```
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     UUUUUUUUUUUUUUUU    GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
```

FILEID**DBGIFTHEN

```
DDDDDDDD    BBBBBBBB      GGGGGGGG    IIIIII    FFFFFFFFFF  TTTTTTTTTT  HH      HH  EEEEEEEEEE  NN      NN
DDDDDDDD    BBBBBBBB      GGGGGGGG    IIIIII    FFFFFFFFFF  TTTTTTTTTT  HH      HH  EEEEEEEEEE  NN      NN
DD      DD  BB      BB  GG              II      FF              TT      HH      HH  EE          NN      NN
DD      DD  BB      BB  GG              II      FF              TT      HH      HH  EE          NN      NN
DD      DD  BB      BB  GG              II      FF              TT      HH      HH  EE          NNNN    NN
DD      DD  BB      BB  GG              II      FF              TT      HH      HH  EE          NNNN    NN
DD      DD  BBBBBBBB    GG              II      FFFFFFFF        TT      HHHHHHHHHH  EEEEEEEE    NN  NN  NN
DD      DD  BBBBBBBB    GG              II      FFFFFFFF        TT      HHHHHHHHHH  EEEEEEEE    NN  NN  NN
DD      DD  BB      BB  GG    GGGGGG    II      FF              TT      HH      HH  EE          NN    NNNN
DD      DD  BB      BB  GG    GGGGGG    II      FF              TT      HH      HH  EE          NN    NNNN
DD      DD  BB      BB  GG        GG    II      FF              TT      HH      HH  EE          NN      NN      ....
DD      DD  BB      BB  GG        GG    II      FF              TT      HH      HH  EE          NN      NN      ....
DDDDDDDD    BBBBBBBB      GGGGGG      IIIIII    FF              TT      HH      HH  EEEEEEEEEE  NN      NN      ....
DDDDDDDD    BBBBBBBB      GGGGGG      IIIIII    FF              TT      HH      HH  EEEEEEEEEE  NN      NN      ....

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 DBGIFTHEN (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 !++
;    31          0031  1 ! FACILITY:
;    32          0032  1 !
;    33          0033  1 !       DEBUG
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This module contains the parse and execution networks for the
;    38          0038  1 !       DEBUG control structures: IF...THEN...ELSE, WHILE...DO,
;    39          0039  1 !       FOR loops, and REPEAT...DO
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:
;    42          0042  1 !
;    43          0043  1 !       VAX/VMS
;    44          0044  1 !
;    45          0045  1 ! AUTHOR:
;    46          0046  1 !
;    47          0047  1 !       Richard Title
;    48          0048  1 !
;    49          0049  1 ! CREATION DATE:
;    50          0050  1 !
;    51          0051  1 !       1-10-82
;    52          0052  1 !
;    53          0053  1 ! VERSION:
;    54          0054  1 !
;    55          0055  1 !       V03.0-001
;    56          0056  1 !
;    57          0057  1 ! MODIFIED BY:
```

```
;    58          0058  1 !
;    59          0059  1 !
;    60          0060  1 ! REVISION HISTORY:
;    61          0061  1 !
;    62          0062  1 !--
```

```
;    64          0063  1 !
;    65          0064  1 ! TABLE OF CONTENTS:
;    66          0065  1 !
;    67          0066  1
;    68          0067  1 FORWARD ROUTINE
;    69          0068  1         DBG$NPARSE_IF,                  ! Parse network
;    70          0069  1         DBG$NEXECUTE_IF,                ! Execution network
;    71          0070  1         DBG$NPARSE_WHILE,               ! Parse network
;    72          0071  1         DBG$NEXECUTE_WHILE,             ! Execution network
;    73          0072  1         DBG$NPARSE_FOR,                 ! Parse network for FOR
;    74          0073  1         DBG$NEXECUTE_FOR,               ! Execution network for FOR
;    75          0074  1         DBG$NPARSE_REPEAT,              ! Parse network
;    76          0075  1         DBG$NEXECUTE_REPEAT;            ! Execution network
;    77          0076  1
;    78          0077  1 ! REQUIRE FILES:
;    79          0078  1 !
;    80          0079  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    81          0213  1 LIBRARY 'LIB$:DBGGEN.L32';
;    82          0214  1
;    83          0215  1 EXTERNAL
;    84          0216  1     dbg$gb_language: BYTE,              ! Current language setting
;    85          0217  1     dbg$gb_radix: VECTOR[3, BYTE],      ! Radix settings
;    86          0218  1     dbg$gb_take_cmd: BYTE,              ! Flag that controls command taking
;    87          0219  1     dbg$gl_cishead: REF cis$link;       ! Head of command input stream
;    88          0220  1
;    89          0221  1 EXTERNAL ROUTINE
;    90          0222  1     dbg$def_sym_add,                    ! Add a defined symbol
;    91          0223  1     dbg$get_memory,                     ! Allocate permanent memory
;    92          0224  1     dbg$get_tempmem,                    ! Allocates space
;    93          0225  1     dbg$ncis_add,                       ! Add a link to the command input stream
;    94          0226  1     dbg$nget_symid,                     ! Obtain a symid list
;    95          0227  1     dbg$nmake_arg_vect,                 ! Constructs error messages
;    96          0228  1     dbg$nmatch,                         ! Tries to match the next token
;    97          0229  1     dbg$nnext_word,                     ! Gets next word from input
;    98          0230  1     dbg$nparse_expression,              ! Language specific expression interpreter
;    99          0231  1     dbg$nread_name,                     ! Pick up a name
;   100          0232  1     dbg$nsave_break_buffer: NOVALUE,    ! Saves the action clause in a buffer
;   101          0233  1     dbg$nsyntax_error,                  ! Reports a syntax error
;   102          0234  1     dbg$ntype_conv,                     ! Language specific type converter
;   103          0235  1     dbg$rel_memory: NOVALUE,            ! Releases memory from DEBUG memory pool
;   104          0236  1     dbg$sta_lock_symid: NOVALUE;        ! Lock a symid list
;   105          0237  1
;   106        M 0238  1 MACRO report_error =
;   107        M 0239  1         BEGIN
;   108        M 0240  1         .message_vect =
;   109        M 0241  1             (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;   110        M 0242  1              THEN
;   111        M 0243  1                  dbg$nmake_arg_vect(dbg$_needmore)
;   112        M 0244  1              ELSE
;   113        M 0245  1                  dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;   114        M 0246  1         RETURN sts$k_severe;
;   115          0247  1         END%;
```

```
;  117     0248  1 GLOBAL ROUTINE dbg$nparse_if(input_desc, verb_node, message_vect) =
;  118     0249  1 !
;  119     0250  1 ! Functional Description
;  120     0251  1 !
;  121     0252  1 !       ATN parse network for the IF verb.
;  122     0253  1 !       This routine takes a verb node for the IF verb, and a string
;  123     0254  1 !       descriptor for the remaining (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 |[-->--| noun node |]
;  128     0259  1 !        -----------       -----------       -----------       -----------
;  129     0260  1 !
;  130     0261  1 !       The first noun node points to a value descriptor for the IF clause.
;  131     0262  1 !       The second noun node points to a counted string with the THEN clause.
;  132     0263  1 !       The third noun node, which may be absent, points to a counted string
;  133     0264  1 !       with the ELSE clause.
;  134     0265  1 !
;  135     0266  1 ! Formal Parameters
;  136     0267  1 !
;  137     0268  1 !       input_desc      - A longword containing the address of the
;  138     0269  1 !                         command input descriptor.
;  139     0270  1 !       verb_node       - A longword containing the address of the verb node.
;  140     0271  1 !       message_vect    - The address of a longword to contain the address
;  141     0272  1 !                         of a standard message argument vector.
;  142     0273  1 !
;  143     0274  1 ! Implicit Inputs
;  144     0275  1 !
;  145     0276  1 !       none
;  146     0277  1 !
;  147     0278  1 ! Implicit Outputs
;  148     0279  1 !
;  149     0280  1 !       On success, the command execution tree is constructed.
;  150     0281  1 !       On failure, a message argument vector is constructed or obtained.
;  151     0282  1 !
;  152     0283  1 ! Routine value
;  153     0284  1 !
;  154     0285  1 !       sts$k_success (1)       - Success. Command execution tree constructed.
;  155     0286  1 !       sts$k_severe  (4)       - Failure. Error encountered. Message argument
;  156     0287  1 !                                 constructed and returned.
;  157     0288  1 !
;  158     0289  1 ! Side Effects
;  159     0290  1 !
;  160     0291  1 !       Permanent storage is allocated for the string holding the THEN
;  161     0292  1 !       clause; this is released in DBG$NEXECUTE_IF after execution
;  162     0293  1 !       of the THEN clause.
;  163     0294  1 !
;  164     0295  1
;  165     0296  2 BEGIN
;  166     0297  2
;  167     0298  2 MAP
;  168     0299  2     input_desc  : REF dbg$stg_desc,
;  169     0300  2     verb_node   : REF dbg$verb_node;
;  170     0301  2
;  171     0302  2 BIND
;  172     0303  2     dbg$cs_cr           = UPLIT BYTE (1, dbg$k_car_return),
;  173     0304  2     dbg$cs_left_paren   = UPLIT BYTE (1, dbg$k_left_parenthesis),
```

```
;  174     0305  2      dbg$cs_else          = UPLIT BYTE (4, 'ELSE'),
;  175     0306  2      dbg$cs_then          = UPLIT BYTE (4, 'THEN');
;  176     0307  2
;  177     0308  2  LOCAL
;  178     0309  2      link,                                  ! Temporary to links in the command
;  179     0310  2                                             !     execution tree.
;  180     0311  2      noun_node : REF dbg$noun_node,         ! A node in the command execution tree.
;  181     0312  2      radix,                                 ! Holds the current radix setting.
;  182     0313  2      status;                                ! Holds return status from subroutine
;  183     0314  2                                             !     calls.
;  184     0315  2
;  185     0316  2
;  186     0317  2
;  187     0318  2      ! Create and link a noun node
;  188     0319  2      !
;  189     0320  2      noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  190     0321  2      verb_node[dbg$l_verb_object_ptr] = .noun_node;
;  191     0322  2
;  192     0323  2
;  193     0324  2      ! Determine the current radix.
;  194     0325  2      !
;  195     0326  2      radix  = .dbg$gb_radix[dbg$b_radix_input];
;  196     0327  2
;  197     0328  2
;  198     0329  2      ! Obtain a value descriptor for the condition. The first noun node
;  199     0330  2      ! points to this descriptor.
;  200     0331  2      !
;  201     0332  2      STATUS = DBG$NPARSE_EXPRESSION (.INPUT_DESC, .RADIX,
;  202     0333  2                                      NOUN_NODE [DBG$L_NOUN_VALUE],
;  203     0334  2                                      TOKEN$K_TERM_THEN, .MESSAGE_VECT);
;  204     0335  2
;  205     0336  2
;  206     0337  2      ! The return status should be "warning", meaning that an expression
;  207     0338  2      ! was parsed and further input reamins. If an expression was parsed
;  208     0339  2      ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
;  209     0340  2      ! In this context, it is an error since "IF condition" by itself
;  210     0341  2      ! is an error.
;  211     0342  2      !
;  212     0343  2      IF .status EQL sts$k_success THEN SIGNAL(DBG$_NEEDMORE);
;  213     0344  2
;  214     0345  2
;  215     0346  2      ! Severe status is also an error.
;  216     0347  2      !
;  217     0348  2      IF .status EQL sts$k_severe
;  218     0349  2      THEN
;  219     0350  2          RETURN sts$k_severe;
;  220     0351  2
;  221     0352  2
;  222     0353  2      ! Eat the THEN
;  223     0354  2      !
;  224     0355  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_then, 1)
;  225     0356  2      THEN
;  226     0357  3          BEGIN
;  227     0358  3          .message_vect =
;  228     0359  4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  229     0360  4               THEN
;  230     0361  4                  dbg$nmake_arg_vect (dbg$_needmore)
```

```
;   231          0362  4              ELSE
;   232          0363  3                 dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;   233          0364  3
;   234          0365  3          RETURN sts$k_severe;
;   235          0366  2          END;
;   236          0367  2
;   237          0368  2
;   238          0369  2      ! Allocate and link a noun node for the THEN clause.
;   239          0370  2      !
;   240          0371  2      link = noun_node [dbg$l_noun_link];
;   241          0372  2      noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;   242          0373  2      .link = .noun_node;
;   243          0374  2
;   244          0375  2      ! Eat the left parenthesis which we require be present.
;   245          0376  2      !
;   246          0377  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;   247          0378  2      THEN
;   248          0379  3          BEGIN
;   249          0380  3          .message_vect =
;   250          0381  4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;   251          0382  4               THEN
;   252          0383  4                  dbg$nmake_arg_vect (dbg$_needmore)
;   253          0384  4               ELSE
;   254          0385  5                  BEGIN
;   255          0386  5                  SIGNAL(dbg$_needparen);
;   256          0387  5                  dbg$nsyntax_error (dbg$nnext_word (.input_desc))
;   257          0388  3                  END);
;   258          0389  3          RETURN sts$k_severe;
;   259          0390  2          END;
;   260          0391  2
;   261          0392  2      ! Put a pointer to the counted string representing the THEN
;   262          0393  2      ! clause into the second noun node. (Note - the counted string
;   263          0394  2      ! is constructed out of "permanent" memory which is released
;   264          0395  2      ! in DBG$NEXECUTE_IF).
;   265          0396  2      ! (The third argument indicates that this routine is not being
;   266          0397  2      ! called during a SET BREAK DO (the behavior is slightly different
;   267          0398  2      ! in that case.))
;   268          0399  2      !
;   269          0400  2      dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
;   270          0401  2
;   271          0402  2      ! If we have reached the end of the line, return success (no else
;   272          0403  2      ! clause is present).
;   273          0404  2      !
;   274          0405  2      IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;   275          0406  2         OR .input_desc [dsc$w_length] EQL 0
;   276          0407  2      THEN
;   277          0408  2          RETURN sts$k_success;
;   278          0409  2
;   279          0410  2      ! Look for ELSE clause.
;   280          0411  2      !
;   281          0412  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_else, 1)
;   282          0413  2      THEN
;   283          0414  3          BEGIN
;   284          0415  3          .message_vect = dbg$nsyntax_error (dbg$nnext_word (.input_desc));
;   285          0416  3          RETURN sts$k_severe;
;   286          0417  2          END;
;   287          0418  2
```

```
;  288          0419  2     ! Allocate and link a noun node for the ELSE clause.
;  289          0420  2     !
;  290          0421  2     link = noun_node [dbg$l_noun_link];
;  291          0422  2     noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  292          0423  2     .link = .noun_node;
;  293          0424  2
;  294          0425  2     ! Eat the left parenthesis which we require be present.
;  295          0426  2     !
;  296          0427  2     IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  297          0428  2     THEN
;  298          0429  3         BEGIN
;  299          0430  3         .message_vect =
;  300          0431  4             (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  301          0432  4              THEN
;  302          0433  4                 dbg$nmake_arg_vect (dbg$_needmore)
;  303          0434  4              ELSE
;  304          0435  5                 BEGIN
;  305          0436  5                 SIGNAL(dbg$_needparen);
;  306          0437  5                 dbg$nsyntax_error (dbg$nnext_word (.input_desc))
;  307          0438  3                 END);
;  308          0439  3         RETURN sts$k_severe;
;  309          0440  2         END;
;  310          0441  2
;  311          0442  2     ! Put a pointer to the counted string representing the ELSE
;  312          0443  2     ! clause into the third noun node. (Note - the counted string
;  313          0444  2     ! is constructed out of "permanent" memory which is released
;  314          0445  2     ! in DBG$NEXECUTE_IF).
;  315          0446  2     !
;  316          0447  2     dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
;  317          0448  2
;  318          0449  2     ! Return success.
;  319          0450  2     !
;  320          0451  2     RETURN sts$k_success;
;  321          0452  2
;  322          0453  1     END;


                                                .TITLE  DBGIFTHEN
                                                .IDENT  \V04-000\

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

                                    0D  01  00000 P.AAA:  .BYTE   1, 13                                    ;
                                    28  01  00002 P.AAB:  .BYTE   1, 40                                    ;
                                        04  00004 P.AAC:  .BYTE   4                                        ;
                                45  53  4C  45  00005     .ASCII  \ELSE\                                   ;
                                        04  00009 P.AAD:  .BYTE   4                                        ;
                                4E  45  48  54  0000A     .ASCII  \THEN\                                   ;

                                                DBG$CS_CR=          P.AAA
                                                DBG$CS_LEFT_PAREN=  P.AAB
                                                DBG$CS_ELSE=        P.AAC
                                                DBG$CS_THEN=        P.AAD
                                                .EXTRN  DBG$GB_LANGUAGE
                                                .EXTRN  DBG$GB_RADIX, DBG$GB_TAKE_CMD
                                                .EXTRN  DBG$GL_CISHEAD, DBG$DEF_SYM_ADD
                                                .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
```

```
                                        .EXTRN  DBG$NCIS_ADD, DBG$NGET_SYMID
                                        .EXTRN  DBG$NMAKE_ARG_VECT
                                        .EXTRN  DBG$NMATCH, DBG$NNEXT_WORD
                                        .EXTRN  DBG$NPARSE_EXPRESSION
                                        .EXTRN  DBG$NREAD_NAME, DBG$NSAVE_BREAK_BUFFER
                                        .EXTRN  DBG$NSYNTAX_ERROR
                                        .EXTRN  DBG$NTYPE_CONV, DBG$REL_MEMORY
                                        .EXTRN  DBG$STA_LOCK_SYMID

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

                          07FC 00000    .ENTRY  DBG$NPARSE_IF, Save R2,R3,R4,R5,R6,R7,R8,-      ; 0248
                                                R9,R10
                5A 00000000G   00  9E 00002    MOVAB   DBG$NSAVE_BREAK_BUFFER, R10             ;
                59 00000000G   00  9E 00009    MOVAB   LIB$SIGNAL, R9                          ;
                58 00000000G   00  9E 00010    MOVAB   DBG$GET_TEMPMEM, R8                     ;
                57 00000000G   00  9E 00017    MOVAB   DBG$NMATCH, R7                          ;
                56 00000000'   EF  9E 0001E    MOVAB   DBG$CS_CR, R6                           ;
                               04  DD 00025    PUSHL   #4                                      ; 0320
                68             01  FB 00027    CALLS   #1, DBG$GET_TEMPMEM                     ;
                55             50  D0 0002A    MOVL    R0, NOUN_NODE                           ;
                50       08    AC  D0 0002D    MOVL    VERB_NODE, R0                           ; 0321
             08 A0             55  D0 00031    MOVL    NOUN_NODE, 8(R0)                        ;
                50 00000000G   00  9A 00035    MOVZBL  DBG$GB_RADIX, RADIX                     ; 0326
                53       0C    AC  D0 0003C    MOVL    MESSAGE_VECT, R3                        ; 0334
                               53  DD 00040    PUSHL   R3                                      ;
