```
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**DBGDEFINE

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

```
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 DBGDEFINE (IDENT = 'V04-000') =
:     2          0002  1 BEGIN
:     3          0003  1
:     4          0004  1 !
:     5          0005  1 !**************************************************************************
:     6          0006  1 !*                                                                        *
:     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                             *
:     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
:     9          0009  1 !*  ALL RIGHTS RESERVED.                                                 *
:    10          0010  1 !*                                                                        *
:    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
:    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
:    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
:    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
:    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
:    16          0016  1 !*  TRANSFERRED.                                                         *
:    17          0017  1 !*                                                                        *
:    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
:    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
:    20          0020  1 !*  CORPORATION.                                                         *
:    21          0021  1 !*                                                                        *
:    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
:    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.              *
:    24          0024  1 !*                                                                        *
:    25          0025  1 !*                                                                        *
:    26          0026  1 !**************************************************************************
:    27          0027  1 !
:    28          0028  1
:    29          0029  1 !++
:    30          0030  1 ! FACILITY:     DEBUG
:    31          0031  1 !
:    32          0032  1 ! ABSTRACT:
:    33          0033  1 !
:    34          0034  1 !       This module contains all the routines that are used to implement
:    35          0035  1 !       the DEFINE command. This includes the parse and execute networks,
:    36          0036  1 !       and also the utility routines for managing the DEFINE symbol table.
:    37          0037  1 !
:    38          0038  1 ! ENVIRONMENT:  VAX/VMS
:    39          0039  1 !
:    40          0040  1 ! AUTHOR:       Richard Title, CREATION DATE:   Mar 1982
:    41          0041  1 !
:    42          0042  1 ! VERSION:      V3.1-001
:    43          0043  1 !
:    44          0044  1 ! MODIFIED BY:
:    45          0045  1 !                V. Holt, May 1982
:    46          0046  1 !
:    47          0047  1 ! REVISION HISTORY:
:    48          0048  1 ! 3B.0  14-May-82       VJH     Added call to DBG$FLUSHBUF to replace
:    49          0049  1 !                               initialization of local buffer pointer.
:    50          0050  1 ! 3B.1   3-Jun-82       VJH     Removed all references to DBG$FAO_PUT and
:    51          0051  1 !                               DBG$OUT_PUT, as these routines are now obsolete.
:    52          0052  1 !                               Replaced them with calls to DBG$PRINT and
:    53          0053  1 !                               DBG$NEWLINE, respectively.
:    54          0054  1 ! 3B.2   8-JUN-82       VJH     Removed reference to local output buffer in
:    55          0055  1 !                               routine dump_entries.
:    56          0056  1 ! 4.0   13-SEP-83       BAB     Implemented full functionality for the
:    57          0057  1 !                               DEFINE/KEY and the DELETE/KEY or UNDEFINE/KEY.
```

```
;   58          0058  1 !
```

```
;    60          0059  1 !
;    61          0060  1 ! TABLE OF CONTENTS:
;    62          0061  1 !
;    63          0062  1
;    64          0063  1 FORWARD ROUTINE
;    65          0064  1         DBG$CANCEL_LOC_VAL: NOVALUE,    ! Cancel saved values of dot and backslash
;    66          0065  1         DBG$DEF_SYM_ADD,                ! Adds a symbol to the DEFINE symbol table
;    67          0066  1         DBG$DEF_SYM_FIND,               ! Finds a symbols in the DEFINE symbol table
;    68          0067  1         DBG$DEF_SYM_REMOVE,             ! Removes a symbol from the DEFINE symbol table
;    69          0068  1         DBG$DEF_SYM_REMOVE_ALL,         ! Remove all define symbols from a list
;    70          0069  1         DBG$DUMP_DEFINE,                ! Dump the define symbol table
;    71          0070  1         DBG$NPARSE_DEFINE,              ! Parse network for DEFINE
;    72          0071  1         DBG$NPARSE_DEF_KEY,             ! Parse network for DEFINE/KEY
;    73          0072  1         DBG$NPARSE_DELETE,              ! Parse network for DELETE
;    74          0073  1         DBG$NPARSE_UNDEFINE,            ! Parse network for UNDEFINE
;    75          0074  1         DBG$NPARSE_DEL_KEY,             ! Parse network for DELETE/KEY and/or UNDEFINE/KEY
;    76          0075  1         DBG$NEXECUTE_DEFINE,            ! Execution network for DEFINE
;    77          0076  1         DBG$NEXECUTE_DELETE,            ! Execution network for DELETE
;    78          0077  1         DBG$NEXECUTE_UNDEFINE,          ! Execution network for UNDEFINE
;    79          0078  1         DBG$NREAD_NAME,                 ! Parses a name
;    80          0079  1         DBG$SAVE_LOC: NOVALUE,          ! Save .
;    81          0080  1         DBG$SAVE_VAL: NOVALUE,          ! Save \
;    82          0081  1         DBG$WILDCARD_NAME_MATCH,        ! Matches a pair of names with wildcards
;    83          0082  1         DBG$READ_KEY_INFO,              ! Returns a pointer with key info string.
;    84          0083  1         DUMP_ENTRY,                     ! Dump a define entry
;    85          0084  1         FREE_ENTRY,                     ! Frees up space occupied by a define entry
;    86          0085  1         NAME_MATCH;                     ! Matches a pair of names
;    87          0086  1
;    88          0087  1 !
;    89          0088  1 ! REQUIRE FILES:
;    90          0089  1 !
;    91          0090  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    92          0224  1 LIBRARY 'LIB$:DBGGEN.L32';
;    93          0225  1
;    94          0226  1 !
;    95          0227  1 ! EQUATED SYMBOLS:
;    96          0228  1 !
;    97          0229  1 LITERAL
;    98          0230  1         define_local  = 0,              ! Code to indicate the symbol is local
;    99          0231  1         define_global = 1,              ! Code to indicate the symbol is global
;   100          0232  1         undefine_all  = 1,              ! Code for UNDEFINE/ALL
;   101          0233  1         undefine_all_global = 2,        ! Code for UNDEFINE/ALL/GLOBAL
;   102          0234  1         undefine_key = 3;               ! Code for UNDEFINE/KEY
;   103          0235  1
;   104          0236  1 !
;   105          0237  1 ! EXTERNAL REFERENCES:
;   106          0238  1 !
;   107          0239  1 EXTERNAL LITERAL
;   108          0240  1         DBG$K_MAX_PR_NESTING,           ! Maximum level of procedure nesting
;   109          0241  1         SMG$_NOMOREKEYS,                ! No more keys in table
;   110          0242  1         SMG$_KEYNOTDEF;                 ! Key is not defined
;   111          0243  1
;   112          0244  1 EXTERNAL
;   113          0245  1         DBG$GB_KEYPAD_INPUT: BYTE,
;   114          0246  1         DBG$GB_DEFINE_PTR:REF VECTOR[,BYTE], ! Points to data structure for SET DEFINE
;   115          0247  1         DBG$GB_RADIX: VECTOR[3, BYTE],  ! Radix settings
;   116          0248  1         DBG$GL_KEY_TABLE_ID,            ! Used in DEFINE/KEY
```

```
;  117          0249  1          DBG$GL_PARAM_COUNT: VECTOR[],       ! Count of number of params
;  118          0250  1          DBG$GL_PR_NEST_LEVEL,               ! Nesting level of a procedures
;  119          0251  1          DBG$GL_SIGN_FLAG;                   ! Print '+' before signed variable
;  120          0252  1
;  121          0253  1  EXTERNAL ROUTINE
;  122          0254  1          DBG$COPY_MEMORY,                    ! Copy a block of memory
;  123          0255  1          DBG$GET_MEMORY,                     ! Allocate permanent memory
;  124          0256  1          DBG$GET_TEMPMEM,                    ! Allocate permanent memory
;  125          0257  1          DBG$PRIM_TO_VAL,                    ! Convert descriptors
;  126          0258  1          DBG$PRINT:  NOVALUE,                ! Formatted ASCII output
;  127          0259  1          DBG$PRINT_IDENTIFIER: NOVALUE,      ! Print name of Primary
;  128          0260  1          DBG$PRINT_VALUE: NOVALUE,           ! Print value of Value Descriptor
;  129          0261  1          DBG$POP_TEMPMEM: NOVALUE,           ! Pop temporary memory pool
;  130          0262  1          DBG$PUSH_TEMPMEM,                   ! Push temporary memory pool
;  131          0263  1          DBG$FLUSHBUF:  NOVALUE,             ! Initialize new print line
;  132          0264  1          DBG$NEWLINE:  NOVALUE,              ! Output print buffer to terminal
;  133          0265  1          DBG$NACCEPT_STRING,                 ! Accepts a quoted string
;  134          0266  1          DBG$NCOPY_DESC,                     ! Copies a descriptor
;  135          0267  1          DBG$NEXTLOC,                        ! Logical successor
;  136          0268  1          DBG$NFREE_DESC,                     ! Free up space occupied by a descriptor
;  137          0269  1          DBG$NGET_SYMID,                     ! Obtain symid list
;  138          0270  1          DBG$NMAKE_ARG_VECT,                 ! Constructs a message vector
;  139          0271  1          DBG$NMATCH,                         ! Matches a token
;  140          0272  1          DBG$NNEXT_WORD,                     ! Gets the next word of input
;  141          0273  1          DBG$NPARSE_ADDRESS,                 ! Parse an address expression
;  142          0274  1          DBG$NPARSE_EXPRESSION,              ! Parse a value expression
;  143          0275  1          DBG$NSAVE_BREAK_BUFFER,             ! Reads a buffer of DEBUG commands
;  144          0276  1          DBG$NSYNTAX_ERROR,                  ! Constructs a message vector for a syntax error
;  145          0277  1          DBG$NTYPE_CONV,                     ! Type converter
;  146          0278  1          DBG$PREVLOC,                        ! Logical predecessor
;  147          0279  1          DBG$REL_MEMORY,                     ! Release memory
;  148          0280  1          DBG$SET_DEFINE_LVL,                 ! Manipulates data structure for
;  149          0281  1                                              !    SET DEFINE command
;  150          0282  1          DBG$STA_LOCK_SYMID: NOVALUE,        ! Lock symid list
;  151          0283  1          DBG$STA_UNLOCK_SYMID: NOVALUE,      ! Unlock symid list
;  152          0284  1          STR$COMPARE_EQL,                    ! Compares two descriptors
;  153          0285  1          SMG$ADD_KEY_DEF,                    ! Execute the DEFINE/KEY command
;  154          0286  1          SMG$DELETE_KEY_DEF,                 ! Processes DELETE/KEY
;  155          0287  1          SMG$LIST_KEY_DEFS,                  ! Returns key definitions from the table
;  156          0288  1          SMG$SET_DEFAULT_STATE;              ! Returns the current default state
;  157          0289  1
;  158          0290  1  !
;  159          0291  1  ! OWN STORAGE
;  160          0292  1  !
;  161          0293  1  OWN
;  162          0294  1          CURLOC_VMSDESC: BLOCK[12, BYTE];    ! Area to store a copy of the vms descriptor
;  163          0295  1                                              !     representing dot (current location)
;  164          0296  1
;  165          0297  1  GLOBAL
;  166          0298  1          DBG$GL_CURLOC_VMSDESC: INITIAL(0),  ! Pointer to a copy of the vms descriptor
;  167          0299  1                                              !     representing dot (current location)
;  168          0300  1          DBG$GL_GLOBAL_DEFINE_PTR,           ! Pointer to head of global define list
;  169          0301  1          DBG$GL_LOCAL_DEFINE_PTR;            ! Pointer to head of local define list
;  170          0302  1
;  171          0303  1  !
;  172          0304  1  ! MACROS
;  173          0305  1  !
```

