the string over
;  855          0984  2      ! as we read it character by character. (The call below may reserve
;  856          0985  2      ! more space than is needed, but doing things this way simplifies the
;  857          0986  2      ! algorithm. The alternative would be to loop through character by
;  858          0987  2      ! character, keeping a count, then reserve space, then loop through again
;  859          0988  2      ! to copy it over).
;  860          0989  2      !
;  861          0990  2      IF .PERM_FLAG
;  862          0991  2      THEN
;  863          0992  2          RESULT_STR = DBG$GET_MEMORY(((1 + .INPUT_DESC[DSC$W_LENGTH])/%UPVAL) + 1)
;  864          0993  2
;  865          0994  2      ELSE
;  866          0995  2          RESULT_STR = DBG$GET_TEMPMEM(((1 + .INPUT_DESC[DSC$W_LENGTH])/%UPVAL) + 1);
;  867          0996  2
;  868          0997  2
;  869          0998  2      ! Perform some initialization.
;  870          0999  2      !
;  871          1000  2      count = 0;
;  872          1001  2      result_str[0] = 0;
;  873          1002  2      input_ptr = ch$ptr (.input_desc [dsc$a_pointer]);
;  874          1003  2      output_ptr = ch$ptr (result_str[1]);
;  875          1004  2      length = .input_desc [dsc$w_length];
;  876          1005  2
;  877          1006  2      ! Read the first character. Each time we read a character we decrement
;  878          1007  2      ! the length variable since it represents remaining length.
;  879          1008  2      !
;  880          1009  2      char = ch$rchar_a (input_ptr);
;  881          1010  2      length = .length - 1;
;  882          1011  2
;  883          1012  2      ! Loop until we hit the delimiter
;  884          1013  2      !
;  885          1014  2      WHILE TRUE DO
;  886          1015  3          BEGIN
;  887          1016  3          ! First check for the delimiter
;  888          1017  3          !
;  889          1018  3          IF .char EQL .delimiter
;  890          1019  3          THEN
;  891          1020  3              ! If we see the delimiter, then check to see whether we are looking
;  892          1021  3              ! at a pair of quotes, E.g.,
;  893          1022  3              ! SEARCH "AB""CD"
;  894          1023  3              ! In this case we want to undouble the quotes and continue.
;  895          1024  3              !
```

```
;  896          1025  4            BEGIN
;  897          1026  4            lahead_char = ch$rchar (.input_ptr);
;  898          1027  4            IF .lahead_char EQL .char  AND .delimiter NEQ dbg$k_car_return
;  899          1028  4            THEN
;  900          1029  4                ! Undouble the quotes
;  901          1030  4                !
;  902          1031  5                BEGIN
;  903          1032  5                input_ptr = ch$plus (.input_ptr, 1);
;  904          1033  5                END
;  905          1034  4            ELSE
;  906          1035  4                ! Not a case of double quotes, so just exit the loop.
;  907          1036  4                EXITLOOP;
;  908          1037  3            END;
;  909          1038  3
;  910          1039  3        ! Translate lower case to upper case.
;  911          1040  3        !
;  912          1041  3        IF .char GEQ %C'a' AND .char LEQ %C'z' AND .uppercase_flag
;  913          1042  3        THEN
;  914          1043  3            char = .char - (%C'a'-%C'A');
;  915          1044  3
;  916          1045  3        ! Write the current character to the output buffer and
;  917          1046  3        ! get the next character.
;  918          1047  3        !
;  919          1048  3        ch$wchar_a (.char, output_ptr);
;  920          1049  3        count = .count + 1;
;  921          1050  3        if .count gtr 255 then signal (dbg$_strtoolong);
;  922          1051  3        result_str[0] = .result_str[0] + 1;
;  923          1052  3        char = ch$rchar_a (input_ptr);
;  924          1053  3
;  925          1054  3            ! Check for exhausted input
;  926          1055  3            !
;  927          1056  3            IF .length EQL 0
;  928          1057  3            THEN
;  929          1058  4                BEGIN
;  930          1059  4                ! If we reach the end of the input without seeing the delimiter
;  931          1060  4                ! character then this is an error
;  932          1061  4                !
;  933          1062  4                input_desc[dsc$w_length] = .input_desc[dsc$w_length] - 1;
;  934          1063  4                IF .delimiter EQL dbg$k_quote
;  935          1064  4                THEN ! this was signaled as (nodelims)
;  936          1065  4                    .message_vect = dbg$nmake_arg_vect (dbg$_MATQUOMIS)
;  937          1066  4                ELSE
;  938          1067  4                    IF .delimiter EQL dbg$k_dblquote
;  939          1068  4                    THEN    ! this was signaled as (nodelimd)
;  940          1069  4                        .message_vect = dbg$nmake_arg_vect (dbg$_MATQUOMIS)
;  941          1070  4                    ELSE
;  942          1071  4                        $DBG_ERROR('DBGNSEARC\DBG$NPARSE_SEARCH');
;  943          1072  4                RETURN sts$k_severe;
;  944          1073  3                END;
;  945          1074  3
;  946          1075  3            length = .length - 1;
;  947          1076  2            END;
;  948          1077  2
;  949          1078  2        ! Now back up so that we do not include trailing blanks
;  950          1079  2        !
;  951          1080  2        IF .delimiter EQL dbg$k_car_return
;  952          1081  2        THEN
```