                               06  DD 00042    PUSHL   #6                                      ; 0333
                               21  BB 00044    PUSHR   #^M<R0,R5>                              ;
                52       04    AC  D0 00046    MOVL    INPUT_DESC, R2                          ; 0332
                               52  DD 0004A    PUSHL   R2                                      ; 0333
00000000G       00             05  FB 0004C    CALLS   #5, DBG$NPARSE_EXPRESSION               ;
                54             50  D0 00053    MOVL    R0, STATUS                              ;
                01             54  D1 00056    CMPL    STATUS, #1                              ; 0343
                               09  12 00059    BNEQ    1$                                      ;
                   000280D0    8F  DD 0005B    PUSHL   #164048                                 ;
                69             01  FB 00061    CALLS   #1, LIB$SIGNAL                          ;
                04             54  D1 00064 1$: CMPL    STATUS, #4                              ; 0348
                               03  12 00067    BNEQ    2$                                      ;
                             00AE  31 00069    BRW     10$                                     ;
                               01  DD 0006C 2$: PUSHL   #1                                      ; 0355
                          09   A6  9F 0006E    PUSHAB  DBG$CS_THEN                             ;
                               52  DD 00071    PUSHL   R2                                      ;
                67             03  FB 00073    CALLS   #3, DBG$NMATCH                          ;
                0E             50  E8 00076    BLBS    R0, 3$                                  ;
                               01  DD 00079    PUSHL   #1                                      ; 0359
                        0044   8F  BB 0007B    PUSHR   #^M<R2,R6>                              ;
                67             03  FB 0007F    CALLS   #3, DBG$NMATCH                          ;
                3D             50  E9 00082    BLBC    R0, 4$                                  ;
                               66  11 00085    BRB     6$                                      ; 0361
                54       08    A5  9E 00087 3$: MOVAB   8(R5), LINK                            ; 0371
                               04  DD 0008B    PUSHL   #4                                      ; 0372
                68             01  FB 0008D    CALLS   #1, DBG$GET_TEMPMEM                     ;
                55             50  D0 00090    MOVL    R0, NOUN_NODE                           ;
                64             55  D0 00093    MOVL    NOUN_NODE, (LINK)                       ; 0373
                               01  DD 00096    PUSHL   #1                                      ; 0377
                          02   A6  9F 00098    PUSHAB  DBG$CS_LEFT_PAREN                       ;
                               52  DD 0009B    PUSHL   R2                                      ;
```

```
                67              03 FB 0009D             CALLS   #3, DBG$NMATCH                          ;
                3E              50 E9 000A0             BLBC    R0, 5$                                  ;
                                24 BB 000A3             PUSHR   #^M<R2,R5>                              ; 0400
                6A              02 FB 000A5             CALLS   #2, DBG$NSAVE_BREAK_BUFFER              ;
                                01 DD 000A8             PUSHL   #1                                      ; 0405
                        0044    8F BB 000AA             PUSHR   #^M<R2,R6>                              ;
                67              03 FB 000AE             CALLS   #3, DBG$NMATCH                          ;
                6F              50 E8 000B1             BLBS    R0, 12$                                 ;
                                62 B5 000B4             TSTW    (R2)                                    ; 0406
                                6B 13 000B6             BEQL    12$                                     ;
                                01 DD 000B8             PUSHL   #1                                      ; 0412
                            04  A6 9F 000BA             PUSHAB  DBG$CS_ELSE                             ;
                                52 DD 000BD             PUSHL   R2                                      ;
                67              03 FB 000BF             CALLS   #3, DBG$NMATCH                          ;
                40              50 E9 000C2 4$:         BLBC    R0, 8$                                  ;
                54          08  A5 9E 000C5             MOVAB   8(R5), LINK                             ; 0421
                                04 DD 000C9             PUSHL   #4                                      ; 0422
                68              01 FB 000CB             CALLS   #1, DBG$GET_TEMPMEM                     ;
                55              50 D0 000CE             MOVL    R0, NOUN_NODE                           ;
                64              55 D0 000D1             MOVL    NOUN_NODE, (LINK)                       ; 0423
                                01 DD 000D4             PUSHL   #1                                      ; 0427
                            02  A6 9F 000D6             PUSHAB  DBG$CS_LEFT_PAREN                       ;
                                52 DD 000D9             PUSHL   R2                                      ;
                67              03 FB 000DB             CALLS   #3, DBG$NMATCH                          ;
                3D              50 E8 000DE             BLBS    R0, 11$                                 ;
                                01 DD 000E1 5$:         PUSHL   #1                                      ; 0431
                        0044    8F BB 000E3             PUSHR   #^M<R2,R6>                              ;
                67              03 FB 000E7             CALLS   #3, DBG$NMATCH                          ;
                0F              50 E9 000EA             BLBC    R0, 7$                                  ;
                    000280D0    8F DD 000ED 6$:         PUSHL   #164048                                 ; 0433
        00000000G   00          01 FB 000F3             CALLS   #1, DBG$NMAKE_ARG_VECT                  ;
                                1B 11 000FA             BRB     9$                                      ;
                    00028743    8F DD 000FC 7$:         PUSHL   #165699                                 ; 0436
                69              01 FB 00102             CALLS   #1, LIB$SIGNAL                          ;
                                52 DD 00105 8$:         PUSHL   R2                                      ; 0437
        00000000G   00          01 FB 00107             CALLS   #1, DBG$NNEXT_WORD                      ;
                                50 DD 0010E             PUSHL   R0                                      ;
        00000000G   00          01 FB 00110             CALLS   #1, DBG$NSYNTAX_ERROR                   ;
                63              50 D0 00117 9$:         MOVL    R0, (R3)                                ; 0431
                50              04 D0 0011A 10$:        MOVL    #4, R0                                  ; 0439
                                04 0011D                RET                                             ;
                                24 BB 0011E 11$:        PUSHR   #^M<R2,R5>                              ; 0447
                6A              02 FB 00120             CALLS   #2, DBG$NSAVE_BREAK_BUFFER              ;
                50              01 D0 00123 12$:        MOVL    #1, R0                                  ; 0451
                                04 00126                RET                                             ; 0453

; Routine Size:  295 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  324     0454  1 GLOBAL ROUTINE dbg$nexecute_if (verb_node,message_vect) =
;  325     0455  1 !++
;  326     0456  1 ! Functional Description
;  327     0457  1 !
;  328     0458  1 !       This routine performs the action associated with the IF
;  329     0459  1 !       command.
;  330     0460  1 !
;  331     0461  1 ! Formal Parameters
;  332     0462  1 !
;  333     0463  1 !       verb_node       - A longword containing the address of the
;  334     0464  1 !                         head (verb) node.
;  335     0465  1 !       message_vect    - The address of a longword to contain the
;  336     0466  1 !                         address of an error message vector
;  337     0467  1 !
;  338     0468  1 ! Implicit Inputs
;  339     0469  1 !
;  340     0470  1 !       The command tree contains a verb node and a linked list
;  341     0471  1 !       of two or three noun nodes. (See the diagram in the header for
;  342     0472  1 !       DBG$NPARSE_IF).
;  343     0473  1 !
;  344     0474  1 ! Routine Value
;  345     0475  1 !
;  346     0476  1 !       A completion code.
;  347     0477  1 !
;  348     0478  1 ! Completion Codes
;  349     0479  1 !
;  350     0480  1 !       sts$k_success (1)       - Success. Command executed
;  351     0481  1 !       sts$k_severe (4)        - Failure. The command could not be
;  352     0482  1 !                                 executed. An error message is constructed.
;  353     0483  1 !
;  354     0484  1 ! Side Effects
;  355     0485  1 !
;  356     0486  1 !       Storage allocated for the THEN clause is freed up.
;  357     0487  1 !--
;  358     0488  2     BEGIN
;  359     0489  2
;  360     0490  2     MAP
;  361     0491  2         verb_node : REF dbg$verb_node;
;  362     0492  2
;  363     0493  2     LOCAL
;  364     0494  2         condition_node: REF dbg$noun_node,       ! The noun node for the IF condition
;  365     0495  2         condition_value,                        ! Should be TRUE or FALSE
;  366     0496  2         else_node: REF dbg$noun_node,           ! The noun node for the ELSE clause.
;  367     0497  2         else_string: REF VECTOR[,WORD],         ! Counted string with the ELSE clause
;  368     0498  2         then_node:      REF dbg$noun_node,      ! The noun node for the THEN clause
;  369     0499  2         then_string:    REF VECTOR[,WORD],      ! Counted string with the THEN clause
;  370     0500  2         vax_desc:       dbg$stg_desc;           ! Target of the conversion from
;  371     0501  2                                                 !    the value descriptor
;  372     0502  2                                                 !    representing the condition.
;  373     0503  2
;  374     0504  2     ! Recover the two noun nodes.
;  375     0505  2     !
;  376     0506  2     condition_node = .verb_node [dbg$l_verb_object_ptr];
;  377     0507  2     then_node = .condition_node [dbg$l_noun_link];
;  378     0508  2     else_node = .then_node [dbg$l_noun_link];
;  379     0509  2
;  380     0510  2     ! Set up the vax descriptor for the condition.
```

```
;  381          0511  2      ! *** For now, we just declare the descriptor to be longword integer,
;  382          0512  2      ! since this causes the fewest problems in the type converter.
;  383          0513  2      ! Eventually, if we get a Boolean type and all languages support
;  384          0514  2      ! it then we will build a target descriptor of this type.
;  385          0515  2      !
;  386          0516  2      vax_desc [dsc$b_class] = dsc$k_class_s;
;  387          0517  2      vax_desc [dsc$b_dtype] = dsc$k_dtype_l;
;  388          0518  2      vax_desc [dsc$w_length] = 4;
;  389          0519  2      vax_desc [dsc$a_pointer] = condition_value;
;  390          0520  2      vax_desc [dsc$l_pos] = 0;
;  391          0521  2
;  392          0522  2      ! *** Special case for PASCAL. Level 3 PASCAL returns descriptors
;  393          0523  2      ! of type Boolean (dsc$k_dtype_tf) for relational expressions.
;  394          0524  2      !
;  395          0525  2      IF .dbg$gb_language EQL dbg$k_pascal
;  396          0526  2      THEN
;  397          0527  2          BEGIN
;  398          0528  3          vax_desc [dsc$b_dtype] = dsc$k_dtype_tf;
;  399          0529  3          vax_desc [dsc$w_length] = 1;
;  400          0530  2          END;
;  401          0531  2
;  402          0532  2      ! Initialize condition_value to 0
;  403          0533  2      !
;  404          0534  2      condition_value = 0;
;  405          0535  2
;  406          0536  2      ! Do the conversion from value descriptor to integer.
;  407          0537  2      !
;  408          0538  2      IF NOT dbg$ntype_conv (.condition_node [dbg$l_noun_value],
;  409          0539  2                             dbg$k_default,
;  410          0540  2                             dbg$k_vax_desc,
;  411          0541  2                             vax_desc,
;  412          0542  2                             .message_vect)
;  413          0543  2      THEN
;  414          0544  2          RETURN sts$k_severe;
;  415          0545  2
;  416          0546  2      ! Recover the string(s).
;  417          0547  2      !
;  418          0548  2      then_string = .then_node [dbg$l_noun_value];
;  419          0549  2      IF .else_node NEQ 0
;  420          0550  2      THEN
;  421          0551  2          else_string = .else_node [dbg$l_noun_value]
;  422          0552  2      ELSE
;  423          0553  2          else_string = 0;
;  424          0554  2
;  425          0555  2      ! Process the THEN clause only if value of the condition is TRUE.
;  426          0556  2      ! For now, just use the BLISS semantics which say that a value is
;  427          0557  2      ! true iff the low bit is 1. We need to research which languages
;  428          0558  2      ! have different semantics and come up with a language-dependent
;  429          0559  2      ! method of doing this.
;  430          0560  2      !
;  431          0561  2      IF .condition_value
;  432          0562  2      THEN
;  433          0563  3          BEGIN
;  434          0564  3
;  435          0565  3          ! Add a new link to the command input stream.
;  436          0566  3          !
;  437          0567  3          IF NOT dbg$ncis_add (then_string[1], .then_string[0],
```

```
;  438          0568  3                                cis_if, 0, 0, 0, .message_vect)
;  439          0569  3          THEN
;  440          0570  3              RETURN sts$k_severe;
;  441          0571  3
;  442          0572  3          END
;  443          0573  3
;  444          0574  2      ELSE ! Process the ELSE clause
;  445          0575  2
;  446          0576  2          IF .else_string NEQ 0
;  447          0577  2          THEN
;  448          0578  2              BEGIN
;  449          0579  2
;  450          0580  2              ! Add a new link to the command input stream.
;  451          0581  2              !
;  452          0582  2              IF NOT dbg$ncis_add (else_string[1], .else_string[0],
;  453          0583  2                                   cis_if, 0, 0, 0, .message_vect)
;  454          0584  2              THEN
;  455          0585  2                  RETURN sts$k_severe;
;  456          0586  2
;  457          0587  2              END;
;  458          0588  2
;  459          0589  2      ! Return success.
;  460          0590  2      !
;  461          0591  2      RETURN sts$k_success;
;  462          0592  2
;  463          0593  1      END; ! dbg$nexecute_if
```

```
                                 000C  00000          .ENTRY  DBG$NEXECUTE_IF, Save R2,R3                     ; 0454
                           5E    10    C2  00002      SUBL2   #16, SP                                         ;
                           50    04    AC  D0  00005  MOVL    VERB_NODE, R0                                   ; 0506
                           50    08    A0  D0  00009  MOVL    8(R0), CONDITION_NODE                           ;
                           52    08    A0  D0  0000D  MOVL    8(CONDITION_NODE), THEN_NODE                    ; 0507
                           53    08    A2  D0  00011  MOVL    8(THEN_NODE), ELSE_NODE                         ; 0508
                     04    AE 01080004  8F  D0  00015  MOVL    #17301508, VAX_DESC                             ; 0518
                     08    AE           6E  9E  0001D  MOVAB   CONDITION_VALUE, VAX_DESC+4                     ; 0519
                                 0C    AE  D4  00021  CLRL    VAX_DESC+8                                      ; 0520
                           06 00000000G 00  91  00024  CMPB    DBG$GB_LANGUAGE, #6                             ; 0525
                                       08  12  0002B  BNEQ    1$                                              ;
                     06    AE           28  90  0002D  MOVB    #40, VAX_DESC+2                                 ; 0528
                     04    AE           01  B0  00031  MOVW    #1, VAX_DESC                                    ; 0529
                                       6E  D4  00035  1$:     CLRL    CONDITION_VALUE                         ; 0534
                                 08    AC  DD  00037  PUSHL   MESSAGE_VECT                                    ; 0542
                                 08    AE  9F  0003A  PUSHAB  VAX_DESC                                        ; 0538
                           7E    82    8F  9A  0003D  MOVZBL  #130, -(SP)                                     ;
                                       01  DD  00041  PUSHL   #1                                             ;
                                       60  DD  00043  PUSHL   (CONDITION_NODE)                              ;
                 00000000G 00          05  FB  00045  CALLS   #5, DBG$NTYPE_CONV                             ;
                           3B          50  E9  0004C  BLBC    R0, 6$                                          ;
                           50          62  D0  0004F  MOVL    (THEN_NODE), THEN_STRING                        ; 0548
                                       53  D5  00052  TSTL    ELSE_NODE                                       ; 0549
                                       05  13  00054  BEQL    2$                                              ;
                           52          63  D0  00056  MOVL    (ELSE_NODE), ELSE_STRING                        ; 0551
                                       02  11  00059  BRB     3$                                              ;
```

```
                                 52  D4 0005B 2$:    CLRL    ELSE_STRING                             ; 0553
                              10 6E  E9 0005D 3$:    BLBC    CONDITION_VALUE, 4$                     ; 0561
                           08 AC  DD 00060           PUSHL   MESSAGE_VECT                            ; 0568
                              7E  7C 00063           CLRQ    -(SP)                                   ; 0567
                           7E 06  7D 00065           MOVQ    #6, -(SP)                               ;
                           7E 60  3C 00068           MOVZWL  (THEN_STRING), -(SP)                    ;
                           02 A0  9F 0006B           PUSHAB  2(THEN_STRING)                          ;
                              10  11 0006E           BRB     5$                                      ;
                              1C  13 00070 4$:       BEQL    7$                                      ; 0576
                           08 AC  DD 00072           PUSHL   MESSAGE_VECT                            ; 0583
                              7E  7C 00075           CLRQ    -(SP)                                   ; 0582
                           7E 06  7D 00077           MOVQ    #6, -(SP)                               ;
                           7E 62  3C 0007A           MOVZWL  (ELSE_STRING), -(SP)                    ;
                           02 A2  9F 0007D           PUSHAB  2(ELSE_STRING)                          ;
                00000000G  00 07  FB 00080 5$:       CALLS   #7, DBG$NCIS_ADD                        ;
                           04 50  E8 00087           BLBS    R0, 7$                                  ;
                           50 04  D0 0008A 6$:       MOVL    #4, R0                                  ; 0585
                              04  0008D              RET                                             ;
                           50 01  D0 0008E 7$:       MOVL    #1, R0                                  ; 0591
                              04  00091              RET                                             ; 0593

; Routine Size:  146 bytes,    Routine Base:  DBG$CODE + 0127
```

```
;  465     0594  1 GLOBAL ROUTINE dbg$nparse_while(input_desc, verb_node, message_vect) =
;  466     0595  1 !
;  467     0596  1 ! Functional Description
;  468     0597  1 !
;  469     0598  1 !       ATN parse network for the WHILE verb.
;  470     0599  1 !       This routine takes a verb node for the WHILE verb, and a string