```
;  174          0306  1 ! The following macro is just an abbreviation for some error-reporting
;  175          0307  1 ! code that occurs repeatedly
;  176          0308  1 !
;  177        M 0309  1 MACRO report_error =
;  178        M 0310  1     BEGIN
;  179        M 0311  1     .message_vect = (
;  180        M 0312  1         IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  181        M 0313  1         THEN
;  182        M 0314  1             dbg$nmake_arg_vect (dbg$_needmore)
;  183        M 0315  1         ELSE
;  184        M 0316  1             dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;  185        M 0317  1     RETURN sts$k_severe;
;  186        M 0318  1     END %;
;  187          0319  1
;  188          0320  1 !+
;  189          0321  1 ! Definition for the list of state names in the IF_STATE qualifier of the
;  190          0322  1 ! Define/key command.
;  191          0323  1 !-
;  192          0324  1
;  193          0325  1 FIELD
;  194          0326  1     DBG$STATE_NAME_FIELDS =
;  195          0327  1         SET
;  196          0328  1
;  197          0329  1         DBG$L_STATE_NAME_PTR            = [0, 0, 32, 0],        ! Pointer to name descriptor
;  198          0330  1         DBG$L_STATE_NAME_LINK           = [1, 0, 32, 0]         ! Pointer to next state name
;  199          0331  1
;  200          0332  1         TES;
;  201          0333  1
;  202          0334  1 LITERAL
;  203          0335  1         DBG$K_STATE_NAME_SIZE           = 2;                    ! length in long words
;  204          0336  1
;  205          0337  1 MACRO
;  206          0338  1         DBG$STATE_NAME_NODE = BLOCK [DBG$K_STATE_NAME_SIZE] FIELD (DBG$STATE_NAME_FIELDS) %;
;  207          0339  1
;  208          0340  1
```

```
;  210          0341  1 GLOBAL ROUTINE dbg$cancel_loc_val: NOVALUE =
;  211          0342  1 !++
;  212          0343  1 ! Routine Description
;  213          0344  1 !
;  214          0345  1 !       Cancels the stored values for dot and backslash. This routine is
;  215          0346  1 !       called during the processing of the SET LANGUAGE command.
;  216          0347  1 !
;  217          0348  1 ! Inputs
;  218          0349  1 !
;  219          0350  1 !       None.
;  220          0351  1 !
;  221          0352  1 ! Outputs
;  222          0353  1 !
;  223          0354  1 !       The DEFINE symbol table is modified.
;  224          0355  1 !--
;  225          0356  2     BEGIN
;  226          0357  2     LOCAL
;  227          0358  2         dummy;          ! Output parameter - value not used here
;  228          0359  2
;  229          0360  2     ! Remove definition for backslash.
;  230          0361  2     !
;  231          0362  2     dbg$def_sym_remove (UPLIT BYTE (%ASCIC '%CURVAL'),
;  232          0363  2                         TRUE,
;  233          0364  2                         dummy,
;  234          0365  2                         dummy);
;  235          0366  2
;  236          0367  2     ! Remove the definition for dot.
;  237          0368  2     !
;  238          0369  2     dbg$def_sym_remove (UPLIT BYTE (%ASCIC '%CURLOC'),
;  239          0370  2                         TRUE,
;  240          0371  2                         dummy,
;  241          0372  2                         dummy);
;  242          0373  2
;  243          0374  2     ! Zero out the pointer to the saved vms descriptor for dot.
;  244          0375  2     !
;  245          0376  2     dbg$gl_curloc_vmsdesc = 0;
;  246          0377  1     END;


                                                        .TITLE  DBGDEFINE
                                                        .IDENT  \V04-000\

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

4C  41  56  52  55  43  25  07  00000 P.AAA:    .ASCII  <7>\%CURVAL\                            ;
43  4F  4C  52  55  43  25  07  00008 P.AAB:    .ASCII  <7>\%CURLOC\                            ;

                                                        .PSECT  DBG$OWN,NOEXE,  PIC,2

                                    00000 CURLOC_VMSDESC:
                                                        .BLKB   12

                                                        .PSECT  DBG$GLOBAL,NOEXE,  PIC,2

                                00000000    00000 DBG$GL_CURLOC_VMSDESC::
                                                        .LONG   0                               ;
                                    00004 DBG$GL_GLOBAL_DEFINE_PTR::
```

```
                        .BLKB   4
00008 DBG$GL_LOCAL_DEFINE_PTR::
                        .BLKB   4

                        .EXTRN  DBG$K_MAX_PR_NESTING
                        .EXTRN  SMG$_NOMOREKEYS
                        .EXTRN  SMG$_KEYNOTDEF, DBG$GB_KEYPAD_INPUT
                        .EXTRN  DBG$GB_DEFINE_PTR
                        .EXTRN  DBG$GB_RADIX, DBG$GL_KEY_TABLE_ID
                        .EXTRN  DBG$GL_PARAM_COUNT
                        .EXTRN  DBG$GL_PR_NEST_LEVEL
                        .EXTRN  DBG$GL_SIGN_FLAG
                        .EXTRN  DBG$COPY_MEMORY
                        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
                        .EXTRN  DBG$PRIM_TO_VAL
                        .EXTRN  DBG$PRINT, DBG$PRINT_IDENTIFIER
                        .EXTRN  DBG$PRINT_VALUE
                        .EXTRN  DBG$POP_TEMPMEM
                        .EXTRN  DBG$PUSH_TEMPMEM
                        .EXTRN  DBG$FLUSHBUF, DBG$NEWLINE
                        .EXTRN  DBG$NACCEPT_STRING
                        .EXTRN  DBG$NCOPY_DESC, DBG$NEXTLOC
                        .EXTRN  DBG$NFREE_DESC, DBG$NGET_SYMID
                        .EXTRN  DBG$NMAKE_ARG_VECT
                        .EXTRN  DBG$NMATCH, DBG$NNEXT_WORD
                        .EXTRN  DBG$NPARSE_ADDRESS
                        .EXTRN  DBG$NPARSE_EXPRESSION
                        .EXTRN  DBG$NSAVE_BREAK_BUFFER
                        .EXTRN  DBG$NSYNTAX_ERROR
                        .EXTRN  DBG$NTYPE_CONV, DBG$PREVLOC
                        .EXTRN  DBG$REL_MEMORY, DBG$SET_DEFINE_LVL
                        .EXTRN  DBG$STA_LOCK_SYMID
                        .EXTRN  DBG$STA_UNLOCK_SYMID
                        .EXTRN  STR$COMPARE_EQL
                        .EXTRN  SMG$ADD_KEY_DEF
                        .EXTRN  SMG$DELETE_KEY_DEF
                        .EXTRN  SMG$LIST_KEY_DEFS
                        .EXTRN  SMG$SET_DEFAULT_STATE

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

                    0000 00000  .ENTRY  DBG$CANCEL_LOC_VAL, Save nothing  ; 0341
                5E      04  C2  00002  SUBL2  #4, SP                      ;
                5E      DD  00005  PUSHL  SP                              ; 0362
                    04  AE  9F  00007  PUSHAB  DUMMY                     ;
                    01  DD  0000A  PUSHL  #1                             ;
            00000000'   EF  9F  0000C  PUSHAB  P.AAA                      ;
    0000V   CF          04  FB  00012  CALLS  #4, DBG$DEF_SYM_REMOVE      ;
                5E      DD  00017  PUSHL  SP                              ; 0369
                    04  AE  9F  00019  PUSHAB  DUMMY                     ;
                    01  DD  0001C  PUSHL  #1                             ;
            00000000'   EF  9F  0001E  PUSHAB  P.AAB                      ;
    0000V   CF          04  FB  00024  CALLS  #4, DBG$DEF_SYM_REMOVE      ;
            00000000'   EF  D4  00029  CLRL  DBG$GL_CURLOC_VMSDESC        ; 0376
                        04  0002F  RET                                   ; 0377