```
;  953          1082  2          DECR i FROM .result_str[0] TO 1 DO
;  954          1083  3              BEGIN
;  955          1084  3              IF .result_str[.i] NEQ %C' '
;  956          1085  3              THEN
;  957          1086  3                  EXITLOOP;
;  958          1087  3              .result_str[0] = .result_str[0] - 1;
;  959          1088  2              END;
;  960          1089  2
;  961          1090  2      ! Update the command string descriptor so it points to beyond
;  962          1091  2      ! the end of the string just read.
;  963          1092  2      !
;  964          1093  2      input_desc [dsc$a_pointer] = .input_ptr;
5 ;  965          1094  2      input_desc [dsc$w_length] = .length;
;  966          1095  2
;  967          1096  2      ! The address of the counted string we have just read goes into result_addr.
;  968          1097  2      !
1 ;  969          1098  2      .result_addr = .result_str;
0 ;  970          1099  2
7 ;  971          1100  2      RETURN sts$k_success
;  972          1101  2
;  973          1102  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
2
24 47 42 44 5C 43 52 41 45 53 4E 47 42 44 1B  0004E P.AAM:    .ASCII  <27>\DBGNSEARC\<92>\DBG$NPARSE_SEARCH\      ;
       48 43 52 41 45 53 5F 45 53 52 41 50 4E  0005D                                                          ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0
4
5
6                                       0FFC 00000              .ENTRY  DBG$NACCEPT_STRING, Save R2,R3,R4,R5,R6,R7,-; 0911
9                                                                       R8,R9,R10,R11                               ;
9                  5B 00000000G   00    9E 00002              MOVAB   LIB$SIGNAL, R11                             ;
                   53       04    AC    D0 00009              MOVL    INPUT_DESC, R3                              ; 0992
                   16       10    AC    E9 0000D              BLBC    PERM_FLAG, 1$                               ; 0996
                   52             63    3C 00011              MOVZWL  (R3), R2                                    ; 0992
                   50       01    A2    9E 00014              MOVAB   1(R2), R0                                   ;
                   50             04    C6 00018              DIVL2   #4, R0                                      ;
1                  50       01    A0    9F 0001B              PUSHAB  1(R0)                                       ;
2 00000000G        00             01    FB 0001E              CALLS   #1, DBG$GET_MEMORY                          ;
3                                 14    11 00025              BRB     2$                                          ;
9                  52             63    3C 00027 1$:          MOVZWL  (R3), R2                                    ; 0995
6                  50       01    A2    9E 0002A              MOVAB   1(R2), R0                                   ;
                   50             04    C6 0002E              DIVL2   #4, R0                                      ;
                   50       01    A0    9F 00031              PUSHAB  1(R0)                                       ;
                00000000G  00             01    FB 00034      CALLS   #1, DBG$GET_TEMPMEM                         ;
                   54             50    D0 0003B 2$:          MOVL    R0, RESULT_STR                              ;
                                  59    D4 0003E              CLRL    COUNT                                       ; 1000
8                                 64    94 00040              CLRB    (RESULT_STR)                                ; 1001
9                  55       04    A3    D0 00042              MOVL    4(R3), INPUT_PTR                            ; 1002
0                  58       01    A4    9E 00046              MOVAB   1(R4), OUTPUT_PTR                           ; 1003
9                  57             52    D0 0004A              MOVL    R2, LENGTH                                  ; 1004
3                  56             85    9A 0004D              MOVZBL  (INPUT_PTR)+, CHAR                          ; 1009
                                  57    D7 00050              DECL    LENGTH                                      ; 1010
```