;  471     0600  1 !       descriptor for the remaining (unparsed) input.
;  472     0601  1 !       A command execution tree is built. The form of the tree is:
;  473     0602  1 !
;  474     0603  1 !        -----------       -----------       -----------
;  475     0604  1 !       | verb node |-->--| noun node |-->--| noun node |
;  476     0605  1 !        -----------       -----------       -----------
;  477     0606  1 !
;  478     0607  1 !       The first noun node points to a value descriptor for the condition.
;  479     0608  1 !       The second noun node points to a counted string with the DO clause.
;  480     0609  1 !
;  481     0610  1 ! Formal Parameters
;  482     0611  1 !
;  483     0612  1 !       input_desc      - A longword containing the address of the
;  484     0613  1 !                         command input descriptor.
;  485     0614  1 !       verb_node       - A longword containing the address of the verb node.
;  486     0615  1 !       message_vect    - The address of a longword to contain the address
;  487     0616  1 !                         of a standard message argument vector.
;  488     0617  1 !
;  489     0618  1 ! Implicit Inputs
;  490     0619  1 !
;  491     0620  1 !       none
;  492     0621  1 !
;  493     0622  1 ! Implicit Outputs
;  494     0623  1 !
;  495     0624  1 !       On success, the command execution tree is constructed.
;  496     0625  1 !       On failure, a message argument vector is constructed or obtained.
;  497     0626  1 !
;  498     0627  1 ! Routine value
;  499     0628  1 !
;  500     0629  1 !       sts$k_success (1)       - Success. Command execution tree constructed.
;  501     0630  1 !       sts$k_severe  (4)       - Failure. Error encountered. Message argument
;  502     0631  1 !                                 constructed and returned.
;  503     0632  1 !
;  504     0633  1 ! Side Effects
;  505     0634  1 !
;  506     0635  1 !       Permanent storage is allocated for the string holding the DO clause;
;  507     0636  1 !       this is released in DBG$NEXECUTE_WHILE after execution.
;  508     0637  1 !
;  509     0638  1
;  510     0639  2     BEGIN
;  511     0640  2
;  512     0641  2     MAP
;  513     0642  2         input_desc: REF dbg$stg_desc,
;  514     0643  2         verb_node: REF dbg$verb_node;
;  515     0644  2
;  516     0645  2     BIND
;  517     0646  2         dbg$cs_cr                   = UPLIT BYTE (1, dbg$k_car_return),
;  518     0647  2         dbg$cs_left_paren           = UPLIT BYTE (1, dbg$k_left_parenthesis),
;  519     0648  2         dbg$cs_do                   = UPLIT BYTE (4, 'DO');
;  520     0649  2
;  521     0650  2     LOCAL
```

```
;  522          0651  2          link,                                  ! Temporary to hold links in the command
;  523          0652  2                                                 !    execution tree.
;  524          0653  2          noun_node : REF dbg$noun_node,         ! A node in the command execution tree.
;  525          0654  2          radix,                                 ! Holds the current radix setting.
;  526          0655  2          status;                                ! Return status from subroutine calls.
;  527          0656  2
;  528          0657  2
;  529          0658  2      ! Create and link a noun node
;  530          0659  2      !
;  531          0660  2      noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  532          0661  2      verb_node[dbg$l_verb_object_ptr] = .noun_node;
;  533          0662  2
;  534          0663  2      ! Determine the current radix.
;  535          0664  2      !
;  536          0665  2      radix = .dbg$gb_radix[dbg$b_radix_input];
;  537          0666  2
;  538          0667  2
;  539          0668  2      ! Obtain a value descriptor for the condition. The first noun node
;  540          0669  2      ! points to this value descriptor.
;  541          0670  2      !
;  542          0671  2      STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
;  543          0672  2                                     NOUN_NODE [DBG$L_NOUN_VALUE],
;  544          0673  2                                     TOKEN$K_TERM_DO, .MESSAGE_VECT);
;  545          0674  2
;  546          0675  2
;  547          0676  2      ! The return status should be "warning", meaning that an expression
;  548          0677  2      ! was parsed and further input remains. If an expression was parsed
;  549          0678  2      ! and no input remains, NPARSE_EXPRESSION will return "success".
;  550          0679  2      ! In this context, it is an error since "WHILE exp" by itself
;  551          0680  2      ! is an error.
;  552          0681  2      !
;  553          0682  2      !
;  554          0683  2      IF .status EQL sts$k_success
;  555          0684  2      THEN
;  556          0685  3          BEGIN
;  557          0686  3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  558          0687  3          RETURN sts$k_severe;
;  559          0688  2          END;
;  560          0689  2
;  561          0690  2      ! Severe status is also an error.
;  562          0691  2      !
;  563          0692  2      IF .status EQL sts$k_severe
;  564          0693  2      THEN
;  565          0694  2          RETURN sts$k_severe;
;  566          0695  2
;  567          0696  2      ! Eat the DO
;  568          0697  2      !
;  569          0698  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_do, 1)
;  570          0699  2      THEN
;  571          0700  3          BEGIN
;  572          0701  3          .message_vect =
;  573          0702  4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  574          0703  4                  OR .input_desc [dsc$w_length] EQL 0
;  575          0704  4               THEN
;  576          0705  4                  dbg$nmake_arg_vect (dbg$_needmore)
;  577          0706  4               ELSE
;  578          0707  3                  dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
```

```
;  579          0708  3          RETURN sts$k_severe;
;  580          0709  2          END;
;  581          0710  2
;  582          0711  2      ! Allocate and link a noun node for the DO clause.
;  583          0712  2      !
;  584          0713  2      link = noun_node [dbg$l_noun_link];
;  585          0714  2      noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  586          0715  2      .link = .noun_node;
;  587          0716  2
;  588          0717  2      ! Eat the left parenthesis which we require be present.
;  589          0718  2      !
;  590          0719  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  591          0720  2      THEN
;  592          0721  3          BEGIN
;  593          0722  3          .message_vect =
;  594          0723  4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  595          0724  4               THEN
;  596          0725  4                   dbg$nmake_arg_vect (dbg$_needmore)
;  597          0726  4               ELSE
;  598          0727  5                   BEGIN
;  599          0728  5                   SIGNAL(dbg$_needparen);
;  600          0729  5                   dbg$nsyntax_error (dbg$nnext_word (.input_desc))
;  601          0730  5                   END);
;  602          0731  3          RETURN sts$k_severe;
;  603          0732  2          END;
;  604          0733  2
;  605          0734  2      ! Put a pointer to the counted string representing the DO
;  606          0735  2      ! clause into the second noun node. (Note - the counted string
;  607          0736  2      ! is constructed out of "permanent" memory which is released
;  608          0737  2      ! in DBG$NEXECUTE_IF).
;  609          0738  2      !
;  610          0739  2      dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
;  611          0740  2
;  612          0741  2      ! Return success.
;  613          0742  2      !
;  614          0743  2      RETURN sts$k_success;
;  615          0744  2
;  616          0745  1      END;


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

                                    0D  01  0000E P.AAE:  .BYTE   1, 13                                   ;
                                    28  01  00010 P.AAF:  .BYTE   1, 40                                   ;
                                        04  00012 P.AAG:  .BYTE   4                                       ;
                                    4F  44  00013         .ASCII  \DO\                                    ;

                                                DBG$CS_CR=          P.AAE
                                                DBG$CS_LEFT_PAREN=  P.AAF
                                                DBG$CS_DO=          P.AAG


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

                                          00FC 00000            .ENTRY  DBG$NPARSE_WHILE, Save R2,R3,R4,R5,R6,R7    ; 0594
                                57 00000000G  00  9E 00002      MOVAB   DBG$GET_TEMPMEM, R7                         ;
```

```
                    56 00000000G   00   9E 00009          MOVAB   DBG$NMATCH, R6                                ;
                    55 00000000'   EF   9E 00010          MOVAB   DBG$CS_CR, R5                               ;
                                   04   DD 00017          PUSHL   #4                                          ; 0661
                    67             01   FB 00019          CALLS   #1, DBG$GET_TEMPMEM                         ;
                    54             50   D0 0001C          MOVL    R0, NOUN_NODE                               ;
                    50       08    AC   D0 0001F          MOVL    VERB_NODE, R0                               ; 0662
              08    A0             54   D0 00023          MOVL    NOUN_NODE, 8(R0)                            ;
                    50 00000000G   00   9A 00027          MOVZBL  DBG$GB_RADIX, RADIX                         ; 0666
                             0C    AC   DD 0002E          PUSHL   MESSAGE_VECT                                ; 0674
                                   05   DD 00031          PUSHL   #5                                          ; 0673
                                   11   BB 00033          PUSHR   #^M<R0,R4>                                  ;
                    52       04    AC   D0 00035          MOVL    INPUT_DESC, R2                              ; 0672
                                   52   DD 00039          PUSHL   R2                                          ; 0673
 00000000G          00             05   FB 0003B          CALLS   #5, DBG$NPARSE_EXPRESSION                   ;
                    53             50   D0 00042          MOVL    R0, STATUS                                  ;
                    01             53   D1 00045          CMPL    STATUS, #1                                  ; 0683
                                   48   13 00048          BEQL    2$                                          ;
                    04             53   D1 0004A          CMPL    STATUS, #4                                  ; 0692
                                   75   13 0004D          BEQL    6$                                          ;
                                   01   DD 0004F          PUSHL   #1                                          ; 0698
                             04    A5   9F 00051          PUSHAB  DBG$CS_DO                                   ;
                                   52   DD 00054          PUSHL   R2                                          ;
                    66             03   FB 00056          CALLS   #3, DBG$NMATCH                              ;
                    10             50   E8 00059          BLBS    R0, 1$                                      ;
                                   01   DD 0005C          PUSHL   #1                                          ; 0702
                                   24   BB 0005E          PUSHR   #^M<R2,R5>                                  ;
                    66             03   FB 00060          CALLS   #3, DBG$NMATCH                              ;
                    2C             50   E8 00063          BLBS    R0, 2$                                      ;
                                   62   B5 00066          TSTW    (R2)                                        ; 0703
                                   44   12 00068          BNEQ    4$                                          ;
                                   26   11 0006A          BRB     2$                                          ; 0705
                    53       08    A4   9E 0006C  1$:     MOVAB   8(R4), LINK                                 ; 0713
                                   04   DD 00070          PUSHL   #4                                          ; 0714
                    67             01   FB 00072          CALLS   #1, DBG$GET_TEMPMEM                         ;
                    54             50   D0 00075          MOVL    R0, NOUN_NODE                               ;
                    63             54   D0 00078          MOVL    NOUN_NODE, (LINK)                           ; 0715
                                   01   DD 0007B          PUSHL   #1                                          ; 0719
                             02    A5   9F 0007D          PUSHAB  DBG$CS_LEFT_PAREN                           ;
                                   52   DD 00080          PUSHL   R2                                          ;
                    66             03   FB 00082          CALLS   #3, DBG$NMATCH                              ;
                    40             50   E8 00085          BLBS    R0, 7$                                      ;
                                   01   DD 00088          PUSHL   #1                                          ; 0723
                                   24   BB 0008A          PUSHR   #^M<R2,R5>                                  ;
                    66             03   FB 0008C          CALLS   #3, DBG$NMATCH                              ;
                    0F             50   E9 0008F          BLBC    R0, 3$                                      ;
                          000280D0 8F   DD 00092  2$:     PUSHL   #164048                                     ; 0725
 00000000G          00             01   FB 00098          CALLS   #1, DBG$NMAKE_ARG_VECT                      ;
                                   1F   11 0009F          BRB     5$                                          ;
                          00028743 8F   DD 000A1  3$:     PUSHL   #165699                                     ; 0728
 00000000G          00             01   FB 000A7          CALLS   #1, LIB$SIGNAL                              ;
                                   52   DD 000AE  4$:     PUSHL   R2                                          ; 0729
 00000000G          00             01   FB 000B0          CALLS   #1, DBG$NNEXT_WORD                          ;
                                   50   DD 000B7          PUSHL   R0                                          ;
 00000000G          00             01   FB 000B9          CALLS   #1, DBG$NSYNTAX_ERROR                       ;
                    0C    BC       50   D0 000C0  5$:     MOVL    R0, @MESSAGE_VECT                           ; 0723
                    50             04   D0 000C4  6$:     MOVL    #4, R0                                      ; 0731
                                   04   000C7              RET                                                 ;
```

```
                                    14  BB 000C8 7$:    PUSHR   #^M<R2,R4>                              ; 0739
                00000000G   00      02  FB 000CA        CALLS   #2, DBG$NSAVE_BREAK_BUFFER              ;
                            50      01  D0 000D1        MOVL    #1, R0                                  ; 0743
                                        04 000D4        RET                                             ; 0745

; Routine Size:  213 bytes,    Routine Base:  DBG$CODE + 01B9
```

```
;  618     0746  1 GLOBAL ROUTINE dbg$nexecute_while (verb_node,message_vect) =
;  619     0747  1 !++
;  620     0748  1 ! Functional Description
;  621     0749  1 !
;  622     0750  1 !       This routine performs the action associated with the WHILE
;  623     0751  1 !       command.
;  624     0752  1 !
;  625     0753  1 ! Formal Parameters
;  626     0754  1 !
;  627     0755  1 !       verb_node       - A longword containing the address of the
;  628     0756  1 !                         head (verb) node.
;  629     0757  1 !       message_vect    - The address of a longword to contain the
;  630     0758  1 !                         address of an error message vector
;  631     0759  1 !
;  632     0760  1 ! Implicit Inputs
;  633     0761  1 !
;  634     0762  1 !       The command tree contains a verb node and a linked list
;  635     0763  1 !       of two noun nodes. (See the diagram in the header for
;  636     0764  1 !       DBG$NPARSE_WHILE).
;  637     0765  1 !
;  638     0766  1 ! Routine Value
;  639     0767  1 !
;  640     0768  1 !       A completion code.
;  641     0769  1 !
;  642     0770  1 ! Completion Codes
;  643     0771  1 !
;  644     0772  1 !       sts$k_success (1)       - Success. Command executed
;  645     0773  1 !       sts$k_severe (4)        - Failure. The command could not be
;  646     0774  1 !                                 executed. An error message is constructed.
;  647     0775  1 !
;  648     0776  1 ! Side Effects
;  649     0777  1 !
;  650     0778  1 !       None
;  651     0779  1 !--
;  652     0780  2     BEGIN
;  653     0781  2
;  654     0782  2     MAP
;  655     0783  2         verb_node : REF dbg$verb_node;
;  656     0784  2
;  657     0785  2     LOCAL
;  658     0786  2         condition_node: REF dbg$noun_node,       ! The noun node for the IF condition
;  659     0787  2         condition_value,                        ! Should be TRUE or FALSE
;  660     0788  2         do_node:        REF dbg$noun_node,      ! The noun node for the THEN clause
;  661     0789  2         do_string: REF VECTOR[,WORD],           ! Counted string for the do clause
;  662     0790  2         vax_desc:  dbg$stg_desc;                ! Target of the conversion from
;  663     0791  2                                                 !       the value descriptor.
;  664     0792  2
;  665     0793  2
;  666     0794  2     ! Recover the two noun nodes.
;  667     0795  2     !
;  668     0796  2     condition_node = .verb_node [dbg$l_verb_object_ptr];
;  669     0797  2     do_node = .condition_node [dbg$l_noun_link];
;  670     0798  2
;  671     0799  2     ! Set up the vax descriptor for the condition.
;  672     0800  2     !
;  673     0801  2     vax_desc [dsc$b_class] = dsc$k_class_s;
;  674     0802  2     vax_desc [dsc$b_dtype] = dsc$k_dtype_t;
```

```
;  675          0803  2      vax_desc [dsc$w_length] = 4;
;  676          0804  2      vax_desc [dsc$a_pointer] = condition_value;
;  677          0805  2      vax_desc [dsc$l_pos] = 0;
;  678          0806  2
;  679          0807  2      ! Special case for level 3 PASCAL. PASCAL returns descriptors
;  680          0808  2      ! of type boolean (dsc$k_dtype_tf) for relational expressions.
;  681          0809  2      !
;  682          0810  2      IF .dbg$gb_language EQL dbg$k_pascal
;  683          0811  2      THEN
;  684          0812  3          BEGIN
;  685          0813  3          vax_desc [dsc$b_dtype] = dsc$k_dtype_tf;
;  686          0814  3          vax_desc [dsc$w_length] = 1;
;  687          0815  3          END;
;  688          0816  2
;  689          0817  2      ! Initialize condition_value to zero.
;  690          0818  2      !
;  691          0819  2      condition_value = 0;
;  692          0820  2
;  693          0821  2      ! Do the conversion from value descriptor to integer.
;  694          0822  2      !
;  695          0823  2      IF NOT dbg$ntype_conv (.condition_node [dbg$l_noun_value],
;  696          0824  2                             dbg$k_default,
;  697          0825  2                             dbg$k_vax_desc,
;  698          0826  2                             vax_desc,
;  699          0827  2                             .message_vect)
;  700          0828  2      THEN
;  701          0829  2          RETURN sts$k_severe;
;  702          0830  2
;  703          0831  2      ! Continue only of condition is true. For now, just use BLISS semantics.
;  704          0832  2      !
;  705          0833  2      IF .condition_value
;  706          0834  2      THEN
;  707          0835  3          BEGIN
;  708          0836  3
;  709          0837  3          ! Recover the do string.
;  710          0838  3          !
;  711          0839  3          do_string = .do_node [dbg$l_noun_value];
;  712          0840  3
;  713          0841  3          ! Add a link to the command input stream
;  714          0842  3          !