; Routine Size:  48 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;   248     0378  1 GLOBAL ROUTINE dbg$def_sym_add (symbol_name, symbol_kind,   symbol_value,
;   249     0379  1                                  global_flag, replaced_flag, message_vect) =
;   250     0380  1 !++
;   251     0381  1 ! Functional Description
;   252     0382  1 !
;   253     0383  1 !       This routine adds a new symbol to DEBUG's internal DEFINE symbol
;   254     0384  1 !       table. It is given the symbol's name, value, kind, and an
;   255     0385  1 !       indication of whether the symbol is to be local or global.
;   256     0386  1 !       It sets the output parameter REPLACED_FLAG to TRUE if the
;   257     0387  1 !       symbol replaced an existing occurence.
;   258     0388  1 !
;   259     0389  1 ! Routine Inputs
;   260     0390  1 !
;   261     0391  1 !       SYMBOL_NAME -   Points to a counted string with the symbol name.
;   262     0392  1 !       SYMBOL_KIND -   One of the legal kinds DEFINE_ADDRESS, DEFINE_VALUE,
;   263     0393  1 !                       and so on, (defined in DBGLIB.REQ)
;   264     0394  1 !       SYMBOL_VALUE -  Points to the "value" of the symbol. This may be
;   265     0395  1 !                       an address expression descriptor, a value
;   266     0396  1 !                       descriptor, and so on, depending on kind.
;   267     0397  1 !       GLOBAL_FLAG -   TRUE if the symbol is global, FALSE otherwise.
;   268     0398  1 !       REPLACED_FLAG - This is the address of a flag to be set to TRUE if
;   269     0399  1 !                       adding the symbol replaces an existing occurence
;   270     0400  1 !                       of the name.
;   271     0401  1 !       MESSAGE_VECT -  The address of an error message vector.
;   272     0402  1 !
;   273     0403  1 ! Routine Outputs
;   274     0404  1 !
;   275     0405  1 !       REPLACED_FLAG - This is the address of a flag to be set to TRUE if
;   276     0406  1 !                       adding the symbol replaces an existing occurence
;   277     0407  1 !                       of the name.
;   278     0408  1 !
;   279     0409  1 ! Routine Value
;   280     0410  1 !
;   281     0411  1 !       An unsigned longword completion code:
;   282     0412  1 !
;   283     0413  1 !       STS$K_SUCCESS - Success. Symbol addition was successful.
;   284     0414  1 !       STS$K_SEVERE -  Failure. Message argument vector is constructed
;   285     0415  1 !                               with an indication of the error.
;   286     0416  1 !
;   287     0417  1 !--
;   288     0418  2     BEGIN
;   289     0419  2
;   290     0420  2     LOCAL
;   291     0421  2         head_ptr: REF define$header,    ! Points to the head of the
;   292     0422  2                                         !     appropriate DEFINE list
;   293     0423  2         first_entry,                    ! Flag saying whether this is the
;   294     0424  2                                         !     first entry on the list
;   295     0425  2         name: REF VECTOR [,BYTE],       ! Points to a counted string with
;   296     0426  2                                         !     the DEFINE symbol name
;   297     0427  2         prev_sym_ptr: REF define$entry, ! Points to a DEFINE entry
;   298     0428  2         sym_ptr: REF define$entry;      !
;   299     0429  2
;   300     0430  2     !++
;   301     0431  2     ! First we search for a pre-existing occurence of the symbol.
;   302     0432  2     !--
;   303     0433  2
;   304     0434  2     ! Set up pointer to the appropriate define list.
```

```
;  305     0435  2      !
;  306     0436  2      IF .global_flag
;  307     0437  2      THEN
;  308     0438  2          head_ptr = .dbg$gl_global_define_ptr
;  309     0439  2      ELSE
;  310     0440  2          head_ptr = .dbg$gl_local_define_ptr;
;  311     0441  2
;  312     0442  2      ! Walk through the list searching for the given name.
;  313     0443  2      !
;  314     0444  2      sym_ptr = .head_ptr [def$a_define_list];
;  315     0445  2
;  316     0446  2      ! Set the flag saying whether there are any entries on the list.
;  317     0447  2      !
;  318     0448  2      IF .sym_ptr EQL 0
;  319     0449  2      THEN
;  320     0450  2          first_entry = TRUE
;  321     0451  2      ELSE
;  322     0452  2          first_entry = FALSE;
;  323     0453  2
;  324     0454  2      prev_sym_ptr = 0;
;  325     0455  2      WHILE .sym_ptr NEQ 0 DO
;  326     0456  3          BEGIN
;  327     0457  3          name = .sym_ptr [def$a_name];
;  328     0458  3          IF name_match (.symbol_name, .name)
;  329     0459  3          THEN
;  330     0460  4              BEGIN
;  331     0461  4
;  332     0462  4              !++
;  333     0463  4              ! If we find a match then replace the existing entry
;  334     0464  4              ! with the new definition.
;  335     0465  4              !--
;  336     0466  4
;  337     0467  4              ! First free up the space taken up by the old entry.
;  338     0468  4              !
;  339     0469  4              IF NOT free_entry (.sym_ptr, .message_vect)
;  340     0470  4              THEN
;  341     0471  4                  RETURN sts$k_severe;
;  342     0472  4
;  343     0473  4              ! Fill in the fields with the new values.
;  344     0474  4              !
;  345     0475  4              sym_ptr [def$b_kind] = .symbol_kind;
;  346     0476  4              sym_ptr [def$a_name] = .symbol_name;
;  347     0477  4              sym_ptr [def$a_value] = .symbol_value;
;  348     0478  4              .replaced_flag = TRUE;
;  349     0479  4
;  350     0480  4              ! Return success
;  351     0481  4              !
;  352     0482  4              RETURN sts$k_success;
;  353     0483  3              END;
;  354     0484  3
;  355     0485  3          ! Set up for the next time around the loop.
;  356     0486  3          !
;  357     0487  3          prev_sym_ptr = .sym_ptr;
;  358     0488  3          sym_ptr = .sym_ptr [def$a_entry_next_link];
;  359     0489  2          END; ! While loop
;  360     0490  2
;  361     0491  2      ! If we get here then we have failed to find a define entry
```

```
;  362          0492  2     ! with that name. Allocate a new entry and link it in.
;  363          0493  2     !
;  364          0494  2     sym_ptr = dbg$get_memory (dbg$k_define_entry_size_w);
;  365          0495  2
;  366          0496  2     ! If we are adding the first one, then it is linked to the
;  367          0497  2     ! header; else it is linked to the last symbol on the list.
;  368          0498  2     !
;  369          0499  2     IF .first_entry
;  370          0500  2     THEN
;  371          0501  2         head_ptr [def$a_define_list] = .sym_ptr
;  372          0502  2     ELSE
;  373          0503  2         prev_sym_ptr [def$a_entry_next_link] = .sym_ptr;
;  374          0504  2
;  375          0505  2     ! Fill in the fields of the newly allocated entry.
;  376          0506  2     !
;  377          0507  2     sym_ptr [def$a_entry_next_link] = 0;
;  378          0508  2     sym_ptr [def$a_entry_prev_link] = .prev_sym_ptr;
;  379          0509  2     sym_ptr [def$b_kind] = .symbol_kind;
;  380          0510  2     sym_ptr [def$a_name] = .symbol_name;
;  381          0511  2     sym_ptr [def$a_value] = .symbol_value;
;  382          0512  2
;  383          0513  2     ! Set the output parameter and return success.
;  384          0514  2     !
;  385          0515  2     .replaced_flag = FALSE;
;  386          0516  2     RETURN sts$k_success;
;  387          0517  2
;  388          0518  1     END; ! dbg$def_sym_add


                                007C 00000          .ENTRY  DBG$DEF_SYM_ADD, Save R2,R3,R4,R5,R6                    ; 0378
                09          AC   E9 00002          BLBC    GLOBAL_FLAG, 1$                                         ; 0436
                53 00000000'EF   D0 00006          MOVL    DBG$GL_GLOBAL_DEFINE_PTR, HEAD_PTR                      ; 0438
                            07   11 0000D          BRB     2$                                                      ;
                53 00000000'EF   D0 0000F 1$:      MOVL    DBG$GL_LOCAL_DEFINE_PTR, HEAD_PTR                       ; 0440
                52          08   A3 D0 00016 2$:   MOVL    8(HEAD_PTR), SYM_PTR                                    ; 0444
                            05   12 0001A          BNEQ    3$                                                      ; 0448
                55              01 D0 0001C       MOVL    #1, FIRST_ENTRY                                         ; 0450
                                02 11 0001F       BRB     4$                                                      ;
                                55 D4 00021 3$:   CLRL    FIRST_ENTRY                                             ; 0452
                                54 D4 00023 4$:   CLRL    PREV_SYM_PTR                                            ; 0454
                                52 D5 00025 5$:   TSTL    SYM_PTR                                                 ; 0455
                                3F 13 00027       BEQL    8$                                                      ;
                56          08   A2 D0 00029       MOVL    8(SYM_PTR), NAME                                        ; 0457
                                56 DD 0002D       PUSHL   NAME                                                    ; 0458
                            04   AC DD 0002F       PUSHL   SYMBOL_NAME                                             ;
0000V           CF              02 FB 00032       CALLS   #2, NAME_MATCH                                          ;
                26              50 E9 00037       BLBC    R0, 7$                                                  ;
                            18   AC DD 0003A       PUSHL   MESSAGE_VECT                                            ; 0469
                                52 DD 0003D       PUSHL   SYM_PTR                                                 ;
0000V           CF              02 FB 0003F       CALLS   #2, FREE_ENTRY                                          ;
                04              50 E8 00044       BLBS    R0, 6$                                                  ;
                50              04 D0 00047       MOVL    #4, R0                                                  ; 0471
                                   04 0004A       RET                                                             ;
              10    A2        08   AC 90 0004B 6$:   MOVB    SYMBOL_KIND, 16(SYM_PTR)                                ; 0475
```