```
                 52       0C    AC  D0 00052       MOVL    DELIMITER, R2                      ; 1018
                 52             56  D1 00056 3$:   CMPL    CHAR, R2                           ;
                                0F  12 00059       BNEQ    4$                                 ;
                 5A             65  9A 0005B       MOVZBL  (INPUT_PTR), LAHEAD_CHAR           ; 1026
                 56             5A  D1 0005E       CMPL    LAHEAD_CHAR, CHAR                  ; 1027
                                79  12 00061       BNEQ    11$                                ;
                 0D             52  D1 00063       CMPL    R2, #13                            ;
                                74  13 00066       BEQL    11$                                ;
                                55  D6 00068       INCL    INPUT_PTR                          ; 1032
       00000061  8F             56  D1 0006A 4$:   CMPL    CHAR, #97                          ; 1041
                                10  19 00071       BLSS    5$                                 ;
       0000007A  8F             56  D1 00073       CMPL    CHAR, #122                         ;
                                07  14 0007A       BGTR    5$                                 ;
                 03       18    AC  E9 0007C       BLBC    UPPERCASE_FL?G, 5$                 ;
                 56             20  C2 00080       SUBL2   #32, CHAR                          ; 1043
                 88             56  90 00083 5$:   MOVB    CHAR, (OUTP? _PTR)+                ; 1048
                                59  D6 00086       INCL    COUNT                              ; 1049
       000000FF  8F             59  D1 00088       CMPL    COUNT, #255                        ; 1050
                                09  15 0008F       BLEQ    6$                                 ;
                    00028160    8F  DD 00091       PUSHL   #164192                            ;
                 6B             01  FB 00097       CALLS   #1, LIB$SIGNAL                     ;
                                64  96 0009A 6$:   INCB    (RESULT_STR)                       ; 1051
                 56             85  9A 0009C       MOVZBL  (INPUT_PTR)+, CHAR                 ; 1052
                                57  D5 0009F       TSTL    LENGTH                             ; 1056
                                34  12 000A1       BNEQ    10$                                ;
                                63  B7 000A3       DECW    (R3)                               ; 1062
                 27             52  D1 000A5       CMPL    R2, #39                            ; 1063
                                05  13 000A8       BEQL    7$                                 ;
                 22             52  D1 000AA       CMPL    R2, #34                            ; 1067
                                13  12 000AD       BNEQ    8$                                 ;
                    00028E30    8F  DD 000AF 7$:   PUSHL   #167472                            ; 1069
      00000000G  00             01  FB 000B5       CALLS   #1, DBG$NMAKE_ARG_VECT             ;
             14  BC             50  D0 000BC       MOVL    R0, @MESSAGE_VECT                  ;
                                11  11 000C0       BRB     9$                                 ;
                    00000000'   EF  9F 000C2 8$:   PUSHAB  P.AAM                              ; 1071
                                01  DD 000C8       PUSHL   #1                                 ;
                    00028362    8F  DD 000CA       PUSHL   #164706                            ;
                 6B             03  FB 000D0       CALLS   #3, LIB$SIGNAL                     ;
                 50             04  D0 000D3 9$:   MOVL    #4, R0                             ; 1072
                                    04 000D6       RET                                        ;
                                57  D7 000D7 10$:  DECL    LENGTH                             ; 1075
                              FF7A  31 000D9       BRW     3$                                 ; 1014
                 0D             52  D1 000DC 11$:  CMPL    R2, #13                            ; 1080
                                16  12 000DF       BNEQ    14$                                ;
                 51             64  9A 000E1       MOVZBL  (RESULT_STR), R1                   ; 1082
                 50       01    A1  9E 000E4       MOVAB   1(R1), I                           ;
                                0A  11 000E8       BRB     13$                                ;
                 20           6044  91 000EA 12$:  CMPB    (I)[RESULT_STR], #32               ; 1084
                                07  12 000EE       BNEQ    14$                                ;
                 61       FF    A1  9E 000F0       MOVAB   -1(R1), (R1)                       ; 1087
                 F3             50  F5 000F4 13$:  SOBGTR  I, 12$                             ; 1082
             04  A3             55  D0 000F7 14$:  MOVL    INPUT_PTR, 4(R3)                   ; 1093
                 63             57  B0 000FB       MOVW    LENGTH, (R3)                       ; 1094
             08  BC             54  D0 000FE       MOVL    RESULT_STR, @RESULT_ADDR           ; 1098
                 50             01  D0 00102       MOVL    #1, R0                             ; 1100
                                    04 00105       RET                                        ; 1102
```

; Routine Size: 262 bytes,   Routine Base: DBG$CODE + 048B