;  715          0843  3          IF NOT dbg$ncis_add (do_string[1], .do_string[0], cis_while,
;  716          0844  3                  0, TRUE, 0, .message_vect)
;  717          0845  3          THEN
;  718          0846  4              RETURN sts$k_severe;
;  719          0847  3
;  720          0848  3          END
;  721          0849  2
;  722          0850  2      ELSE
;  723          0851  3          ! Add a cis for null action
;  724          0852  3          !
;  725          0853  3          BEGIN
;  726          0854  3          LOCAL
;  727          0855  3              dummy: REF VECTOR[,WORD];
;  728          0856  3          dummy = dbg$get_memory (1);
;  729          0857  3          IF NOT dbg$ncis_add (dummy[1], 0, cis_while, 0, FALSE, 0, .message_vect)
;  730          0858  3          THEN
;  731          0859  4              RETURN sts$k_severe;
```

```
;  732          0860  2          END;
;  733          0861  2
;  734          0862  2      ! Return success.
;  735          0863  2      !
;  736          0864  2      RETURN sts$k_success;
;  737          0865  2
;  738          0866  1      END; ! dbg$nexecute_while


                                        0004 00000          .ENTRY  DBG$NEXECUTE_WHILE, Save R2                     ; 0746
                                    5E  10  C2 00002        SUBL2   #16, SP                                         ;
                                    50  04  AC  D0 00005    MOVL    VERB_NODE, R0                                   ; 0796
                                    50  08  A0  D0 00009    MOVL    8(R0), CONDITION_NODE                           ;
                                    52  08  A0  D0 0000D    MOVL    8(CONDITION_NODE), DO_NODE                      ; 0797
                              04    AE 01080004   8F  D0 00011    MOVL    #17301508, VAX_DESC                     ; 0803
                              08    AE            6E  9E 00019    MOVAB   CONDITION_VALUE, VAX_DESC+4             ; 0804
                                       0C         AE  D4 0001D    CLRL    VAX_DESC+8                              ; 0805
                                    06 00000000G  00  91 00020    CMPB    DBG$GB_LANGUAGE, #6                     ; 0810
                                                  08  12 00027    BNEQ    1$                                      ;
                              06    AE            28  90 00029    MOVB    #40, VAX_DESC+2                         ; 0813
                              04    AE            01  B0 0002D    MOVW    #1, VAX_DESC                            ; 0814
                                                  6E  D4 00031 1$:  CLRL    CONDITION_VALUE                         ; 0819
                                          08      AC  DD 00033    PUSHL   MESSAGE_VECT                            ; 0827
                                          08      AE  9F 00036    PUSHAB  VAX_DESC                                ; 0823
                                       7E         82  8F  9A 00039    MOVZBL  #130, -(SP)                         ;
                                                  01  DD 0003D    PUSHL   #1                                      ;
                                                  60  DD 0003F    PUSHL   (CONDITION_NODE)                        ;
                                 00000000G        00  05  FB 00041    CALLS   #5, DBG$NTYPE_CONV                  ;
                                                  34  50  E9 00048    BLBC    R0, 4$                              ;
                                                  11  6E  E9 0004B    BLBC    CONDITION_VALUE, 2$                 ; 0833
                                                  50  62  D0 0004E    MOVL    (DO_NODE), DO_STRING                ; 0839
                                          08      AC  DD 00051    PUSHL   MESSAGE_VECT                            ; 0844
                                                  7E  01  7D 00054    MOVQ    #1, -(SP)                           ; 0843
                                                  7E  05  7D 00057    MOVQ    #5, -(SP)                           ;
                                                  7E  60  3C 0005A    MOVZWL  (DO_STRING), -(SP)                  ;
                                                      13  11 0005D    BRB     3$                                  ;
                                                  01  DD 0005F 2$:  PUSHL   #1                                  ; 0856
                                 00000000G        00  01  FB 00061    CALLS   #1, DBG$GET_MEMORY                  ;
                                          08      AC  DD 00068    PUSHL   MESSAGE_VECT                            ; 0857
                                                  7E  7C 0006B    CLRQ    -(SP)                                   ;
                                                  7E  05  7D 0006D    MOVQ    #5, -(SP)                           ;
                                                  7E  D4 00070    CLRL    -(SP)                                   ;
                                          02      A0  9F 00072 3$:  PUSHAB  2(DUMMY)                            ;
                                 00000000G        00  07  FB 00075    CALLS   #7, DBG$NCIS_ADD                    ;
                                                  04  50  E8 0007C    BLBS    R0, 5$                              ;
                                                  50  04  D0 0007F 4$:  MOVL    #4, R0                          ; 0859
                                                      04  04 00082    RET                                         ;
                                                  50  01  D0 00083 5$:  MOVL    #1, R0                          ; 0864
                                                      04  04 00086    RET                                         ; 0866

; Routine Size:  135 bytes,    Routine Base:  DBG$CODE + 028E
```

```
;  740     0867  1 GLOBAL ROUTINE dbg$nparse_for (input_desc, verb_node, message_vect) =
;  741     0868  1 !
;  742     0869  1 ! Functional Description
;  743     0870  1 !
;  744     0871  1 !       ATN parse network for the FOR verb.
;  745     0872  1 !       This routine takes a verb node for the FOR verb, and a string
;  746     0873  1 !       descriptor for the remaining (unparsed) input.
;  747     0874  1 !       A command execution tree is built. The form of the tree is:
;  748     0875  1 !
;  749     0876  1 !        -----------      -----------      -----------      -----------
;  750     0877  1 !       ! verb node !-->--! noun node !-->--! noun node ! -->-- ! noun node !
;  751     0878  1 !        -----------      -----------      -----------      -----------
;  752     0879  1 !
;  753     0880  1 !       The first noun node contains a counted string with the name of the
;  754     0881  1 !       loop variable.
;  755     0882  1 !       The second noun node contains value descriptors with the lower and
;  756     0883  1 !       upper bounds, and loop increment
;  757     0884  1 !       The third noun node contains a counted string with the command list.
;  758     0885  1 !
;  759     0886  1 ! Formal Parameters
;  760     0887  1 !
;  761     0888  1 !       input_desc      - A longword containing the address of the
;  762     0889  1 !                         command input descriptor.
;  763     0890  1 !       verb_node       - A longword containing the address of the verb node.
;  764     0891  1 !       message_vect    - The address of a longword to contain the address
;  765     0892  1 !                         of a standard message argument vector.
;  766     0893  1 !
;  767     0894  1 ! Implicit Inputs
;  768     0895  1 !
;  769     0896  1 !       none
;  770     0897  1 !
;  771     0898  1 ! Implicit Outputs
;  772     0899  1 !
;  773     0900  1 !       On success, the command execution tree is constructed.
;  774     0901  1 !       On failure, a message argument vector is constructed or obtained.
;  775     0902  1 !
;  776     0903  1 ! Routine value
;  777     0904  1 !
;  778     0905  1 !       sts$k_success (1)       - Success. Command execution tree constructed.
;  779     0906  1 !       sts$k_severe  (4)       - Failure. Error encountered. Message argument
;  780     0907  1 !                                 constructed and returned.
;  781     0908  1 !
;  782     0909  1 ! Side Effects
;  783     0910  1 !
;  784     0911  1 !       Permanent storage is allocated for the string holding the action
;  785     0912  1 !       clause and for the string holding the loop variable name.
;  786     0913  1 !       This is released in DBG$NCIS_REMOVE after execution
;  787     0914  1 !       of the action clause.
;  788     0915  1 !
;  789     0916  1
;  790     0917  2     BEGIN
;  791     0918  2
;  792     0919  2     MAP
;  793     0920  2         input_desc  : REF dbg$stg_desc,
;  794     0921  2         verb_node       : REF dbg$verb_node;
;  795     0922  2
;  796     0923  2     BIND
```

```
;  797     0924  2          dbg$cs_comma            = UPLIT BYTE (1, dbg$k_comma),
;  798     0925  2          dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
;  799     0926  2          dbg$cs_equal            = UPLIT BYTE (1, dbg$k_equal),
;  800     0927  2          dbg$cs_left_paren       = UPLIT BYTE (1, dbg$k_left_parenthesis),
;  801     0928  2          dbg$cs_by               = UPLIT BYTE (2, 'BY'),
;  802     0929  2          dbg$cs_do               = UPLIT BYTE (2, 'DO'),
;  803     0930  2          dbg$cs_to               = UPLIT BYTE (2, 'TO');
;  804     0931  2
;  805     0932  2      LOCAL
;  806     0933  2          link,                                   ! Temporary to hold links in the command
;  807     0934  2                                                  !     execution tree.
;  808     0935  2          noun_node : REF dbg$noun_node,          ! A node in the command execution tree.
;  809     0936  2          radix,                                  ! Holds the current radix setting.
;  810     0937  2          status;                                 ! Holds return status from subroutine
;  811     0938  2                                                  !     calls.
;  812     0939  2
;  813     0940  2
;  814     0941  2      ! Create and link a noun node
;  815     0942  2      !
;  816     0943  2      noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  817     0944  2      verb_node[dbg$l_verb_object_ptr] = .noun_node;
;  818     0945  2
;  819     0946  2      ! Pick up the name of the loop counter.
;  820     0947  2      ! Note that dbg$nread_name allocates permanent storage for the name.
;  821     0948  2      ! This must be released in DBG$NCIS_REMOVE when the command buffer is
;  822     0949  2      ! no longer needed.
;  823     0950  2      !
;  824     0951  2      IF NOT dbg$nread_name (.input_desc,
;  825     0952  2                             noun_node [dbg$l_noun_value],
;  826     0953  2                             .message_vect)
;  827     0954  2      THEN
;  828     0955  2          RETURN sts$k_severe;
;  829     0956  2
;  830     0957  2      ! Eat the =
;  831     0958  2      !
;  832     0959  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_equal, 1)
;  833     0960  2      THEN
;  834     0961  2          report_error;
;  835     0962  2
;  836     0963  2      ! Create and link a noun node
;  837     0964  2      !
;  838     0965  2      link = noun_node [dbg$l_noun_link];
;  839     0966  2      noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  840     0967  2      .link = .noun_node;
;  841     0968  2
;  842     0969  2      ! Determine the current radix.
;  843     0970  2      !
;  844     0971  2      radix  = .dbg$gb_radix[dbg$b_radix_input];
;  845     0972  2
;  846     0973  2      ! Obtain a value descriptor for the lower bound. The noun_value field
;  847     0974  2      ! points to this descriptor.
;  848     0975  2      !
;  849     0976  2      STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
;  850     0977  2                                     NOUN_NODE [DBG$L_NOUN_VALUE],
;  851     0978  2                                     TOKEN$K_TERM_TO, .MESSAGE_VECT);
;  852     0979  2
;  853     0980  2      ! The return status should be "warning", meaning that an expression
```

```
;  854          0981  2      ! was parsed and further input reamins. If an expression was parsed
;  855          0982  2      ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
;  856          0983  2      ! In this context, it is an error since "REPEAT count" by itself
;  857          0984  2      ! is an error.
;  858          0985  2      !
;  859          0986  2      IF .status EQL sts$k_success
;  860          0987  2      THEN
;  861          0988  3          BEGIN
;  862          0989  3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  863          0990  3          RETURN sts$k_severe;
;  864          0991  2          END;
;  865          0992  2
;  866          0993  2      ! Severe status is also an error.
;  867          0994  2      !
;  868          0995  2      IF .status EQL sts$k_severe
;  869          0996  2      THEN
;  870          0997  2          RETURN sts$k_severe;
;  871          0998  2
;  872          0999  2      ! Eat the "TO".
;  873          1000  2      !
;  874          1001  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_to, 2)
;  875          1002  2      THEN
;  876          1003  2          report_error;
;  877          1004  2
;  878          1005  2      ! Obtain a value descriptor for the upper bound. The noun_value2 field
;  879          1006  2      ! points to this descriptor.
;  880          1007  2      !
;  881          1008  2      STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
;  882          1009  2                                     NOUN_NODE [DBG$L_NOUN_VALUE2],
;  883          1010  2                                     TOKEN$K_TERM_BY, .MESSAGE_VECT);
;  884          1011  2
;  885          1012  2
;  886          1013  2      ! The return status should be "warning", meaning that an expression
;  887          1014  2      ! was parsed and further input reamins. If an expression was parsed
;  888          1015  2      ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
;  889          1016  2      ! In this context, it is an error since "REPEAT count" by itself
;  890          1017  2      ! is an error.
;  891          1018  2      !
;  892          1019  2      IF .status EQL sts$k_success
;  893          1020  2      THEN
;  894          1021  3          BEGIN
;  895          1022  3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  896          1023  3          RETURN sts$k_severe;
;  897          1024  2          END;
;  898          1025  2
;  899          1026  2      ! Severe status is also an error.
;  900          1027  2      !
;  901          1028  2      IF .status EQL sts$k_severe
;  902          1029  2      THEN
;  903          1030  2          RETURN sts$k_severe;
;  904          1031  2
;  905          1032  2      ! Check for BY clause.
;  906          1033  2      !
;  907          1034  2      IF dbg$nmatch (.input_desc, dbg$cs_by, 2)
;  908          1035  2      THEN
;  909          1036  3          BEGIN
;  910          1037  3
```

```
;  911     1038  3          ! Obtain a value descriptor for the increment.
;  912     1039  3          !
;  913     1040  3          STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
;  914     1041  3                                         NOUN_NODE [DBG$L_ADJECTIVE_PTR],
;  915     1042  3                                         TOKEN$K_TERM_DO, .MESSAGE_VECT);
;  916     1043  3
;  917     1044  3          ! The return status should be "warning", meaning that an expression
;  918     1045  3          ! was parsed and further input reamins. If an expression was parsed
;  919     1046  3          ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
;  920     1047  3          ! In this context, it is an error since "FOR I=1 TO N BY" by itself
;  921     1048  3          ! is an error.
;  922     1049  3          !
;  923     1050  3          IF .status EQL sts$k_success
;  924     1051  3          THEN
;  925     1052  4              BEGIN
;  926     1053  4              .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  927     1054  4              RETURN sts$k_severe;
;  928     1055  3              END;
;  929     1056  3
;  930     1057  3          ! Severe status is also an error.
;  931     1058  3          !
;  932     1059  3          IF .status EQL sts$k_severe
;  933     1060  3          THEN
;  934     1061  3              RETURN sts$k_severe;
;  935     1062  3
;  936     1063  3          END
;  937     1064  2      ELSE
;  938     1065  2          noun_node [dbg$l_adjective_ptr] = 0;
;  939     1066  2
;  940     1067  2      ! Eat the "DO".
;  941     1068  2      !
;  942     1069  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_do, 2)
;  943     1070  2      THEN
;  944     1071  2          report_error;
;  945     1072  2
;  946     1073  2      ! Allocate and link a noun node for the action clause.
;  947     1074  2      !
;  948     1075  2      link = noun_node [dbg$l_noun_link];
;  949     1076  2      noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  950     1077  2      .link = .noun_node;
;  951     1078  2
;  952     1079  2      ! Eat the left parenthesis which we require be present.
;  953     1080  2      !
;  954     1081  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  955     1082  2      THEN
;  956     1083  2          report_error;
;  957     1084  2
;  958     1085  2      ! Put a pointer to the counted string representing the action
;  959     1086  2      ! clause into the second noun node. (Note - the counted string
;  960     1087  2      ! is constructed out of "permanent" memory which is released
;  961     1088  2      ! in DBG$NCIS_REMOVE).
;  962     1089  2      ! The third argument indicates that save_break_buffer is not being
;  963     1090  2      ! called during parsing of a SET BREAK DO (The routine behaves
;  964     1091  2      ! slightly differently in that case)
;  965     1092  2      !
;  966     1093  2      dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
;  967     1094  2
```

```
;  968          1095  2
;  969          1096  2      ! Return success.
;  970          1097  2      !
;  971          1098  2      RETURN sts$k_success;
;  972          1099  2
;  973          1100  1      END;


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

                                2C  01  00015 P.AAH:  .BYTE   1, 44                                     ;
                                0D  01  00017 P.AAI:  .BYTE   1, 13                                     ;
                                3D  01  00019 P.AAJ:  .BYTE   1, 61                                     ;
                                28  01  0001B P.AAK:  .BYTE   1, 40                                     ;
                                    02  0001D P.AAL:  .BYTE   2                                         ;
                                59  42  0001E         .ASCII  \BY\                                      ;
                                    02  00020 P.AAM:  .BYTE   2                                         ;
                                4F  44  00021         .ASCII  \DO\                                      ;
                                    02  00023 P.AAN:  .BYTE   2                                         ;
                                4F  54  00024         .ASCII  \TO\                                      ;

                                        DBG$CS_COMMA=           P.AAH
                                        DBG$CS_CR=              P.AAI
                                        DBG$CS_EQUAL=           P.AAJ
                                        DBG$CS_LEFT_PAREN=      P.AAK
                                        DBG$CS_BY=              P.AAL
                                        DBG$CS_DO=              P.AAM
                                        DBG$CS_TO=              P.AAN

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

                                    0FFC 00000         .ENTRY  DBG$NPARSE_FOR, Save R2,R3,R4,R5,R6,R7,R8,- ; 0867
                                                               R9,R10,R11                                   ;
                            5B 00000000G   00  9E 00002        MOVAB   DBG$GET_TEMPMEM, R11                ;
                            5A 00000000G   00  9E 00009        MOVAB   DBG$NPARSE_EXPRESSION, R10          ;
                            59 00000000G   00  9E 00010        MOVAB   DBG$NMATCH, R9                      ;
                            58 00000000'   EF  9E 00017        MOVAB   DBG$CS_CR, R8                       ;
                                           04  DD 0001E        PUSHL   #4                                  ; 0943