```
                08   A2        04   AC   D0 00050          MOVL    SYMBOL_NAME, 8(SYM_PTR)                 ; 0476
                0C   A2        0C   AC   D0 00055          MOVL    SYMBOL_VALUE, 12(SYM_PTR)               ; 0477
                14   BC             01   D0 0005A          MOVL    #1, @REPLACED_FLAG                      ; 0478
                                    38   11 0005E          BRB     11$                                     ; 0482
                     54             52   D0 00060 7$:      MOVL    SYM_PTR, PREV_SYM_PTR                   ; 0487
                     52             62   D0 00063          MOVL    (SYM_PTR), SYM_PTR                      ; 0488
                                    BD   11 00066          BRB     5$                                      ; 0455
                                    05   DD 00068 8$:      PUSHL   #5                                      ; 0494
          00000000G  00             01   FB 0006A          CALLS   #1, DBG$GET_MEMORY                      ;
                     52             50   D0 00071          MOVL    R0, SYM_PTR                             ;
                     06             55   E9 00074          BLBC    FIRST_ENTRY, 9$                         ; 0499
                08   A3             52   D0 00077          MOVL    SYM_PTR, 8(HEAD_PTR)                    ; 0501
                                    03   11 0007B          BRB     10$                                     ;
                     64             52   D0 0007D 9$:      MOVL    SYM_PTR, (PREV_SYM_PTR)                 ; 0503
                                    62   D4 00080 10$:     CLRL    (SYM_PTR)                               ; 0507
                04   A2             54   D0 00082          MOVL    PREV_SYM_PTR, 4(SYM_PTR)                ; 0508
                10   A2        08   AC   90 00086          MOVB    SYMBOL_KIND, 16(SYM_PTR)                ; 0509
                08   A2        04   AC   D0 0008B          MOVL    SYMBOL_NAME, 8(SYM_PTR)                 ; 0510
                0C   A2        0C   AC   D0 00090          MOVL    SYMBOL_VALUE, 12(SYM_PTR)               ; 0511
                               14   BC   D4 00095          CLRL    @REPLACED_FLAG                          ; 0515
                     50             01   D0 00098 11$:     MOVL    #1, R0                                  ; 0516
                                         04 0009B          RET                                             ; 0518

; Routine Size:  156 bytes,    Routine Base:  DBG$CODE + 0030
```

```
;  390     0519  1 GLOBAL ROUTINE dbg$def_sym_find (symbol_name, returned_kind,
;  391     0520  1                                   returned_value, global_flag,
;  392     0521  1                                   message_vect) =
;  393     0522  1 !++
;  394     0523  1 ! Functional Description
;  395     0524  1 !
;  396     0525  1 !       This routine looks up a name in the DEFINE symbol table. It
;  397     0526  1 !       returns the kind of symbol and the value of the symbol.
;  398     0527  1 !
;  399     0528  1 ! Routine Inputs
;  400     0529  1 !
;  401     0530  1 !       SYMBOL_NAME -   Points to a counted string with the name to
;  402     0531  1 !                       be looked up.
;  403     0532  1 !       RETURNED_KIND - The address of a longword, in which a code for
;  404     0533  1 !                       the kind of symbol will be deposited.
;  405     0534  1 !       RETURNED_VALUE- The address of a longword, in which a pointer
;  406     0535  1 !                       to the symbol value will be deposited.
;  407     0536  1 !       GLOBAL_FLAG -   The address of a longword. The longword will
;  408     0537  1 !                       be filled in with FALSE if the symbol was found
;  409     0538  1 !                       in the local define list, and to TRUE if the
;  410     0539  1 !                       symbol was found only in the global DEFINE list.
;  411     0540  1 !
;  412     0541  1 ! Routine Outputs
;  413     0542  1 !
;  414     0543  1 !       RETURNED_KIND - See above
;  415     0544  1 !       RETURNED_VALUE- See above
;  416     0545  1 !       GLOBAL_FLAG -   See above
;  417     0546  1 !
;  418     0547  1 ! Return Value
;  419     0548  1 !
;  420     0549  1 !       TRUE -  A matching symbol was found.
;  421     0550  1 !       FALSE - A matching symbol was not found.
;  422     0551  1 !--
;  423     0552  2     BEGIN
;  424     0553  2     MAP
;  425     0554  2         symbol_name: REF VECTOR[,BYTE];
;  426     0555  2
;  427     0556  2     LOCAL
;  428     0557  2         desc_copy,                          ! Points to a copy of a primary descriptor
;  429     0558  2         head_ptr : REF define$header,       ! Points to a header block in the
;  430     0559  2                                             !    DEFINE symbol table.
;  431     0560  2         name : REF VECTOR [,BYTE],          ! Points to a symbol name.
;  432     0561  2         nextloc_flag,                       ! TRUE for %NEXTLOC
;  433     0562  2         prevloc_flag,                       ! TRUE for %PREVLOC
;  434     0563  2         sym_ptr : REF define$entry;         ! Points to an entry in the
;  435     0564  2                                             !    DEFINE symbol table.
;  436     0565  2
;  437     0566  2     ! Set up nextloc_flag and prevloc_flag.
;  438     0567  2     !
;  439     0568  2     prevloc_flag = FALSE;
;  440     0569  2     nextloc_flag = FALSE;
;  441     0570  2     IF .symbol_name[0] EQL 8
;  442     0571  2     THEN
;  443     0572  3         BEGIN
;  444     0573  3         IF ch$eql (8, symbol_name[1],
;  445     0574  3                    8, UPLIT BYTE (%ASCII '%PREVLOC'))
;  446     0575  3         THEN
```

```
;  447          0576  4          BEGIN
;  448          0577  4          prevloc_flag = TRUE;
;  449          0578  4
;  450          0579  4          ! Dummy up the symbol to look like %CURLOC.
;  451          0580  4          !
;  452          0581  4          symbol_name = UPLIT BYTE (%ASCIC '%CURLOC');
;  453          0582  4          END;
;  454          0583  3      IF ch$eql (8, symbol_name[1],
;  455          0584  3                 8, UPLIT BYTE (%ASCII '%NEXTLOC'))
;  456          0585  3      THEN
;  457          0586  4          BEGIN
;  458          0587  4          nextloc_flag = TRUE;
;  459          0588  4
;  460          0589  4          ! Dummy up the symbol to look like %CURLOC.
;  461          0590  4          !
;  462          0591  4          symbol_name = UPLIT BYTE (%ASCIC '%CURLOC');
;  463          0592  3          END;
;  464          0593  2      END;
;  465          0594  2
;  466          0595  2  ! First search the local define lists, then the global define list.
;  467          0596  2  ! When I EQL 1 we are searching local, and when I EQL 2 we are
;  468          0597  2  ! searching the global list.
;  469          0598  2  !
;  470          0599  2  INCR i FROM 1 TO 2 DO
;  471          0600  3      BEGIN
;  472          0601  3
;  473          0602  3      ! Set up HEAD_PTR to point to the appropriate list,
;  474          0603  3      ! and SYM_PTR to point to the first entry in this list.
;  475          0604  3      !
;  476          0605  3      IF .i EQL 1
;  477          0606  3      THEN
;  478          0607  3          head_ptr = .dbg$gl_local_define_ptr
;  479          0608  3      ELSE
;  480          0609  3          head_ptr = .dbg$gl_global_define_ptr;
;  481          0610  3
;  482          0611  3      ! Loop through all the define lists, from the current (innermost)
;  483          0612  3      ! scope outward.
;  484          0613  3      !
;  485          0614  3      WHILE .head_ptr NEQ 0 DO
;  486          0615  4          BEGIN
;  487          0616  4          sym_ptr = .head_ptr [def$a_define_list];
;  488          0617  4
;  489          0618  4          ! Walk the list looking for a name match.
;  490          0619  4          !
;  491          0620  4          WHILE .sym_ptr NEQ 0 DO
;  492          0621  5              BEGIN
;  493          0622  5              name = .sym_ptr [def$a_name];
;  494          0623  5              IF name_match (.symbol_name, .name)
;  495          0624  5              THEN
;  496          0625  6                  BEGIN
;  497          0626  6
;  498          0627  6                  ! We have found a match. Fill in the output
;  499          0628  6                  ! parameters.
;  500          0629  6                  !
;  501          0630  6                  .returned_kind = .sym_ptr [def$b_kind];
;  502          0631  6                  .returned_value = .sym_ptr [def$a_value];
;  503          0632  6                  IF .i EQL 1
```

```
;  504          0633  6          THEN
;  505          0634  6              .global_flag = FALSE
;  506          0635  6          ELSE
;  507          0636  6              .global_flag = TRUE;
;  508          0637  6
;  509          0638  6          ! Check for special symbols %PREVLOC and %NEXTLOC.
;  510          0639  6          ! These are handled specially - we call the new
;  511          0640  6          ! routines DBG$PREVLOC or DBG$NEXTLOC to obtain the
;  512          0641  6          ! logical successor or predecessor.
;  513          0642  6          !
;  514          0643  6          IF .prevloc_flag
;  515          0644  6          THEN
;  516          0645  7              BEGIN
;  517          0646  7
;  518          0647  7              ! Construct a copy of the descriptor into
;  519          0648  7              ! temporary memory so that DBG$PREVLOC
;  520          0649  7              ! can play with it. Then call DBG$PREVLOC to
;  521          0650  7              ! give us the logical predecessor.
;  522          0651  7              !
;  523          0652  7              dbg$ncopy_desc (.sym_ptr [def$a_value],
;  524          0653  7                              desc_copy,
;  525          0654  7                              .message_vect,
;  526          0655  7                              FALSE);
;  527          0656  7              .returned_value = dbg$prevloc (.desc_copy);
;  528          0657  6              END;
;  529          0658  6          IF .nextloc_flag
;  530          0659  6          THEN
;  531          0660  7              BEGIN
;  532          0661  7
;  533          0662  7              ! Construct a copy of the descriptor into
;  534          0663  7              ! temporary memory so that DBG$NEXTLOC
;  535          0664  7              ! can play with it. Then call DBG$NEXTLOC to
;  536          0665  7              ! give us the logical predecessor.
;  537          0666  7              !
;  538          0667  7              dbg$ncopy_desc (.sym_ptr [def$a_value],
;  539          0668  7                              desc_copy,
;  540          0669  7                              .message_vect,
;  541          0670  7                              FALSE);
;  542          0671  7              .returned_value = dbg$nextloc (.desc_copy);
;  543          0672  6              END;
;  544          0673  6
;  545          0674  6                  RETURN TRUE;
;  546          0675  5                  END;
;  547          0676  5
;  548          0677  5              ! Set up for next time around loop.
;  549          0678  5              !
;  550          0679  5              sym_ptr = .sym_ptr [def$a_entry_next_link];
;  551          0680  4              END; ! inner while loop
;  552          0681  4
;  553          0682  4          ! Set up for the next time around the loop.
;  554          0683  4          !
;  555          0684  4          head_ptr = .head_ptr [def$a_next_link];
;  556          0685  3          END; ! Outer while loop
;  557          0686  3
;  558          0687  2      END; ! Incr loop
;  559          0688  2
;  560          0689  2  ! If we reach this point we have failed to find a
```