```
;  975          1103  1 GLOBAL ROUTINE dbg$parse_search (parse_stg_desc) =
;  976          1104  1 !+
;  977          1105  1 ! Functional Description
;  978          1106  1 !
;  979          1107  1 !       This routine provides an interface from the old language parsers to
;  980          1108  1 !       the new debugger parse network for SEARCH. It is passed a string
;  981          1109  1 !       descriptor for the remainder of the input line.
;  982          1110  1 !       It calls DBG$NPARSE_SEARCH to construct
;  983          1111  1 !       a command execution network, and returns a pointer to the verb node.
;  984          1112  1 !
;  985          1113  1 ! Inputs
;  986          1114  1 !
;  987          1115  1 !       parse_stg_desc  -       A string descriptor for the remainder of the
;  988          1116  1 !                               input line.
;  989          1117  1 !
;  990          1118  1 ! Outputs
;  991          1119  1 !
;  992          1120  1 !       A command execution network is constructed,
;  993          1121  1 !       consisting of a verb node for the SEARCH
;  994          1122  1 !       verb, 0-2 adverb nodes for the overrides /IDENT, /ALL, etc. , and
;  995          1123  1 !       two noun nodes. A pointer to this network is returned.
;  996          1124  1 !--
;  997          1125  2     BEGIN
;  998          1126  2     MAP
;  999          1127  2         parse_stg_desc : REF BLOCK [,BYTE];
; 1000          1128  2     LOCAL
; 1001          1129  2         char,
; 1002          1130  2         dummy_mess_vect: REF VECTOR,    ! Address for message vector returned
; 1003          1131  2                                         !    from DBG$NPARSE_SEARCH
; 1004          1132  2         len,                            ! Length of command line
; 1005          1133  2         parse_stg_ptr,                  ! Pointer into command line
; 1006          1134  2         stg : REF VECTOR [,BYTE],       ! Pointer to a new copy of the
; 1007          1135  2                                         !    command line
; 1008          1136  2         verb_node;                      ! Pointer to the head of the command
; 1009          1137  2                                         !    execution tree for SEARCH
; 1010          1138  2
; 1011          1139  2     ! Call the 'new style' parse network for the search
; 1012          1140  2     ! command. This builds a command execution network.
; 1013          1141  2     ! We return a pointer to the verb node.
; 1014          1142  2     !
; 1015          1143  2     ! First allocate space for the verb node.
; 1016          1144  2     !
; 1017          1145  2     VERB_NODE = DBG$GET_TEMPMEM(DBG$K_VERB_NODE_SIZE);
; 1018          1146  2
; 1019          1147  2
; 1020          1148  2     ! Then stuff a carriage return character at the end
; 1021          1149  2     ! of the input line since this is what the new style
; 1022          1150  2     ! parser expects to see. Also, translate the line to
; 1023          1151  2     ! upper case (the new debugger does this; the old does not)
; 1024          1152  2     !
; 1025          1153  2     len = .parse_stg_desc[dsc$w_length];
; 1026          1154  2     STG = DBG$GET_TEMPMEM(1 + (1 + .LEN)/%UPVAL);
; 1027          1155  2     parse_stg_ptr = ch$ptr(.parse_stg_desc[dsc$a_pointer]);
; 1028          1156  2     INCR I FROM 0 TO .len-1 DO
; 1029          1157  3         BEGIN
; 1030          1158  3         char = ch$rchar_a(parse_stg_ptr);
; 1031          1159  3         IF .char GEQ %C'a' AND .char LEQ %C'z'
```