                            6B             01  FB 00020        CALLS   #1, DBG$GET_TEMPMEM                 ;
                            52             50  D0 00023        MOVL    R0, NOUN_NODE                       ;
                            50       08    AC  D0 00026        MOVL    VERB_NODE, R0                       ; 0944
                        08  A0             52  D0 0002A        MOVL    NOUN_NODE, 8(R0)                    ;
                            54       0C    AC  D0 0002E        MOVL    MESSAGE_VECT, R4                    ; 0953
                                           14  BB 00032        PUSHR   #^M<R2,R4>                          ; 0952
                            53       04    AC  D0 00034        MOVL    INPUT_DESC, R3                      ; 0951
                                           53  DD 00038        PUSHL   R3                                  ; 0952
                    00000000G   00         03  FB 0003A        CALLS   #3, DBG$NREAD_NAME                  ;
                            03             50  E8 00041        BLBS    R0, 2$                              ;
                                       00FD 31  00044 1$:      BRW     13$                                 ;
                                           01  DD 00047 2$:    PUSHL   #1                                  ; 0959
                                       02  A8  9F 00049        PUSHAB  DBG$CS_EQUAL                        ;
                                           53  DD 0004C        PUSHL   R3                                  ;
                            69             03  FB 0004E        CALLS   #3, DBG$NMATCH                      ;
                            3D             50  E9 00051        BLBC    R0, 3$                              ;
                            57       08    A2  9E 00054        MOVAB   8(R2), LINK                         ; 0965
```

```
                      04  DD 00058        PUSHL   #4                                ; 0966
 6B                   01  FB 0005A        CALLS   #1, DBG$GET_TEMPMEM               ;
 52                   50  D0 0005D        MOVL    R0, NOUN_NODE                     ;
 67                   52  D0 00060        MOVL    NOUN_NODE, (LINK)                 ; 0967
 56 00000000G         00  9A 00063        MOVZBL  DBG$GB_RADIX, RADIX               ; 0971
                      54  DD 0006A        PUSHL   R4                                ; 0978
                      0D  DD 0006C        PUSHL   #13                               ; 0977
                      52  DD 0006E        PUSHL   NOUN_NODE                         ;
            0048      8F  BB 00070        PUSHR   #^M<R3,R6>                        ;
 6A                   05  FB 00074        CALLS   #5, DBG$NPARSE_EXPRESSION         ;
 55                   50  D0 00077        MOVL    R0, STATUS                        ;
 01                   55  D1 0007A        CMPL    STATUS, #1                        ; 0986
                      1E  13 0007D        BEQL    4$                                ;
 04                   55  D1 0007F        CMPL    STATUS, #4                        ; 0995
                      C0  13 00082        BEQL    1$                                ;
                      02  DD 00084        PUSHL   #2                                ; 1001
              0C      A8  9F 00086        PUSHAB  DBG$CS_TO                         ;
                      53  DD 00089        PUSHL   R3                                ;
 69                   03  FB 0008B        CALLS   #3, DBG$NMATCH                    ;
 12                   50  E8 0008E        BLBS    R0, 6$                            ;
                      01  DD 00091 3$:    PUSHL   #1                                ; 1002
            0108      8F  BB 00093        PUSHR   #^M<R3,R8>                        ;
 69                   03  FB 00097        CALLS   #3, DBG$NMATCH                    ;
 03                   50  E9 0009A        BLBC    R0, 5$                            ;
               0080   31 0009D 4$:        BRW     10$                               ;
               008C   31 000A0 5$:        BRW     11$                               ;
                      54  DD 000A3 6$:    PUSHL   R4                                ; 1010
                      0E  DD 000A5        PUSHL   #14                               ; 1009
              0C      A2  9F 000A7        PUSHAB  12(NOUN_NODE)                     ;
            0048      8F  BB 000AA        PUSHR   #^M<R3,R6>                        ;
 6A                   05  FB 000AE        CALLS   #5, DBG$NPARSE_EXPRESSION         ;
 55                   50  D0 000B1        MOVL    R0, STATUS                        ;
 01                   55  D1 000B4        CMPL    STATUS, #1                        ; 1019
                      67  13 000B7        BEQL    10$                               ;
 04                   55  D1 000B9        CMPL    STATUS, #4                        ; 1028
                      86  13 000BC        BEQL    1$                                ;
                      02  DD 000BE        PUSHL   #2                                ; 1034
              06      A8  9F 000C0        PUSHAB  DBG$CS_BY                         ;
                      53  DD 000C3        PUSHL   R3                                ;
 69                   03  FB 000C5        CALLS   #3, DBG$NMATCH                    ;
 1D                   50  E9 000C8        BLBC    R0, 7$                            ;
                      54  DD 000CB        PUSHL   R4                                ; 1042
                      05  DD 000CD        PUSHL   #5                                ; 1041
              04      A2  9F 000CF        PUSHAB  4(NOUN_NODE)                      ;
            0048      8F  BB 000D2        PUSHR   #^M<R3,R6>                        ;
 6A                   05  FB 000D6        CALLS   #5, DBG$NPARSE_EXPRESSION         ;
 55                   50  D0 000D9        MOVL    R0, STATUS                        ;
 01                   55  D1 000DC        CMPL    STATUS, #1                        ; 1050
                      3F  13 000DF        BEQL    10$                               ;
 04                   55  D1 000E1        CMPL    STATUS, #4                        ; 1059
                      05  12 000E4        BNEQ    8$                                ;
                      5C  11 000E6        BRB     13$                               ; 1061
              04      A2  D4 000E8 7$:    CLRL    4(NOUN_NODE)                      ; 1066
                      02  DD 000EB 8$:    PUSHL   #2                                ; 1070
              09      A8  9F 000ED        PUSHAB  DBG$CS_DO                         ;
                      53  DD 000F0        PUSHL   R3                                ;
 69                   03  FB 000F2        CALLS   #3, DBG$NMATCH                    ;
```

```
                    1C  50  E9 000F5          BLBC    R0, 9$                                  ;
                    57  08  A2 9E 000F8          MOVAB   8(R2), LINK                             ; 1076
                        04  DD 000FC          PUSHL   #4                                      ; 1077
                    6B      01  FB 000FE          CALLS   #1, DBG$GET_TEMPMEM                     ;
                    52      50  D0 00101          MOVL    R0, NOUN_NODE                           ;
                    67      52  D0 00104          MOVL    NOUN_NODE, (LINK)                       ; 1078
                        01  DD 00107          PUSHL   #1                                      ; 1082
                            04  A8 9F 00109          PUSHAB  DBG$CS_LEFT_PAREN                       ;
                        53  DD 0010C          PUSHL   R3                                      ;
                    69      03  FB 0010E          CALLS   #3, DBG$NMATCH                          ;
                    34      50  E8 00111          BLBS    R0, 14$                                 ;
                        01  DD 00114  9$:      PUSHL   #1                                      ; 1083
                          0108  8F  BB 00116          PUSHR   #^M<R3,R8>                              ;
                    69      03  FB 0011A          CALLS   #3, DBG$NMATCH                          ;
                    0F      50  E9 0011D          BLBC    R0, 11$                                 ;
                 000280D0   8F  DD 00120  10$:      PUSHL   #164048                                 ;
00000000G   00      01  FB 00126          CALLS   #1, DBG$NMAKE_ARG_VECT                  ;
                        12  11 0012D          BRB     12$                                     ;
                        53  DD 0012F  11$:      PUSHL   R3                                      ;
00000000G   00      01  FB 00131          CALLS   #1, DBG$NNEXT_WORD                      ;
                        50  DD 00138          PUSHL   R0                                      ;
00000000G   00      01  FB 0013A          CALLS   #1, DBG$NSYNTAX_ERROR                   ;
                    64      50  D0 00141  12$:      MOVL    R0, (R4)                                ;
                    50      04  D0 00144  13$:      MOVL    #4, R0                                  ;
                            04 00147          RET                                             ;
                        52  DD 00148  14$:      PUSHL   NOUN_NODE                               ; 1094
                        53  DD 0014A          PUSHL   R3                                      ;
00000000G   00      02  FB 0014C          CALLS   #2, DBG$NSAVE_BREAK_BUFFER              ;
                    50      01  D0 00153          MOVL    #1, R0                                  ; 1098
                            04 00156          RET                                             ; 1100

; Routine Size:  343 bytes,    Routine Base:  DBG$CODE + 0315
```

```
;  975          1101  1 GLOBAL ROUTINE dbg$nexecute_for (verb_node,message_vect) =
;  976          1102  1 !++
;  977          1103  1 ! Functional Description
;  978          1104  1 !
;  979          1105  1 !       This routine performs the action associated with the FOR
;  980          1106  1 !       command.
;  981          1107  1 !
;  982          1108  1 ! Formal Parameters
;  983          1109  1 !
;  984          1110  1 !       verb_node       - A longword containing the address of the
;  985          1111  1 !                         head (verb) node.
;  986          1112  1 !       message_vect    - The address of a longword to contain the
;  987          1113  1 !                         address of an error message vector
;  988          1114  1 !
;  989          1115  1 ! Implicit Inputs
;  990          1116  1 !
;  991          1117  1 !       The command tree contains a verb node and a linked list
;  992          1118  1 !       of three noun nodes. (See the diagram in the header for
;  993          1119  1 !       DBG$NPARSE_FOR).
;  994          1120  1 !
;  995          1121  1 ! Routine Value
;  996          1122  1 !
;  997          1123  1 !       A completion code.
;  998          1124  1 !
;  999          1125  1 ! Completion Codes
; 1000          1126  1 !
; 1001          1127  1 !       sts$k_success (1)       - Success. Command executed
; 1002          1128  1 !       sts$k_severe (4)        - Failure. The command could not be
; 1003          1129  1 !                                 executed. An error message is constructed.
; 1004          1130  1 !
; 1005          1131  1 ! Side Effects
; 1006          1132  1 !
; 1007          1133  1 !       None
; 1008          1134  1 !--
; 1009          1135  2     BEGIN
; 1010          1136  2
; 1011          1137  2     MAP
; 1012          1138  2         verb_node : REF dbg$verb_node;
; 1013          1139  2
; 1014          1140  2     LOCAL
; 1015          1141  2         action_node:    REF dbg$noun_node,      ! The noun node for the action clause
; 1016          1142  2         action_string:  REF VECTOR[,WORD],      ! Counted string with the action clause
; 1017          1143  2         bounds_node: REF dbg$noun_node,         ! Noun node with the upper and
; 1018          1144  2                                                 !     lower bounds
; 1019          1145  2         dummy,                                  ! Dummy output parameter
; 1020          1146  2         loop_incr,                              ! Loop increment
; 1021          1147  2         lower_bound,                            ! An integer with the lower
; 1022          1148  2                                                 !     loop bound
; 1023          1149  2         new_valdesc: REF dbg$valdesc,           ! A copy of a value descriptor
; 1024          1150  2         new_varname,                            ! Pointer to a copy of the
; 1025          1151  2                                                 !     variable name
; 1026          1152  2         symid_list,                             ! Points to a list of symids
; 1027          1153  2         upper_bound,                            ! An integer with the upper
; 1028          1154  2                                                 !     loop bound
; 1029          1155  2         valdesc: REF dbg$valdesc,               ! A pointer to a value
; 1030          1156  2                                                 !     descriptor
; 1031          1157  2         var_node: REF dbg$noun_node,            ! The noun node with the loop
```

```
;  1032          1158  2                                    !       variable
;  1033          1159  2          var_name: REF VECTOR[,BYTE],  ! The counted string with the
;  1034          1160  2                                    !       variable name
;  1035          1161  2          vax_desc:       dbg$stg_desc; ! Target of the conversion from
;  1036          1162  2                                    !       the value descriptor
;  1037          1163  2                                    !       representing the count.
;  1038          1164  2      ! Recover the noun nodes.
;  1039          1165  2      !
;  1040          1166  2      var_node = .verb_node [dbg$l_verb_object_ptr];
;  1041          1167  2      var_name = .var_node [dbg$l_noun_value];
;  1042          1168  2      bounds_node = .var_node [dbg$l_noun_link];
;  1043          1169  2      valdesc = .bounds_node [dbg$l_noun_value];
;  1044          1170  2      action_node = .bounds_node [dbg$l_noun_link];
;  1045          1171  2      action_string = .action_node [dbg$l_noun_value];
;  1046          1172  2
;  1047          1173  2      ! Set up the vax descriptor for the bounds.
;  1048          1174  2      ! This vax descriptor is of type integer longword, and is used to convert the
;  1049          1175  2      ! language specific value descriptor for loop bounds to an
;  1050          1176  2      ! integer quantity that we can use in a language-independent way.
;  1051          1177  2      !
;  1052          1178  2      vax_desc [dsc$b_class] = dsc$k_class_s;
;  1053          1179  2      vax_desc [dsc$b_dtype] = dsc$k_dtype_l;
;  1054          1180  2      vax_desc [dsc$w_length] = 4;
;  1055          1181  2      vax_desc [dsc$a_pointer] = lower_bound;
;  1056          1182  2
;  1057          1183  2      ! Do the conversion from value descriptor to integer.
;  1058          1184  2      !
;  1059          1185  2      IF NOT dbg$ntype_conv (.valdesc,
;  1060          1186  2                             dbg$k_default,
;  1061          1187  2                             dbg$k_vax_desc,
;  1062          1188  2                             vax_desc,
;  1063          1189  2                             .message_vect)
;  1064          1190  2      THEN
;  1065          1191  2          RETURN sts$k_severe;
;  1066          1192  2
;  1067          1193  2      ! Do the conversion again, this time picking up the upper bound.
;  1068          1194  2      !
;  1069          1195  2      vax_desc [dsc$a_pointer] = upper_bound;
;  1070          1196  2      IF NOT dbg$ntype_conv (.bounds_node [dbg$l_noun_value2],
;  1071          1197  2                             dbg$k_default,
;  1072          1198  2                             dbg$k_vax_desc,
;  1073          1199  2                             vax_desc,
;  1074          1200  2                             .message_vect)
;  1075          1201  2      THEN
;  1076          1202  2          RETURN sts$k_severe;
;  1077          1203  2
;  1078          1204  2      ! Do the conversion once again, this time with the loop increment.
;  1079          1205  2      !
;  1080          1206  2      IF .bounds_node [dbg$l_adjective_ptr] EQL 0
;  1081          1207  2      THEN
;  1082          1208  2          loop_incr = 1
;  1083          1209  2      ELSE
;  1084          1210  3          BEGIN
;  1085          1211  3          vax_desc [dsc$a_pointer] = loop_incr;
;  1086          1212  3          IF NOT dbg$ntype_conv (.bounds_node [dbg$l_adjective_ptr],
;  1087          1213  3                                 dbg$k_default,
;  1088          1214  3                                 dbg$k_vax_desc,
```

```
;  1089          1215  3                                  vax_desc,
;  1090          1216  3                                  .message_vect)
;  1091          1217  3              THEN
;  1092          1218  3                  RETURN sts$k_severe;
;  1093          1219  3              END;
;  1094          1220  2
;  1095          1221  2          ! If the loop increment is zero then signal an error.
;  1096          1222  2          !
;  1097          1223  2          IF .loop_incr EQL 0
;  1098          1224  2          THEN
;  1099          1225  2              SIGNAL (dbg$_loopincr);
;  1100          1226  2
;  1101          1227  2          ! If the upper bound is below the lower bound, do nothing.
;  1102          1228  2          !
;  1103          1229  2          IF (.loop_incr GTR 0 AND .upper_bound LSS .lower_bound)
;  1104          1230  2          OR (.loop_incr LSS 0 AND .upper_bound GTR .lower_bound)
;  1105          1231  2          THEN
;  1106          1232  2              RETURN sts$k_success;
;  1107          1233  2
;  1108          1234  2          ! Make a value descriptor for the initial value of the loop variable.
;  1109          1235  2          !
;  1110          1236  2          new_valdesc = dbg$get_memory (dbg$k_valdesc_base_size+4);
;  1111          1237  2          new_valdesc[dbg$w_dhdr_length] = (dbg$k_valdesc_base_size * 4) + 16;
;  1112          1238  2          new_valdesc[dbg$b_dhdr_type] = dbg$k_value_desc;
;  1113          1239  2          new_valdesc[dbg$b_dhdr_lang] = .dbg$gb_language;
;  1114          1240  2          new_valdesc[dbg$b_dhdr_kind] = rst$k_data;
;  1115          1241  2          new_valdesc[dbg$b_dhdr_fcode] = rst$k_type_atomic;
;  1116          1242  2          new_valdesc[dbg$b_value_class] = dsc$k_class_s;
;  1117          1243  2          new_valdesc[dbg$b_value_dtype] = dsc$k_dtype_l;
;  1118          1244  2          new_valdesc[dbg$w_value_length] = 4;
;  1119          1245  2          new_valdesc[dbg$l_value_pointer] = new_valdesc[dbg$l_value_value0];
;  1120          1246  2          new_valdesc[dbg$l_value_value0] = .lower_bound;
;  1121          1247  2
;  1122          1248  2          ! Also make a copy of the variable name. This is because the original
;  1123          1249  2          ! varname pointer is being saved away by dbg$ncis_add and we don't
;  1124          1250  2          ! want to free it twice.
;  1125          1251  2          !
;  1126          1252  2          new_varname = dbg$get_memory (1+.var_name[0]/4);
;  1127          1253  2          ch$move (1+.var_name[0], .var_name, .new_varname);
;  1128          1254  2          IF NOT dbg$def_sym_add (.new_varname, define_value,
;  1129          1255  2                                  .new_valdesc,
;  1130          1256  2                                  FALSE, dummy, .message_vect)
;  1131          1257  2          THEN
;  1132          1258  2              RETURN sts$k_severe;
;  1133          1259  2
;  1134          1260  2          ! Add a link to the command input stream, containing the action
;  1135          1261  2          ! string and the upper bound.
;  1136          1262  2          !
;  1137          1263  2          IF NOT dbg$ncis_add (action_string[1], .action_string[0], cis_for,
;  1138          1264  2                               .upper_bound, .var_name, .loop_incr, .message_vect)
;  1139          1265  2          THEN
;  1140          1266  2              RETURN sts$k_severe;
;  1141          1267  2
;  1142          1268  2          ! Return success.
;  1143          1269  2          !
;  1144          1270  2          RETURN sts$k_success;
;  1145          1271  2
```

```
; 1146          1272  1     END; ! dbg$nexecute_repeat
; INFO#250                  L1:1229
; Referenced LOCAL symbol UPPER_BOUND is probably not initialized