```
;  561          0690  2     ! name match. Return false.
;  562          0691  2     ! Note - if we were searching for %CURLOC or %CURVAL and didn't find
;  563          0692  2     ! it that means that dot or backslash were not defined. In this case,
;  564          0693  2     ! signal an appropriate error message.
;  565          0694  2     !
;  566          0695  2     IF .symbol_name[0] EQL 7
;  567          0696  2     THEN
;  568          0697  2         BEGIN
;  569          0698  2         IF CH$EQL (7, symbol_name[1], 7, UPLIT BYTE (%ASCII '%CURLOC'))
;  570          0699  2         THEN
;  571          0700  2             SIGNAL (dbg$_nocurloc);
;  572          0701  2         IF CH$EQL (7, symbol_name[1], 7, UPLIT BYTE (%ASCII '%CURVAL'))
;  573          0702  2         THEN
;  574          0703  2             SIGNAL (dbg$_nolastval);
;  575          0704  2         END;
;  576          0705  2     RETURN FALSE;
;  577          0706  2
;  578          0707  1     END; ! dbg$def_sym_find


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

                43  4F  4C  56  45  52  50  25  00010 P.AAC:  .ASCII  \%PREVLOC\                    ;
                43  4F  4C  52  55  43  25  07  00018 P.AAD:  .ASCII  <7>\%CURLOC\                  ;
                43  4F  4C  54  58  45  4E  25  00020 P.AAE:  .ASCII  \%NEXTLOC\                   ;
                43  4F  4C  52  55  43  25  07  00028 P.AAF:  .ASCII  <7>\%CURLOC\                  ;
                    43  4F  4C  52  55  43  25  00030 P.AAG:  .ASCII  \%CURLOC\                    ;
                    4C  41  56  52  55  43  25  00037 P.AAH:  .ASCII  \%CURVAL\                    ;


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

                                    0FFC 00000          .ENTRY  DBG$DEF_SYM_FIND, Save R2,R3,R4,R5,R6,R7,-  ; 0519
                                                                R8,R9,R10,R11                               ;
                    5B 00000000G    00   9E 00002      MOVAB   DBG$NCOPY_DESC, R11                         ;
                    5A 00000000'    EF   9E 00009      MOVAB   P.AAC, R10                                  ;
                    5E              04   C2 00010      SUBL2   #4, SP                                      ;
                                    56   7C 00013      CLRQ    PREVLOC_FLAG                                ; 0568
                    50      04      AC   D0 00015      MOVL    SYMBOL_NAME, R0                             ; 0570
                    08              60   91 00019      CMPB    (R0), #8                                    ;
                                    23   12 0001C      BNEQ    2$                                          ;
            6A      01      A0      08   29 0001E      CMPC3   #8, 1(R0), P.AAC                            ; 0573
                                    08   12 00023      BNEQ    1$                                          ;
                    56              01   D0 00025      MOVL    #1, PREVLOC_FLAG                            ; 0577
                04  AC          08  AA   9E 00028      MOVAB   P.AAD, SYMBOL_NAME                          ; 0581
                    50      04      AC   D0 0002D 1$:  MOVL    SYMBOL_NAME, R0                             ; 0583
        10  AA      01      A0      08   29 00031      CMPC3   #8, 1(R0), P.AAE                            ;
                                    08   12 00037      BNEQ    2$                                          ;
                    57              01   D0 00039      MOVL    #1, NEXTLOC_FLAG                            ; 0587
                04  AC          18  AA   9E 0003C      MOVAB   P.AAF, SYMBOL_NAME                          ; 0591
                    54      04      AC   D0 00041 2$:  MOVL    SYMBOL_NAME, R4                             ; 0623
                    55              01   D0 00045      MOVL    #1, I                                       ;
                                    58   D4 00048 3$:  CLRL    R8                                          ; 0605
                    01              55   D1 0004A      CMPL    I, #1                                       ;
                                    0B   12 0004D      BNEQ    4$                                          ;
```

```
                                  58  D6 0004F       INCL     R8                                         ;
              53 00000000'        EF  D0 00051       MOVL     DBG$GL_LOCAL_DEFINE_PTR, HEAD_PTR          ; 0607
                                  07  11 00058       BRB      5$                                         ;
              53 00000000'        EF  D0 0005A 4$:   MOVL     DBG$GL_GLOBAL_DEFINE_PTR, HEAD_PTR         ; 0609
                                  76  13 00061 5$:   BEQL     13$                                        ; 0614
              52           08     A3  D0 00063       MOVL     8(HEAD_PTR), SYM_PTR                       ; 0616
                                  6B  13 00067 6$:   BEQL     12$                                        ; 0620
              59           08     A2  D0 00069       MOVL     8(SYM_PTR), NAME                           ; 0622
                         0210     8F  BB 0006D       PUSHR    #^M<R4,R9>                                 ; 0623
         0000V  CF                02  FB 00071       CALLS    #2, NAME_MATCH                             ;
                56                50  E9 00076       BLBC     R0, 11$                                    ;
           08   BC         10     A2  9A 00079       MOVZBL   16(SYM_PTR), @RETURNED_KIND                ; 0630
           0C   BC         0C     A2  D0 0007E       MOVL     12(SYM_PTR), @RETURNED_VALUE               ; 0631
                05                58  E9 00083       BLBC     R8, 7$                                     ; 0634
                           10     BC  D4 00086       CLRL     @GLOBAL_FLAG                               ;
                                  04  11 00089       BRB      8$                                         ;
           10   BC                01  D0 0008B 7$:   MOVL     #1, @GLOBAL_FLAG                           ; 0636
                1B                56  E9 0008F 8$:   BLBC     PREVLOC_FLAG, 9$                           ; 0643
                                  7E  D4 00092       CLRL     -(SP)                                      ; 0652
                           14     AC  DD 00094       PUSHL    MESSAGE_VECT                               ; 0654
                           08     AE  9F 00097       PUSHAB   DESC_COPY                                  ; 0652
                           0C     A2  DD 0009A       PUSHL    12(SYM_PTR)                                ;
                6B                04  FB 0009D       CALLS    #4, DBG$NCOPY_DESC                         ;
                                  6E  DD 000A0       PUSHL    DESC_COPY                                  ; 0656
     00000000G  00                01  FB 000A2       CALLS    #1, DBG$PREVLOC                            ;
           0C   BC                50  D0 000A9       MOVL     R0, @RETURNED_VALUE                        ;
                1B                57  E9 000AD 9$:   BLBC     NEXTLOC_FLAG, 10$                          ; 0658
                                  7E  D4 000B0       CLRL     -(SP)                                      ; 0667
                           14     AC  DD 000B2       PUSHL    MESSAGE_VECT                               ; 0669
                           08     AE  9F 000B5       PUSHAB   DESC_COPY                                  ; 0667
                           0C     A2  DD 000B8       PUSHL    12(SYM_PTR)                                ;
                6B                04  FB 000BB       CALLS    #4, DBG$NCOPY_DESC                         ;
                                  6E  DD 000BE       PUSHL    DESC_COPY                                  ; 0671
     00000000G  00                01  FB 000C0       CALLS    #1, DBG$NEXTLOC                            ;
           0C   BC                50  D0 000C7       MOVL     R0, @RETURNED_VALUE                        ;
                50                01  D0 000CB 10$:  MOVL     #1, R0                                     ; 0674
                                  04     000CE       RET                                                 ;
                52                62  D0 000CF 11$:  MOVL     (SYM_PTR), SYM_PTR                         ; 0679
                                  93  11 000D2       BRB      6$                                         ; 0620
                53                63  D0 000D4 12$:  MOVL     (HEAD_PTR), HEAD_PTR                       ; 0684
                                  88  11 000D7       BRB      5$                                         ; 0614
  FF69       55                01 02  F1 000D9 13$:  ACBL     #2, #1, I, 3$                              ; 0599
                                  07
                                  64  91 000DF       CMPB     (R4), #7                                   ; 0695
                                  2A  12 000E2       BNEQ     15$                                        ;
        20   AA        01     A4  07  29 000E4       CMPC3    #7, 1(R4), P.AAG                           ; 0698
                                  0D  12 000EA       BNEQ     14$                                        ;
                   00028808       8F  DD 000EC       PUSHL    #165896                                    ; 0700
     00000000G  00                01  FB 000F2       CALLS    #1, LIB$SIGNAL                             ;
        27   AA        01     A4  07  29 000F9 14$:  CMPC3    #7, 1(R4), P.AAH                           ; 0701
                                  0D  12 000FF       BNEQ     15$                                        ;
                   000287E0       8F  DD 00101       PUSHL    #165856                                    ; 0703
     00000000G  00                01  FB 00107       CALLS    #1, LIB$SIGNAL                             ;
                                  50  D4 0010E 15$:  CLRL     R0                                         ; 0705
                                  04     00110       RET                                                 ; 0707