```
; 1032      1160 3          THEN
; 1033      1161 4              stg[.j] = .char - (%C'a'-%C'A')
; 1034      1162 3          ELSE
; 1035      1163 3              stg[.j] = .char;
; 1036      1164 2          END;
; 1037      1165 2      stg[.len] = dbg$k_car_return;
; 1038      1166 2      parse_stg_desc[dsc$a_pointer] = .stg;
; 1039      1167 2      parse_stg_desc[dsc$w_length] =
; 1040      1168 2          .parse_stg_desc[dsc$w_length] + 1;
; 1041      1169 2
; 1042      1170 2      ! Now call the parser on the remainder of the input line
; 1043      1171 2      !
; 1044      1172 2      IF NOT dbg$nparse_search (.parse_stg_desc,
; 1045      1173 2          .verb_node, dummy_mess_vect)
; 1046      1174 2      THEN
; 1047      1175 2          ! If the above routine does not return success, then we signal
; 1048      1176 2          ! an error using the error message vector that we got back.
; 1049      1177 2          !
; 1050      1178 3          BEGIN
; 1051      1179 3          EXTERNAL ROUTINE
; 1052      1180 3              LIB$SIGNAL : ADDRESSING_MODE(GENERAL);
; 1053      1181 3          BUILTIN
; 1054      1182 3              CALLG;
; 1055      1183 3          CALLG (.dummy_mess_vect, lib$signal);
; 1056      1184 2          END;
; 1057      1185 2
; 1058      1186 2      ! Restore pointer field of PARSE_STG_DESC since this can be wiped out
; 1059      1187 2      ! during new style parsing.
; 1060      1188 2      !
; 1061      1189 2      IF .parse_stg_desc[dsc$a_pointer] EQL 0
; 1062      1190 2      THEN
; 1063      1191 2          parse_stg_desc[dsc$a_pointer] = .stg+.len;
; 1064      1192 2
; 1065      1193 2      ! Finally, return a pointer to the verb node.
; 1066      1194 2      !
; 1067      1195 2      RETURN .verb_node
; 1068      1196 2
; 1069      1197 1      END; ! dbg$parse_search