; INFO#250                  L1:1229
; Referenced LOCAL symbol LOWER_BOUND is probably not initialized


                                                0FFC 00000          .ENTRY  DBG$NEXECUTE_FOR, Save R2,R3,R4,R5,R6,R7,-      ; 1101
                                                                            R8,R9,R10,R11
                                          5E        1C  C2 00002          SUBL2   #28, SP
                                          50    04  AC  D0 00005          MOVL    VERB_NODE, R0                               ; 1166
                                          50    08  A0  D0 00009          MOVL    8(R0), VAR_NODE
                                          56        60  D0 0000D          MOVL    (VAR_NODE), VAR_NAME                        ; 1167
                                          52    08  A0  D0 00010          MOVL    8(VAR_NODE), BOUNDS_NODE                    ; 1168
                                          51        62  D0 00014          MOVL    (BOUNDS_NODE), VALDESC                      ; 1169
                                          50    08  A2  D0 00017          MOVL    8(BOUNDS_NODE), ACTION_NODE                 ; 1170
                                          5B        60  D0 0001B          MOVL    (ACTION_NODE), ACTION_STRING                ; 1171
                                    10    AE  01080004  8F  D0 0001E      MOVL    #17301508, VAX_DESC                         ; 1180
                                    14    AE            6E  9E 00026      MOVAB   LOWER_BOUND, VAX_DESC+4                     ; 1181
                                          58    08  AC  D0 0002A          MOVL    MESSAGE_VECT, R8                            ; 1189
                                                    58  DD 0002E          PUSHL   R8
                                                14  AE  9F 00030          PUSHAB  VAX_DESC                                    ; 1185
                                          7E        82  8F  9A 00033      MOVZBL  #130, -(SP)
                                                    01  DD 00037          PUSHL   #1
                                                    51  DD 00039          PUSHL   VALDESC
                                00000000G   00            05  FB 0003B  CALLS   #5, DBG$NTYPE_CONV
                                          42            50  E9 00042      BLBC    R0, 2$
                                    14    AE      04    AE  9E 00045      MOVAB   UPPER_BOUND, VAX_DESC+4                     ; 1195
                                                    58  DD 0004A          PUSHL   R8                                          ; 1200
                                                14  AE  9F 0004C          PUSHAB  VAX_DESC                                    ; 1196
                                          7E        82  8F  9A 0004F      MOVZBL  #130, -(SP)
                                                    01  DD 00053          PUSHL   #1
                                                0C  A2  DD 00055          PUSHL   12(BOUNDS_NODE)
                                00000000G   00            05  FB 00058  CALLS   #5, DBG$NTYPE_CONV
                                          25            50  E9 0005F      BLBC    R0, 2$
                                                04  A2  D5 00062          TSTL    4(BOUNDS_NODE)                              ; 1206
                                                    06  12 00065          BNEQ    1$
                                    08    AE            01  D0 00067      MOVL    #1, LOOP_INCR                               ; 1208
                                                    20  11 0006B          BRB     3$
                                    14    AE      08    AE  9E 0006D 1$:  MOVAB   LOOP_INCR, VAX_DESC+4                     ; 1211
                                                    58  DD 00072          PUSHL   R8                                          ; 1216
                                                14  AE  9F 00074          PUSHAB  VAX_DESC                                    ; 1212
                                          7E        82  8F  9A 00077      MOVZBL  #130, -(SP)
                                                    01  DD 0007B          PUSHL   #1
                                                04  A2  DD 0007D          PUSHL   4(BOUNDS_NODE)
                                00000000G   00            05  FB 00080  CALLS   #5, DBG$NTYPE_CONV
                                          03            50  E8 00087 2$:  BLBS    R0, 3$
                                                00AB        31  00008A    BRW     8$
                                          59        08  AE  D0 0008D 3$:  MOVL    LOOP_INCR, R9                               ; 1223
                                                    0D  12 00091          BNEQ    4$
                                              00028F18  8F  DD 00093      PUSHL   #167704                                     ; 1225
                                00000000G   00            01  FB 00099  CALLS   #1, LIB$SIGNAL
                                                    59  D5 000A0 4$:      TSTL    R9                                          ; 1229
                                                    09  15 000A2          BLEQ    6$
```

```
                        6E      04    AE  D1 000A4          CMPL    UPPER_BOUND, LOWER_BOUND                                    ;
                                      03  18 000A8          BGEQ    6$                                                          ;
                                  008F  31 000AA 5$:        BRW     9$                                                          ;
                                      59  D5 000AD 6$:      TSTL    R9                                                          ; 1230
                                      06  18 000AF          BGEQ    7$                                                          ;
                        6E      04    AE  D1 000B1          CMPL    UPPER_BOUND, LOWER_BOUND                                    ;
                                      F3  14 000B5          BGTR    5$                                                          ;
                                      0C  DD 000B7 7$:      PUSHL   #12                                                         ; 1236
              00000000G 00            01  FB 000B9          CALLS   #1, DBG$GET_MEMORY                                          ;
                        57            50  D0 000C0          MOVL    R0, NEW_VALDESC                                             ;
                        67            30  B0 000C3          MOVW    #48, (NEW_VALDESC)                                          ; 1237
                    02  A7      7A   8F  90 000C6          MOVB    #122, 2(NEW_VALDESC)                                        ; 1238
                    03  A7 00000000G 00  90 000CB          MOVB    DBG$GB_LANGUAGE, 3(NEW_VALDESC)                             ; 1239
                    06  A7      0602  8F  B0 000D3          MOVW    #1538, 6(NEW_VALDESC)                                       ; 1241
                    14  A7 01080004   8F  D0 000D9          MOVL    #17301508, 20(NEW_VALDESC)                                  ; 1244
                    18  A7      20    A7  9E 000E1          MOVAB   32(NEW_VALDESC), 24(NEW_VALDESC)                            ; 1245
                    20  A7            6E  D0 000E6          MOVL    LOWER_BOUND, 32(NEW_VALDESC)                                ; 1246
                        50            66  9A 000EA          MOVZBL  (VAR_NAME), R0                                              ; 1252
                        50            04  C6 000ED          DIVL2   #4, R0                                                      ;
                                  01  A0  9F 000F0          PUSHAB  1(R0)                                                       ;
              00000000G 00            01  FB 000F3          CALLS   #1, DBG$GET_MEMORY                                          ;
                        5A            50  D0 000FA          MOVL    R0, NEW_VARNAME                                             ;
                        50            66  9A 000FD          MOVZBL  (VAR_NAME), R0                                              ; 1253
                                      50  D6 00100          INCL    R0                                                          ;
                  6A                  66  28 00102          MOVC3   R0, (VAR_NAME), (NEW_VARNAME)                               ;
                                      58  DD 00106          PUSHL   R8                                                          ; 1256
                                  10  AE  9F 00108          PUSHAB  DUMMY                                                       ; 1254
                                      7E  D4 0010B          CLRL    -(SP)                                                       ;
                                      57  DD 0010D          PUSHL   NEW_VALDESC                                                 ; 1255
                                      05  DD 0010F          PUSHL   #5                                                          ; 1254
                                      5A  DD 00111          PUSHL   NEW_VARNAME                                                 ;
              00000000G 00            06  FB 00113          CALLS   #6, DBG$DEF_SYM_ADD                                         ;
                        1B            50  E9 0011A          BLBC    R0, 8$                                                      ;
                                      58  DD 0011D          PUSHL   R8                                                          ; 1264
                                0240  8F  BB 0011F          PUSHR   #^M<R6,R9>                                                  ;
                                  10  AE  DD 00123          PUSHL   UPPER_BOUND                                                 ;
                                      07  DD 00126          PUSHL   #7                                                          ; 1263
                        7E            6B  3C 00128          MOVZWL  (ACTION_STRING), -(SP)                                      ;
                                  02  AB  9F 0012B          PUSHAB  2(ACTION_STRING)                                            ;
              00000000G 00            07  FB 0012E          CALLS   #7, DBG$NCIS_ADD                                            ;
                        04            50  E8 00135          BLBS    R0, 9$                                                      ;
                        50            04  D0 00138 8$:      MOVL    #4, R0                                                      ; 1266
                                      04  00013B           RET                                                                 ;
                        50            01  D0 0013C 9$:      MOVL    #1, R0                                                      ; 1270
                                      04  0013F            RET                                                                 ; 1272

; Routine Size:  320 bytes,    Routine Base:  DBG$CODE + 046C
```

```
;  1148          1273  1 GLOBAL ROUTINE dbg$nparse_repeat(input_desc, verb_node, message_vect) =
;  1149          1274  1 !
;  1150          1275  1 ! Functional Description
;  1151          1276  1 !
;  1152          1277  1 !       ATN parse network for the REPEAT verb.
;  1153          1278  1 !       This routine takes a verb node for the REPEAT verb, and a string
;  1154          1279  1 !       descriptor for the remaining (unparsed) input.
;  1155          1280  1 !       A command execution tree is built. The form of the tree is:
;  1156          1281  1 !
;  1157          1282  1 !        -----------       -----------       -----------
;  1158          1283  1 !       | verb node |-->--| noun node |-->--| noun node |
;  1159          1284  1 !        -----------       -----------       -----------
;  1160          1285  1 !
;  1161          1286  1 !       The first noun node points to a value descriptor for the count.
;  1162          1287  1 !       The second noun node points to a counted string with the action clause.
;  1163          1288  1 !
;  1164          1289  1 ! Formal Parameters
;  1165          1290  1 !
;  1166          1291  1 !       input_desc      - A longword containing the address of the
;  1167          1292  1 !                         command input descriptor.
;  1168          1293  1 !       verb_node       - A longword containing the address of the verb node.
;  1169          1294  1 !       message_vect    - The address of a longword to contain the address
;  1170          1295  1 !                         of a standard message argument vector.
;  1171          1296  1 !
;  1172          1297  1 ! Implicit Inputs
;  1173          1298  1 !
;  1174          1299  1 !       none
;  1175          1300  1 !
;  1176          1301  1 ! Implicit Outputs
;  1177          1302  1 !
;  1178          1303  1 !       On success, the command execution tree is constructed.
;  1179          1304  1 !       On failure, a message argument vector is constructed or obtained.
;  1180          1305  1 !
;  1181          1306  1 ! Routine value
;  1182          1307  1 !
;  1183          1308  1 !       sts$k_success (1)       - Success. Command execution tree constructed.
;  1184          1309  1 !       sts$k_severe  (4)       - Failure. Error encountered. Message argument
;  1185          1310  1 !                                 constructed and returned.
;  1186          1311  1 !
;  1187          1312  1 ! Side Effects
;  1188          1313  1 !
;  1189          1314  1 !       Permanent storage is allocated for the string holding the action
;  1190          1315  1 !       clause; this is released in DBG$NEXECUTE_REPEAT after execution
;  1191          1316  1 !       of the action clause.
;  1192          1317  1 !
;  1193          1318  1
;  1194          1319  2     BEGIN
;  1195          1320  2
;  1196          1321  2     MAP
;  1197          1322  2         input_desc  : REF dbg$stg_desc,
;  1198          1323  2         verb_node       : REF dbg$verb_node;
;  1199          1324  2
;  1200          1325  2     BIND
;  1201          1326  2         dbg$cs_cr                   = UPLIT BYTE (1, dbg$k_car_return),
;  1202          1327  2         dbg$cs_left_paren           = UPLIT BYTE (1, dbg$k_left_parenthesis),
;  1203          1328  2         dbg$cs_do                   = UPLIT BYTE (2, 'DO');
;  1204          1329  2
```

```
;  1205          1330  2      LOCAL
;  1206          1331  2          link,                               ! Temporary to hold links in the command
;  1207          1332  2                                              !     execution tree.
;  1208          1333  2          noun_node : REF dbg$noun_node,      ! A node in the command execution tree.
;  1209          1334  2          radix,                              ! Holds the current radix setting.
;  1210          1335  2          status;                             ! Holds return status from subroutine
;  1211          1336  2                                              !     calls.
;  1212          1337  2
;  1213          1338  2
;  1214          1339  2      ! Create and link a noun node
;  1215          1340  2      !
;  1216          1341  2      noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  1217          1342  2      verb_node[dbg$l_verb_object_ptr] = .noun_node;
;  1218          1343  2
;  1219          1344  2      ! Determine the current radix.
;  1220          1345  2      !
;  1221          1346  2      radix  = .dbg$gb_radix[dbg$b_radix_input];
;  1222          1347  2
;  1223          1348  2      ! Obtain a value descriptor for the count. The first noun node
;  1224          1349  2      ! points to this descriptor.
;  1225          1350  2      !
;  1226          1351  2      STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
;  1227          1352  2                                     NOUN_NODE [DBG$L_NOUN_VALUE],
;  1228          1353  2                                     TOKEN$K_TERM_DO, .MESSAGE_VECT);
;  1229          1354  2
;  1230          1355  2
;  1231          1356  2      ! The return status should be "warning", meaning that an expression
;  1232          1357  2      ! was parsed and further input reamins. If an expression was parsed
;  1233          1358  2      ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
;  1234          1359  2      ! In this context, it is an error since "REPEAT count" by itself
;  1235          1360  2      ! is an error.
;  1236          1361  2      !
;  1237          1362  2      IF .status EQL sts$k_success
;  1238          1363  2      THEN
;  1239          1364  3          BEGIN
;  1240          1365  3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  1241          1366  3          RETURN sts$k_severe;
;  1242          1367  2          END;
;  1243          1368  2
;  1244          1369  2      ! Severe status is also an error.
;  1245          1370  2      !
;  1246          1371  2      IF .status EQL sts$k_severe
;  1247          1372  2      THEN
;  1248          1373  2          RETURN sts$k_severe;
;  1249          1374  2
;  1250          1375  2      ! Eat the DO.
;  1251          1376  2      !
;  1252          1377  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_do, 1)
;  1253          1378  2      THEN
;  1254          1379  3          BEGIN
;  1255          1380  3          .message_vect =
;  1256          1381  4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  1257          1382  4               THEN
;  1258          1383  4                   dbg$nmake_arg_vect (dbg$_needmore)
;  1259          1384  4               ELSE
;  1260          1385  3                   dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;  1261          1386  3          RETURN sts$k_severe;
```

```
;  1262          1387  2          END;
;  1263          1388  2
;  1264          1389  2      ! Allocate and link a noun node for the action clause.
;  1265          1390  2      !
;  1266          1391  2      link = noun_node [dbg$l_noun_link];
;  1267          1392  2      noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  1268          1393  2      .link = .noun_node;
;  1269          1394  2
;  1270          1395  2      ! Eat the left parenthesis which we require be present.
;  1271          1396  2      !
;  1272          1397  2      IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  1273          1398  2      THEN
;  1274          1399  3          BEGIN
;  1275          1400  3          .message_vect =
;  1276          1401  4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  1277          1402  4               THEN
;  1278          1403  4                   dbg$nmake_arg_vect (dbg$_needmore)
;  1279          1404  4               ELSE
;  1280          1405  5                   BEGIN
;  1281          1406  5                   SIGNAL(dbg$_needparen);
;  1282          1407  5                   dbg$nsyntax_error (dbg$nnext_word (.input_desc))
;  1283          1408  3                   END);
;  1284          1409  3          RETURN sts$k_severe;
;  1285          1410  2          END;
;  1286          1411  2
;  1287          1412  2      ! Put a pointer to the counted string representing the action
;  1288          1413  2      ! clause into the second noun node. (Note - the counted string
;  1289          1414  2      ! is constructed out of "permanent" memory which is released
;  1290          1415  2      ! in DBG$NEXECUTE_REPEAT).
;  1291          1416  2      ! The third argument indicates that save_break_buffer is not being
;  1292          1417  2      ! called during parsing of a SET BREAK DO (The routine behaves
;  1293          1418  2      ! slightly differently in that case)
;  1294          1419  2      !
;  1295          1420  2      dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
;  1296          1421  2
;  1297          1422  2      ! Return success.
;  1298          1423  2      !
;  1299          1424  2      RETURN sts$k_success;
;  1300          1425  2
;  1301          1426  1      END;


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

                              0D  01  00026 P.AAO:  .BYTE   1, 13                    ;
                              28  01  00028 P.AAP:  .BYTE   1, 40                    ;
                                  02  0002A P.AAQ:  .BYTE   2                        ;
                              4F  44  0002B         .ASCII  \DO\                     ;

                                        DBG$CS_CR=          P.AAO
                                        DBG$CS_LEFT_PAREN=  P.AAP
                                        DBG$CS_DO=          P.AAQ

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