; Routine Size:  273 bytes,    Routine Base:  DBG$CODE + 00CC
```

```
;  580     0708  1 GLOBAL ROUTINE dbg$def_sym_remove (symbol_name, global_flag,
;  581     0709  1                                     found_flag, message_vect) =
;  582     0710  1 !++
;  583     0711  1 ! Functional Description
;  584     0712  1 !
;  585     0713  1 !       This routine removes a symbol from DEBUG's internal DEFINE symbol
;  586     0714  1 !       table. It is given the symbol's name and an
;  587     0715  1 !       indication of whether the symbol is local or global.
;  588     0716  1 !       It sets FOUND_FLAG to true if it found an occurence of the symbol
;  589     0717  1 !       to be removed; false otherwise.
;  590     0718  1 !
;  591     0719  1 ! Routine Inputs
;  592     0720  1 !
;  593     0721  1 !       SYMBOL_NAME -   Points to a counted string with the symbol name.
;  594     0722  1 !       GLOBAL_FLAG -   TRUE if the symbol is global, FALSE otherwise.
;  595     0723  1 !       FOUND_FLAG -    This is the address of a flag to be set to TRUE if
;  596     0724  1 !                           an occurence of the symbol was found and removed.
;  597     0725  1 !       MESSAGE_VECT -  The address of an error message vector.
;  598     0726  1 !
;  599     0727  1 ! Routine Outputs
;  600     0728  1 !
;  601     0729  1 !       FOUND_FLAG -    This is the address of a flag to be set to TRUE if
;  602     0730  1 !                           an occurence of the symbol was found and removed.
;  603     0731  1 !
;  604     0732  1 ! Routine Value
;  605     0733  1 !
;  606     0734  1 !       An unsigned longword completion code:
;  607     0735  1 !
;  608     0736  1 !       STS$K_SUCCESS - Success. Symbol addition was successful.
;  609     0737  1 !       STS$K_SEVERE -  Failure. Message argument vector is constructed
;  610     0738  1 !                           with an indication of the error.
;  611     0739  1 !
;  612     0740  1 !--
;  613     0741  2     BEGIN
;  614     0742  2
;  615     0743  2     LOCAL
;  616     0744  2         head_ptr: REF define$header,    ! Points to the head of the
;  617     0745  2                                         !    appropriate DEFINE list
;  618     0746  2         name: REF VECTOR [,BYTE],       ! Points to a counted string with
;  619     0747  2                                         !    the DEFINE symbol name
;  620     0748  2         next_sym_ptr: REF define$entry, ! Points to a DEFINE entry
;  621     0749  2         prev_sym_ptr: REF define$entry, ! Points to a DEFINE entry
;  622     0750  2         sym_ptr: REF define$entry;      ! Points to a DEFINE entry
;  623     0751  2
;  624     0752  2     ! Initialize found_flag.
;  625     0753  2     !
;  626     0754  2     .found_flag = FALSE;
;  627     0755  2
;  628     0756  2     !++
;  629     0757  2     ! Search for an occurence of the symbol.
;  630     0758  2     !--
;  631     0759  2
;  632     0760  2     ! Set up pointer to the appropriate define list.
;  633     0761  2     !
;  634     0762  2     IF .global_flag
;  635     0763  2     THEN
;  636     0764  2         head_ptr = .dbg$gl_global_define_ptr
```

```
;  637          0765  2      ELSE
;  638          0766  2          head_ptr = .dbg$gl_local_define_ptr;
;  639          0767  2
;  640          0768  2      ! Loop through the define lists.
;  641          0769  2      !
;  642          0770  2      WHILE .head_ptr NEQ 0 DO
;  643          0771  3          BEGIN
;  644          0772  3
;  645          0773  3          ! Walk through the list searching for the given name.
;  646          0774  3          !
;  647          0775  3          prev_sym_ptr = 0;
;  648          0776  3          sym_ptr = .head_ptr [def$a_define_list];
;  649          0777  3          WHILE .sym_ptr NEQ 0 DO
;  650          0778  4              BEGIN
;  651          0779  4              next_sym_ptr = .sym_ptr [def$a_entry_next_link];
;  652          0780  4              name = .sym_ptr [def$a_name];
;  653          0781  4              IF name_match (.symbol_name, .name)
;  654          0782  4              THEN
;  655          0783  5                  BEGIN
;  656          0784  5
;  657          0785  5                  !++
;  658          0786  5                  ! If we find a match then remove the entry.
;  659          0787  5                  !--
;  660          0788  5
;  661          0789  5                  ! First free up the space taken up by the entry.
;  662          0790  5                  !
;  663          0791  5                  IF NOT free_entry (.sym_ptr, .message_vect)
;  664          0792  5                  THEN
;  665          0793  5                      RETURN sts$k_severe;
;  666          0794  5
;  667          0795  5                  ! Unlink the entry.
;  668          0796  5                  !
;  669          0797  5                  IF .prev_sym_ptr EQL 0
;  670          0798  5                  THEN
;  671          0799  5
;  672          0800  5                      ! First entry
;  673          0801  5                      !
;  674          0802  5                      head_ptr [def$a_define_list] = .next_sym_ptr
;  675          0803  5
;  676          0804  5                  ELSE
;  677          0805  5
;  678          0806  5                      ! Not first entry.
;  679          0807  5                      !
;  680          0808  5                      prev_sym_ptr [def$a_entry_next_link] = .next_sym_ptr;
;  681          0809  5
;  682          0810  5                  IF .next_sym_ptr NEQ 0
;  683          0811  5                  THEN
;  684          0812  5
;  685          0813  5                      ! Not last entry
;  686          0814  5                      !
;  687          0815  5                      next_sym_ptr [def$a_entry_prev_link] = .prev_sym_ptr;
;  688          0816  5
;  689          0817  5                  ! Free up the space.
;  690          0818  5                  !
;  691          0819  5                  dbg$rel_memory (.sym_ptr);
;  692          0820  5                  sym_ptr = .prev_sym_ptr;
;  693          0821  5
```

```
;  694          0822  5                ! Set found_flag to true.
;  695          0823  5                !
;  696          0824  5                .found_flag = TRUE;
;  697          0825  4                END;
;  698          0826  4
;  699          0827  4            ! Set up for the next time around the loop.
;  700          0828  4            !
;  701          0829  4            prev_sym_ptr = .sym_ptr;
;  702          0830  4            sym_ptr = .next_sym_ptr;
;  703          0831  3            END; ! inner WHILE loop
;  704          0832  3
;  705          0833  3        ! Set up for next time around outer loop.
;  706          0834  3        !
;  707          0835  3        head_ptr = .head_ptr [def$a_next_link];
;  708          0836  2        END; ! outer WHILE loop
;  709          0837  2
;  710          0838  2    ! We are all done. Return success.
;  711          0839  2    !
;  712          0840  2    RETURN sts$k_success;
;  713          0841  1    END; ! dbg$def_sym_remove


                                007C 00000          .ENTRY  DBG$DEF_SYM_REMOVE, Save R2,R3,R4,R5,R6      ; 0708
                          0C    BC   D4 00002          CLRL    @FOUND_FLAG                                  ; 0754
                    09          08   AC  E9 00005          BLBC    GLOBAL_FLAG, 1$                              ; 0762
                    52 00000000'      EF  D0 00009          MOVL    DBG$GL_GLOBAL_DEFINE_PTR, HEAD_PTR           ; 0764
                                      07  11 00010          BRB     2$                                           ;
                    52 00000000'      EF  D0 00012 1$:      MOVL    DBG$GL_LOCAL_DEFINE_PTR, HEAD_PTR            ; 0766
                                      5D  13 00019 2$:      BEQL    10$                                          ; 0770
                                      55  D4 0001B          CLRL    PREV_SYM_PTR                                 ; 0775
                          53          08  A2 D0 0001D          MOVL    8(HEAD_PTR), SYM_PTR                         ; 0776
                                      50  13 00021 3$:      BEQL    9$                                           ; 0777
                          54              63 D0 00023          MOVL    (SYM_PTR), NEXT_SYM_PTR                      ; 0779
                          56          08  A3 D0 00026          MOVL    8(SYM_PTR), NAME                             ; 0780
                                      56  DD 0002A          PUSHL   NAME                                         ; 0781
                                      04  AC DD 0002C          PUSHL   SYMBOL_NAME                                  ;
                    0000V CF              02 FB 0002F          CALLS   #2, NAME_MATCH                               ;
                                      34  50 E9 00034          BLBC    R0, 8$                                       ;
                                      10  AC DD 00037          PUSHL   MESSAGE_VECT                                 ; 0791
                                      53  DD 0003A          PUSHL   SYM_PTR                                      ;
                    0000V CF              02 FB 0003C          CALLS   #2, FREE_ENTRY                               ;
                                      04  50 E8 00041          BLBS    R0, 4$                                       ;
                                      50  04 D0 00044          MOVL    #4, R0                                       ; 0793
                                          04 00047          RET                                                  ;
                                      55  D5 00048 4$:      TSTL    PREV_SYM_PTR                                 ; 0797
                                      06  12 0004A          BNEQ    5$                                           ;
                          08          A2  54 D0 0004C          MOVL    NEXT_SYM_PTR, 8(HEAD_PTR)                    ; 0802
                                      03  11 00050          BRB     6$                                           ;
                          65              54 D0 00052 5$:      MOVL    NEXT_SYM_PTR, (PREV_SYM_PTR)                 ; 0808
                                      04  13 00055 6$:      BEQL    7$                                           ; 0810
                          04          A4  55 D0 00057          MOVL    PREV_SYM_PTR, 4(NEXT_SYM_PTR)                ; 0815
                                      53  DD 0005B 7$:      PUSHL   SYM_PTR                                      ; 0819
                    00000000G         00  01 FB 0005D          CALLS   #1, DBG$REL_MEMORY                           ;
                                      53  55 D0 00064          MOVL    PREV_SYM_PTR, SYM_PTR                        ; 0820
```

```
                OC  BC          01  D0 00067          MOVL    #1, @FOUND_FLAG                 ; 0824
                    55          53  D0 0006B 8$:      MOVL    SYM_PTR, PREV_SYM_PTR           ; 0829
                    53          54  D0 0006E          MOVL    NEXT_SYM_PTR, SYM_PTR           ; 0830
                                AE  11 00071          BRB     3$                              ; 0777
                    52          62  D0 00073 9$:      MOVL    (HEAD_PTR), HEAD_PTR            ; 0835
                                A1  11 00076          BRB     2$                              ; 0770
                    50          01  D0 00078 10$:     MOVL    #1, R0                          ; 0840
                                    04 0007B          RET                                     ; 0841