                                          .EXTRN  LIB$SIGNAL

                      00FC 00000          .ENTRY  DBG$PARSE_SEARCH, Save R2,R3,R4,R5,R6,R7    ; 1103
57 00000000G     00    9E 00002           MOVAB   DBG$GET_TEMPMEM, R7                        ;
5E               04    C2 00009           SUBL2   #4, SP                                     ;
                 03    DD 0000C           PUSHL   #3                                         ; 1145
67               01    FB 0000E           CALLS   #1, DBG$GET_TEMPMEM                        ;
56               50    D0 00011           MOVL    R0, VERB_NODE                              ;
54         04    AC    D0 00014           MOVL    PARSE_STG_DESC, R4                         ; 1153
52               64    3C 00018           MOVZWL  (R4), LEN                                  ;
50         01    A2    9E 0001B           MOVAB   1(R2), R0                                  ; 1154
50               04    C6 0001F           DIVL2   #4, R0                                     ;
           01    A0    9F 00022           PUSHAB  1(R0)                                      ;
67               01    FB 00025           CALLS   #1, DBG$GET_TEMPMEM                        ;
53               50    D0 00028           MOVL    R0, STG                                    ;
55         04    A4    D0 0002B           MOVL    4(R4), PARSE_STG_PTR                       ; 1155
```

```
                         50         01  CE 0002F        MNEGL   #1, J                                ; 1161
                                    20  11 00032        BRB     3$                                   ;
                         51         85  9A 00034 1$:    MOVZBL  (PARSE_STG_PTR)+, CHAR               ; 1158
            00000061     8F         51  D1 00037        CMPL    CHAR, #97                            ; 1159
                                    10  19 0003E        BLSS    2$                                   ;
            0000007A     8F         51  D1 00040        CMPL    CHAR, #122                           ;
                                    07  14 00047        BGTR    2$                                   ;
      6043               51         20  83 00049        SUBB3   #32, CHAR, (J)[STG]                  ; 1161
                                    04  11 0004E        BRB     3$                                   ;
                       6043         51  90 00050 2$:    MOVB    CHAR, (J)[STG]                       ; 1163
         DC              50         52  F2 00054 3$:    AOBLSS  LEN, J, 1$                           ; 1156
                       6243         0D  90 00058        MOVB    #13, (LEN)[STG]                      ; 1165
                 04      A4         53  D0 0005C        MOVL    STG, 4(R4)                           ; 1166
                                    64  B6 00060        INCW    (R4)                                 ; 1168
                          4050      8F  BB 00062        PUSHR   #^M<R4,R6,SP>                        ; 1172
               FA04      CF         03  FB 00066        CALLS   #3, DBG$NPARSE_SEARCH                ;
                         08         50  E8 0006B        BLBS    R0, 4$                               ;
            00000000G    00    00   BE  FA 0006E        CALLG   @DUMMY_MESS_VECT, LIB$SIGNAL         ; 1183
                         50    04   AC  D0 00076 4$:    MOVL    PARSE_STG_DESC, R0                   ; 1189
                               04   A0  D5 0007A        TSTL    4(R0)                                ;
                                    05  12 0007D        BNEQ    5$                                   ;
   04     A0             53         52  C1 0007F        ADDL3   LEN, STG, 4(R0)                      ; 1191
                         50         56  D0 00084 5$:    MOVL    VERB_NODE, R0                        ; 1195
                                        04 00087        RET                                          ; 1197

; Routine Size:  136 bytes,    Routine Base:  DBG$CODE + 0591


; 1070          1198  1 END
; 1071          1199  0 ELUDOM



                                                        .EXTRN  LIB$SIGNAL

;                           PSECT SUMMARY
;
;       Name                        Bytes                       Attributes
;
;    DBG$PLIT                         106  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;    DBG$OWN                           81  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;    DBG$CODE                        1561  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        4         0         1000        00:01.9
;    _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1    32        0         0            7        00:00.2
;    _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1    1545       35         2           97        00:02.0
;    _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                            418        0         0           31        00:00.4
```

```
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1        386        7        1        22          00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1        150        3        2        12          00:00.3



;                                    COMMAND QUALIFIERS

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

; Size:         1561 code + 187 data bytes
; Run Time:        00:33.3
; Elapsed Time:    01:26.4
; Lines/CPU Min:    2163
; Lexemes/CPU-Min: 11482
; Memory Used:  298 pages
; Compilation Complete
```

DBGNSET
LIS

DBGNSDATA
LIS

DBGNSEARC
LIS