```
                                00FC 00000          .ENTRY  DBG$NPARSE_REPEAT, Save R2,R3,R4,R5,R6,R7    ; 1273
                    57 00000000G  00   9E 00002     MOVAB   DBG$GET_TEMPMEM, R7                          ;
                    56 00000000G  00   9E 00009     MOVAB   DBG$NMATCH, R6                               ;
                    55 00000000'  EF   9E 00010     MOVAB   DBG$CS_CR, R5                                ;
                                  04   DD 00017     PUSHL   #4                                           ; 1341
                    67            01   FB 00019     CALLS   #1, DBG$GET_TEMPMEM                          ;
                    54            50   D0 0001C     MOVL    R0, NOUN_NODE                                ;
                    50      08    AC   D0 0001F     MOVL    VERB_NODE, R0                                ; 1342
              08    A0            54   D0 00023     MOVL    NOUN_NODE, 8(R0)                             ;
                    50 00000000G  00   9A 00027     MOVZBL  DBG$GB_RADIX, RADIX                          ; 1346
                           0C     AC   DD 0002E     PUSHL   MESSAGE_VECT                                 ; 1353
                                  05   DD 00031     PUSHL   #5                                           ; 1352
                                  11   BB 00033     PUSHR   #^M<R0,R4>                                   ;
                    52      04    AC   D0 00035     MOVL    INPUT_DESC, R2                               ; 1351
                                  52   DD 00039     PUSHL   R2                                           ; 1352
 00000000G          00            05   FB 0003B     CALLS   #5, DBG$NPARSE_EXPRESSION                    ;
                    53            50   D0 00042     MOVL    R0, STATUS                                   ;
                    01            53   D1 00045     CMPL    STATUS, #1                                   ; 1362
                                  44   13 00048     BEQL    2$                                           ;
                    04            53   D1 0004A     CMPL    STATUS, #4                                   ; 1371
                                  71   13 0004D     BEQL    6$                                           ;
                                  01   DD 0004F     PUSHL   #1                                           ; 1377
                           04     A5   9F 00051     PUSHAB  DBG$CS_DO                                    ;
                                  52   DD 00054     PUSHL   R2                                           ;
                    66            03   FB 00056     CALLS   #3, DBG$NMATCH                               ;
                    0C            50   E8 00059     BLBS    R0, 1$                                       ;
                                  01   DD 0005C     PUSHL   #1                                           ; 1381
                                  24   BB 0005E     PUSHR   #^M<R2,R5>                                   ;
                    66            03   FB 00060     CALLS   #3, DBG$NMATCH                               ;
                    44            50   E9 00063     BLBC    R0, 4$                                       ;
                                  26   11 00066     BRB     2$                                           ; 1383
                    53      08    A4   9E 00068 1$:  MOVAB   8(R4), LINK                                 ; 1391
                                  04   DD 0006C     PUSHL   #4                                           ; 1392
                    67            01   FB 0006E     CALLS   #1, DBG$GET_TEMPMEM                          ;
                    54            50   D0 00071     MOVL    R0, NOUN_NODE                                ;
                    63            54   D0 00074     MOVL    NOUN_NODE, (LINK)                            ; 1393
                                  01   DD 00077     PUSHL   #1                                           ; 1397
                           02     A5   9F 00079     PUSHAB  DBG$CS_LEFT_PAREN                            ;
                                  52   DD 0007C     PUSHL   R2                                           ;
                    66            03   FB 0007E     CALLS   #3, DBG$NMATCH                               ;
                    40            50   E8 00081     BLBS    R0, 7$                                       ;
                                  01   DD 00084     PUSHL   #1                                           ; 1401
                                  24   BB 00086     PUSHR   #^M<R2,R5>                                   ;
                    66            03   FB 00088     CALLS   #3, DBG$NMATCH                               ;
                    0F            50   E9 0008B     BLBC    R0, 3$                                       ;
                          000280D0  8F   DD 0008E 2$:  PUSHL   #164048                                      ; 1403
 00000000G          00            01   FB 00094     CALLS   #1, DBG$NMAKE_ARG_VECT                       ;
                                  1F   11 0009B     BRB     5$                                           ;
                          00028743  8F   DD 0009D 3$:  PUSHL   #165699                                      ; 1406
 00000000G          00            01   FB 000A3     CALLS   #1, LIB$SIGNAL                               ;
                                  52   DD 000AA 4$:  PUSHL   R2                                           ; 1407
 00000000G          00            01   FB 000AC     CALLS   #1, DBG$NNEXT_WORD                           ;
                                  50   DD 000B3     PUSHL   R0                                           ;
 00000000G          00            01   FB 000B5     CALLS   #1, DBG$NSYNTAX_ERROR                        ;
                    0C      BC    50   D0 000BC 5$:  MOVL    R0, @MESSAGE_VECT                            ; 1401
                    50            04   D0 000C0 6$:  MOVL    #4, R0                                       ; 1409
                                  04 000C3          RET                                                  ;
```