; Routine Size:  124 bytes,    Routine Base:  DBG$CODE + 01DD
```

```
;  715          0842  1 GLOBAL ROUTINE dbg$def_sym_remove_all (global_flag, message_vect) =
;  716          0843  1 !++
;  717          0844  1 ! Functional Description
;  718          0845  1 !
;  719          0846  1 !       This routine removes a all symbols from an internal DEFINE symbol
;  720          0847  1 !       table. global_flag indicates whether to remove local or global
;  721          0848  1 !       symbols.
;  722          0849  1 !
;  723          0850  1 ! Routine Inputs
;  724          0851  1 !
;  725          0852  1 !       GLOBAL_FLAG -   TRUE for global, FALSE otherwise.
;  726          0853  1 !       MESSAGE_VECT -  The address of an error message vector.
;  727          0854  1 !
;  728          0855  1 ! Routine Value
;  729          0856  1 !
;  730          0857  1 !       An unsigned longword completion code:
;  731          0858  1 !
;  732          0859  1 !       STS$K_SUCCESS - Success. Symbol addition was successful.
;  733          0860  1 !       STS$K_SEVERE -  Failure. Message argument vector is constructed
;  734          0861  1 !                       with an indication of the error.
;  735          0862  1 !
;  736          0863  1 !--
;  737          0864  2     BEGIN
;  738          0865  2
;  739          0866  2     LOCAL
;  740          0867  2         head_ptr: REF define$header,    ! Points to the head of the
;  741          0868  2                                         !     appropriate DEFINE list
;  742          0869  2         next_sym_ptr: REF define$entry, ! Points to a DEFINE entry
;  743          0870  2         sym_ptr: REF define$entry;      ! Points to a DEFINE entry
;  744          0871  2
;  745          0872  2
;  746          0873  2     ! Set up pointer to the appropriate define list.
;  747          0874  2     !
;  748          0875  2     IF .global_flag
;  749          0876  2     THEN
;  750          0877  2         head_ptr = .dbg$gl_global_define_ptr
;  751          0878  2     ELSE
;  752          0879  2         head_ptr = .dbg$gl_local_define_ptr;
;  753          0880  2
;  754          0881  2     ! Loop through the define lists.
;  755          0882  2     !
;  756          0883  2     WHILE .head_ptr NEQ 0 DO
;  757          0884  3         BEGIN
;  758          0885  3
;  759          0886  3         ! Walk through the list searching for the given name.
;  760          0887  3         !
;  761          0888  3         sym_ptr = .head_ptr [def$a_define_list];
;  762          0889  3         WHILE .sym_ptr NEQ 0 DO
;  763          0890  4             BEGIN
;  764          0891  4             next_sym_ptr = .sym_ptr [def$a_entry_next_link];
;  765          0892  4
;  766          0893  4             ! First free up the space taken up by the entry.
;  767          0894  4             !
;  768          0895  4             IF NOT free_entry (.sym_ptr, .message_vect)
;  769          0896  4             THEN
;  770          0897  4                 RETURN sts$k_severe;
;  771          0898  4
```

```
;  772          0899  4          ! Free up the space.
;  773          0900  4          !
;  774          0901  4          dbg$rel_memory (.sym_ptr);
;  775          0902  4
;  776          0903  4          ! Set up for the next time around the loop.
;  777          0904  4          !
;  778          0905  4          sym_ptr = .next_sym_ptr;
;  779          0906  4          END; ! inner While loop
;  780          0907  4
;  781          0908  3      ! Zero out the pointer to the define list.
;  782          0909  3      !
;  783          0910  3      head_ptr [def$a_define_list] = 0;
;  784          0911  3
;  785          0912  3      ! Set up for next time around outer loop.
;  786          0913  3      !
;  787          0914  3      head_ptr = .head_ptr [def$a_next_link];
;  788          0915  2      END; ! outer WHILE loop
;  789          0916  2
;  790          0917  2  ! We are all done. Return success.
;  791          0918  2  !
;  792          0919  2  RETURN sts$k_success;
;  793          0920  1  END; ! dbg$def_sym_remove_all


                                    001C 00000          .ENTRY  DBG$DEF_SYM_REMOVE_ALL, Save R2,R3,R4          ; 0842
                            09  04    AC   E9 00002       BLBC    GLOBAL_FLAG, 1$                                ; 0875
                        52  00000000' EF   D0 00006       MOVL    DBG$GL_GLOBAL_DEFINE_PTR, HEAD_PTR             ; 0877
                                    07   11 0000D       BRB     2$                                             ;
                        52  00000000' EF   D0 0000F 1$:    MOVL    DBG$GL_LOCAL_DEFINE_PTR, HEAD_PTR              ; 0879
                                    30   13 00016 2$:    BEQL    6$                                             ; 0883
                            53      08   A2   D0 00018       MOVL    8(HEAD_PTR), SYM_PTR                           ; 0888
                                    22   13 0001C 3$:    BEQL    5$                                             ; 0889
                            54          63   D0 0001E       MOVL    (SYM_PTR), NEXT_SYM_PTR                        ; 0891
                                    08   AC   DD 00021       PUSHL   MESSAGE_VECT                                   ; 0895
                                    53   DD 00024       PUSHL   SYM_PTR                                        ;
                          0000V     CF   02   FB 00026       CALLS   #2, FREE_ENTRY                                 ;
                                    04   50   E8 0002B       BLBS    R0, 4$                                         ;
                                    50   04   D0 0002E       MOVL    #4, R0                                         ; 0897
                                         04   00031       RET                                                    ;
                                    53   DD 00032 4$:    PUSHL   SYM_PTR                                        ; 0901
                    00000000G       00   01   FB 00034       CALLS   #1, DBG$REL_MEMORY                             ;
                                    53   54   D0 0003B       MOVL    NEXT_SYM_PTR, SYM_PTR                          ; 0905
                                    DC   11 0003E       BRB     3$                                             ; 0889
                            08      A2   D4 00040 5$:    CLRL    8(HEAD_PTR)                                    ; 0910
                            52          62   D0 00043       MOVL    (HEAD_PTR), HEAD_PTR                           ; 0914
                                    CE   11 00046       BRB     2$                                             ; 0883
                            50          01   D0 00048 6$:    MOVL    #1, R0                                         ; 0919
                                         04   0004B       RET                                                    ; 0920

; Routine Size:  76 bytes,    Routine Base:  DBG$CODE + 0259
```

```
;  795          0921  1 GLOBAL ROUTINE dbg$def_pr_entry (message_vect) =
;  796          0922  1 !++
;  797          0923  1 ! Routine Description
;  798          0924  1 !
;  799          0925  1 !       This routine is called at entry to a command procedure. It allocates
;  800          0926  1 !       a new define list for the procedure.
;  801          0927  1 !
;  802          0928  1 ! Inputs
;  803          0929  1 !
;  804          0930  1 !       message_vect    -       Error message vector.
;  805          0931  1 !
;  806          0932  1 ! Outputs
;  807          0933  1 !
;  808          0934  1 !       The local define list is modified.
;  809          0935  1 !       A status code is returned. This is one of:
;  810          0936  1 !       sts$k_success - success.
;  811          0937  1 !       sts$k_severe - failure.
;  812          0938  1 !--
;  813          0939  2     BEGIN
;  814          0940  2
;  815          0941  2     MAP
;  816          0942  2         dbg$gl_local_define_ptr: REF define$header;
;  817          0943  2
;  818          0944  2     LOCAL
;  819          0945  2         head_ptr: REF define$header;
;  820          0946  2
;  821          0947  2     ! Save away a pointer to the current top-level local define header.
;  822          0948  2     !
;  823          0949  2     head_ptr = .dbg$gl_local_define_ptr;
;  824          0950  2
;  825          0951  2     ! Allocate space for a new header block.
;  826          0952  2     !
;  827          0953  2     dbg$gl_local_define_ptr = dbg$get_memory (dbg$k_define_header_size_w);
;  828          0954  2
;  829          0955  2     ! Fill in the fields of the newly-allocated header block.
;  830          0956  2     !
;  831          0957  2     dbg$gl_local_define_ptr [def$a_next_link] = .head_ptr;
;  832          0958  2     dbg$gl_local_define_ptr [def$a_prev_link] = 0;
;  833          0959  2     dbg$gl_local_define_ptr [def$a_define_list] = 0;
;  834          0960  2
;  835          0961  2     ! Fill in the back pointer for the second entry.
;  836          0962  2     !
;  837          0963  2     head_ptr [def$a_prev_link] = .dbg$gl_local_define_ptr;
;  838          0964  2
;  839          0965  2     ! Increment the count of the number of levels of procedure nesting.
;  840          0966  2     ! Check for exceeding the maximum nesting level. If so, we will print
;  841          0967  2     ! an error and abort the processing of "@". Even after aborting
;  842          0968  2     ! the "@" processing, DBG$DEF_PR_EXIT will get called to un-do
;  843          0969  2     ! the work we have done in this routine to this point.
;  844          0970  2     !
;  845          0971  2     dbg$gl_pr_nest_level = .dbg$gl_pr_nest_level + 1;
;  846          0972  2     IF .dbg$gl_pr_nest_level GTR dbg$k_max_pr_nesting
;  847          0973  2     THEN
;  848          0974  3         BEGIN
;  849          0975  3         .message_vect = dbg$nmake_arg_vect (dbg$_provrflow);
;  850          0976  3         RETURN sts$k_severe;
;  851          0977  2         END;
```

```
;  852          0978  2     dbg$gl_param_count [.dbg$gl_pr_nest_level] = 0;
;  853          0979  2
;  854          0980  2     RETURN sts$k_success;
;  855          0981  1     END; ! dbg$def_pr_entry



                                    001C 00000          .ENTRY  DBG$DEF_PR_ENTRY, Save R2,R3,R4                 ; 0921
                            54 00000000'  EF   9E 00002  MOVAB   DBG$GL_LOCAL_DEFINE_PTR, R4                     ;
                            53 00000000G  00   9E 00009  MOVAB   DBG$GL_PR_NEST_LEVEL, R3                       ;
                            52            64   D0 00010  MOVL    DBG$GL_LOCAL_DEFINE_PTR, HEAD_PTR             ; 0949
                                          03   DD 00013  PUSHL   #3                                            ; 0953
                  00000000G   00          01   FB 00015  CALLS   #1, DBG$GET_MEMORY                           ;
                              64          50   D0 0001C  MOVL    R0, DBG$GL_LOCAL_DEFINE_PTR                   ;
                              60          52   D0 0001F  MOVL    HEAD_PTR, (R0)                                ; 0957
                                    04    A0   7C 00022  CLRQ    4(R0)                                         ; 0958
                        04    A2          50   D0 00025  MOVL    R0, 4(HEAD_PTR)                               ; 0963
                                          63   D6 00029  INCL    DBG$GL_PR_NEST_LEVEL                          ; 0971
                  00000000G   8F          63   D1 0002B  CMPL    DBG$GL_PR_NEST_LEVEL, #DBG$K_MAX_PR_NESTING   ; 0972
                                          15   15 00032  BLEQ    1$                                            ;
                              00028E70    8F   DD 00034  PUSHL   #167536                                       ; 0975
                  00000000G   00          01   FB 0003A  CALLS   #1, DBG$NMAKE_ARG_VECT                        ;
                        04    BC          50   D0 00041  MOVL    R0, @MESSAGE_VECT                             ;
                              50          04   D0 00045  MOVL    #4, R0                                        ; 0976
                                          04   00048  RET                                                      ;
                              50          63   D0 00049  1$:    MOVL    DBG$GL_PR_NEST_LEVEL, R0              ; 0978
                        00000000G0040     D4   0004C  CLRL    DBG$GL_PARAM_COUNT[R0]                        ;
                              50          01   D0 00053  MOVL    #1, R0                                        ; 0980
                                          04   00056  RET                                                      ; 0981