```
                                 14  BB 000C4 7$:     PUSHR   #^M<R2,R4>                              ; 1420
                                 02  FB 000C6        CALLS   #2, DBG$NSAVE_BREAK_BUFFER              ;
                00000000G   00   01  D0 000CD        MOVL    #1, R0                                  ; 1424
                            50       04 000D0        RET                                             ; 1426

; Routine Size:  209 bytes,    Routine Base:  DBG$CODE + 05AC
```

```
;  1303     1427  1 GLOBAL ROUTINE dbg$nexecute_repeat (verb_node,message_vect) =
;  1304     1428  1 !++
;  1305     1429  1 ! Functional Description
;  1306     1430  1 !
;  1307     1431  1 !       This routine performs the action associated with the REPEAT
;  1308     1432  1 !       command.
;  1309     1433  1 !
;  1310     1434  1 ! Formal Parameters
;  1311     1435  1 !
;  1312     1436  1 !       verb_node       - A longword containing the address of the
;  1313     1437  1 !                         head (verb) node.
;  1314     1438  1 !       message_vect    - The address of a longword to contain the
;  1315     1439  1 !                         address of an error message vector
;  1316     1440  1 !
;  1317     1441  1 ! Implicit Inputs
;  1318     1442  1 !
;  1319     1443  1 !       The command tree contains a verb node and a linked list
;  1320     1444  1 !       of two noun nodes. (See the diagram in the header for
;  1321     1445  1 !       DBG$NPARSE_REPEAT).
;  1322     1446  1 !
;  1323     1447  1 ! Routine Value
;  1324     1448  1 !
;  1325     1449  1 !       A completion code.
;  1326     1450  1 !
;  1327     1451  1 ! Completion Codes
;  1328     1452  1 !
;  1329     1453  1 !       sts$k_success (1)       - Success. Command executed
;  1330     1454  1 !       sts$k_severe (4)        - Failure. The command could not be
;  1331     1455  1 !                                 executed. An error message is constructed.
;  1332     1456  1 !
;  1333     1457  1 ! Side Effects
;  1334     1458  1 !
;  1335     1459  1 !       None
;  1336     1460  1 !--
;  1337     1461  2     BEGIN
;  1338     1462  2
;  1339     1463  2     MAP
;  1340     1464  2         verb_node : REF dbg$verb_node;
;  1341     1465  2
;  1342     1466  2     LOCAL
;  1343     1467  2         action_node:    REF dbg$noun_node,      ! The noun node for the action clause
;  1344     1468  2         action_string:  REF VECTOR[,WORD],      ! Counted string with the action clause
;  1345     1469  2         count_node: REF dbg$noun_node,          ! The noun node for the count
;  1346     1470  2         count_value,                            ! The actual count
;  1347     1471  2         vax_desc:       dbg$stg_desc;           ! Target of the conversion from
;  1348     1472  2                                                 !    the value descriptor
;  1349     1473  2                                                 !    representing the count.
;  1350     1474  2
;  1351     1475  2     ! Recover the noun nodes.
;  1352     1476  2     !
;  1353     1477  2     count_node = .verb_node [dbg$l_verb_object_ptr];
;  1354     1478  2     action_node = .count_node [dbg$l_noun_link];
;  1355     1479  2
;  1356     1480  2     ! Set up the vax descriptor for the count.
;  1357     1481  2     ! This vax descriptor is of type integer longword, and is used to convert the
;  1358     1482  2     ! language specific value descriptor for a count to an
;  1359     1483  2     ! integer quantity that we can use in a language-independent way.
```

```
; 1360      1484 2        !
; 1361      1485 2        vax_desc [dsc$b_class] = dsc$k_class_s;
; 1362      1486 2        vax_desc [dsc$b_dtype] = dsc$k_dtype_l;
; 1363      1487 2        vax_desc [dsc$w_length] = 4;
; 1364      1488 2        vax_desc [dsc$a_pointer] = count_value;
; 1365      1489 2
; 1366      1490 2        ! Initialize count_value to 0
; 1367      1491 2        !
; 1368      1492 2        count_value = 0;
; 1369      1493 2
; 1370      1494 2        ! Do the conversion from value descriptor to boolean.
; 1371      1495 2        !
; 1372      1496 2        IF NOT dbg$ntype_conv (.count_node [dbg$l_noun_value],
; 1373      1497 2                               dbg$k_default,
; 1374      1498 2                               dbg$k_vax_desc,
; 1375      1499 2                               vax_desc,
; 1376      1500 2                               .message_vect)
; 1377      1501 2        THEN
; 1378      1502 2            RETURN sts$k_severe;
; 1379      1503 2
; 1380      1504 2        ! Recover the string.
; 1381      1505 2        !
; 1382      1506 2        action_string = .action_node [dbg$l_noun_value];
; 1383      1507 2
; 1384      1508 2        ! Add a link to the command input stream, containing the action
; 1385      1509 2        ! string and the repeat count.
; 1386      1510 2        !
; 1387      1511 2        IF NOT dbg$ncis_add (action_string[1], .action_string[0], cis_repeat,
; 1388      1512 2                    .count_value, 0, 0, .message_vect)
; 1389      1513 2        THEN
; 1390      1514 2            RETURN sts$k_severe;
; 1391      1515 2
; 1392      1516 2        ! Return success.
; 1393      1517 2        !
; 1394      1518 2        RETURN sts$k_success;
; 1395      1519 2
; 1396      1520 1        END; ! dbg$nexecute_repeat


                                 0004 00000     .ENTRY  DBG$NEXECUTE_REPEAT, Save R2          ; 1427
             5E              10    C2 00002     SUBL2   #16, SP                               ;
             50        04    AC    D0 00005     MOVL    VERB_NODE, R0                         ; 1477
             50        08    A0    D0 00009     MOVL    8(R0), COUNT_NODE                     ;
             52        08    A0    D0 0000D     MOVL    8(COUNT_NODE), ACTION_NODE            ; 1478
      04     AE  01080004    8F    D0 00011     MOVL    #17301508, VAX_DESC                   ; 1487
      08     AE              6E    9E 00019     MOVAB   COUNT_VALUE, VAX_DESC+4               ; 1488
                             6E    D4 0001D     CLRL    COUNT_VALUE                           ; 1492
                       08    AC    DD 0001F     PUSHL   MESSAGE_VECT                          ; 1500
                       08    AE    9F 00022     PUSHAB  VAX_DESC                              ; 1496
             7E        82    8F    9A 00025     MOVZBL  #130, -(SP)                           ;
                             01    DD 00029     PUSHL   #1                                    ;
                             60    DD 0002B     PUSHL   (COUNT_NODE)                          ;
 00000000G   00              05    FB 0002D     CALLS   #5, DBG$NTYPE_CONV                    ;
             1D              50    E9 00034     BLBC    R0, 1$                                ;
```

```
                                50          62  D0 00037          MOVL    (ACTION_NODE), ACTION_STRING                ; 1506
                                        08  AC  DD 0003A          PUSHL   MESSAGE_VECT                                ; 1512
                                            7E  7C 0003D          CLRQ    -(SP)                                       ; 1511
                                        0C  AE  DD 0003F          PUSHL   COUNT_VALUE                                 ; 1512
                                            04  DD 00042          PUSHL   #4                                          ; 1511
                                7E          60  3C 00044          MOVZWL  (ACTION_STRING), -(SP)                      ;
                                        02  A0  9F 00047          PUSHAB  2(ACTION_STRING)                            ;
                    00000000G   00          07  FB 0004A          CALLS   #7, DBG$NCIS_ADD                            ;
                                04          50  E8 00051          BLBS    R0, 2$                                      ;
                                50          04  D0 00054 1$:      MOVL    #4, R0                                      ; 1514
                                                04 00057          RET                                                 ;
                                50          01  D0 00058 2$:      MOVL    #1, R0                                      ; 1518
                                                04 0005B          RET                                                 ; 1520

; Routine Size:  92 bytes,    Routine Base:  DBG$CODE + 067D

; 1397          1521  1 END
; 1398          1522  0 ELUDOM


                                                                  .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$PLIT                       45  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                     1753  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                               Library Statistics
;
;                                       --------  Symbols  --------      Pages       Processing
;       File                            Total     Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1         18619          9         0        1000      00:01.8
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1    32          0         0           7      00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1    1545         97         6          97      00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                          418          2         0          31      00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1     386          3         0          22      00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1     150          0         0          12      00:00.3


; Information:  2
; Warnings:     0
; Errors:       0
```

```
;                                       COMMAND QUALIFIERS

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

; Size:         1753 code + 45 data bytes
; Run Time:       00:37.2
; Elapsed Time:   01:43.6
; Lines/CPU Min:   2454
; Lexemes/CPU-Min: 11235
; Memory Used:  186 pages
; Compilation Complete
```

DBGIFTHEN
LIS

DBGLANVEC
LIS

DBGLANGOP
LIS

DBGGEN
LIS

DBGLEVEL1
LIS