; Routine Size:  87 bytes,    Routine Base:  DBG$CODE + 02A5
```

```
;  857          0982  1 GLOBAL ROUTINE dbg$def_pr_exit (message_vect) =
;  858          0983  1 !++
;  859          0984  1 ! Routine Description
;  860          0985  1 !
;  861          0986  1 !       This routine is called at the exit from a command procedure.
;  862          0987  1 !       It removes the define list for that procedure.
;  863          0988  1 !
;  864          0989  1 ! Inputs
;  865          0990  1 !
;  866          0991  1 !       message_vect    -       An error message vector
;  867          0992  1 !
;  868          0993  1 ! Outputs
;  869          0994  1 !
;  870          0995  1 !       The local define list is modified.
;  871          0996  1 !       A status code is returned:
;  872          0997  1 !       sts$k_success - success
;  873          0998  1 !       sts$k_severe - failure
;  874          0999  1 !--
;  875          1000  2     BEGIN
;  876          1001  2
;  877          1002  2     MAP
;  878          1003  2         dbg$gl_local_define_ptr: REF define$header;
;  879          1004  2
;  880          1005  2     LOCAL
;  881          1006  2         head_ptr: REF define$header,    ! Saved pointer to head of list.
;  882          1007  2         next_sym_ptr: REF define$entry, ! Points to a define entry
;  883          1008  2         sym_ptr: REF define$entry;      ! Points to a define entry
;  884          1009  2
;  885          1010  2     ! Decrement the count of levels of procedure nesting.
;  886          1011  2     !
;  887          1012  2     dbg$gl_pr_nest_level = .dbg$gl_pr_nest_level - 1;
;  888          1013  2     IF .dbg$gl_pr_nest_level LSS 0
;  889          1014  2     THEN
;  890          1015  3         BEGIN
;  891          1016  3         $DBG_ERROR('DBGDEFINE\DBG$DEF_PR_EXIT');
;  892          1017  2         END;
;  893          1018  2
;  894          1019  2     ! Save away a pointer to the top header block.
;  895          1020  2     !
;  896          1021  2     head_ptr = .dbg$gl_local_define_ptr;
;  897          1022  2
;  898          1023  2     ! Cut out the first entry.
;  899          1024  2     !
;  900          1025  2     dbg$gl_local_define_ptr = .dbg$gl_local_define_ptr [def$a_next_link];
;  901          1026  2
;  902          1027  2     !
;  903          1028  2     ! Free up the space being occupied by the list.
;  904          1029  2     !
;  905          1030  2
;  906          1031  2     ! Obtain a pointer to the define list.
;  907          1032  2     !
;  908          1033  2     sym_ptr = .head_ptr [def$a_define_list];
;  909          1034  2
;  910          1035  2     ! Free up space occupied by the header block.
;  911          1036  2     !
;  912          1037  2     dbg$rel_memory (.head_ptr);
;  913          1038  2
```

```
;  914    1039 2      ! Walk through the list freeing up each entry.
;  915    1040 2      !
;  916    1041 2      WHILE .sym_ptr NEQ 0 DO
;  917    1042 3          BEGIN
;  918    1043 3
;  919    1044 3          ! Obtain a pointer to the next list entry.
;  920    1045 3          !
;  921    1046 3          next_sym_ptr = .sym_ptr [def$a_entry_next_link];
;  922    1047 3
;  923    1048 3          ! Free up the space taken up by the name and value.
;  924    1049 3          !
;  925    1050 3          IF NOT free_entry (.sym_ptr, .message_vect)
;  926    1051 3          THEN
;  927    1052 3              RETURN sts$k_severe;
;  928    1053 3
;  929    1054 3          ! Free up the space occupied by the entry.
;  930    1055 3          !
;  931    1056 3          dbg$rel_memory (.sym_ptr);
;  932    1057 3
;  933    1058 3          ! Set up for next time around.
;  934    1059 3          !
;  935    1060 3          sym_ptr = .next_sym_ptr;
;  936    1061 2          END;
;  937    1062 2
;  938    1063 2      RETURN sts$k_success;
;  939    1064 1      END; ! dbg$def_pr_exit


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

24 47 42 44 5C 45 4E 49 46 45 44 47 42 44 19 0003E P.AAI:  .ASCII  <25>\DBGDEFINE\<92>\DBG$DEF_PR_EXIT\     ;
            54 49 58 45 5F 52 50 5F 46 45 44 0004D                                                          ;


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

                                   003C 00000           .ENTRY  DBG$DEF_PR_EXIT, Save R2,R3,R4,R5            ; 0982
                  55 00000000G  00  9E 00002            MOVAB   DBG$REL_MEMORY, R5                          ;
                  54 00000000'  EF  9E 00009            MOVAB   DBG$GL_LOCAL_DEFINE_PTR, R4                 ;
                  15 00000000G  00  F4 00010            SOBGEQ  DBG$GL_PR_NEST_LEVEL, 1$                    ; 1012
                     00000000'  EF  9F 00017            PUSHAB  P.AAI                                       ; 1016
                                01  DD 0001D            PUSHL   #1                                          ;
                     00028362   8F  DD 0001F            PUSHL   #164706                                     ;
      00000000G  00             03  FB 00025            CALLS   #3, LIB$SIGNAL                              ;
                 50             64  D0 0002C 1$:        MOVL    DBG$GL_LOCAL_DEFINE_PTR, HEAD_PTR           ; 1021
                 74             94  D0 0002F            MOVL    @DBG$GL_LOCAL_DEFINE_PTR, -                 ; 1025
                                                                DBG$GL_LOCAL_DEFINE_PTR                     ;
                 52         08  A0  D0 00032            MOVL    8(HEAD_PTR), SYM_PTR                        ; 1033
                                50  DD 00036            PUSHL   HEAD_PTR                                    ; 1037
                 65             01  FB 00038            CALLS   #1, DBG$REL_MEMORY                          ;
                                52  D5 0003B 2$:        TSTL    SYM_PTR                                     ; 1041
                                1E  13 0003D            BEQL    4$                                          ;
                 53             62  D0 0003F            MOVL    (SYM_PTR), NEXT_SYM_PTR                     ; 1046
                            04  AC  DD 00042            PUSHL   MESSAGE_VECT                                ; 1050
                                52  DD 00045            PUSHL   SYM_PTR                                     ;
```

```
                0000V  CF               02  FB 00047          CALLS   #2, FREE_ENTRY                          ;
                       04               50  E8 0004C          BLBS    R0, 3$                                  ;
                       50               04  D0 0004F          MOVL    #4, R0                                  ; 1052
                                            04 00052          RET                                             ;
                                        52  DD 00053 3$:       PUSHL   SYM_PTR                                 ; 1056
                       65               01  FB 00055          CALLS   #1, DBG$REL_MEMORY                      ;
                       52               53  D0 00058          MOVL    NEXT_SYM_PTR, SYM_PTR                   ; 1060
                                        DE  11 0005B          BRB     2$                                      ; 1041
                       50               01  D0 0005D 4$:       MOVL    #1, R0                                  ; 1063
                                            04 00060          RET                                             ; 1064

; Routine Size:  97 bytes,    Routine Base:  DBG$CODE + 02FC
```

```
;  941          1065  1 GLOBAL ROUTINE dbg$dump_define (string, addr_flag, global_flag, type_flag,
;  942          1066  1                                 found_status, message_vect) =
;  943          1067  1 !++
;  944          1068  1 ! Routine Description
;  945          1069  1 !
;  946          1070  1 !       This routine dumps the DEFINE symbol table. It is called from
;  947          1071  1 !       the SHOW SYM/DEFINED command.
;  948          1072  1 !
;  949          1073  1 ! Inputs
;  950          1074  1 !
;  951          1075  1 !       string          - A string representing the defined symbol
;  952          1076  1 !                         whose definition the users wants to see.
;  953          1077  1 !                         Asterisk may be used as a wild card character.
;  954          1078  1 !       addr_flag       - Says to print the thing that the defined symbol
;  955          1079  1 !                         is bound to.
;  956          1080  1 !       global_flag     - A flag set to TRUE if the user is requesting
;  957          1081  1 !                         globally defined symbols; false otherwise.
;  958          1082  1 !       type_flag       - says to print the kind of defined symbol.
;  959          1083  1 !       found_status    - This is used in the case of SHOW SYMBOL xxx,
;  960          1084  1 !                         where we have previously called the
;  961          1085  1 !                         DBG$STA_SHOSYMBOL routine. This flag is set to:
;  962          1086  1 !                         TRUE: DBG$STA_SHOSYMBOL has already matched some
;  963          1087  1 !                               symbols, so don't signal an error if this
;  964          1088  1 !                               routine does not match anything.
;  965          1089  1 !                         FALSE: DBG$STA_SHOSYMBOL has not matched any symbols.
;  966          1090  1 !       message_vect    - An error message vector.
;  967          1091  1 !
;  968          1092  1 ! Implicit Inputs
;  969          1093  1 !
;  970          1094  1 !       The DEFINE symbol table.
;  971          1095  1 !
;  972          1096  1 ! Output
;  973          1097  1 !
;  974          1098  1 !       The contents of the DEFINE symbol table are displayed at the terminal.
;  975          1099  1 !       A condition code is returned, which is one of:
;  976          1100  1 !       STS$K_SUCCESS   - Routine was successful
;  977          1101  1 !       STS$K_SEVERE    - Routine was not successful. Error message vector
;  978          1102  1 !                         constructed.
;  979          1103  1 !
;  980          1104  1 ! Side Effects
;  981          1105  1 !
;  982          1106  1 !       None.
;  983          1107  1 !--
;  984          1108  2     BEGIN
;  985          1109  2     LOCAL
;  986          1110  2         found_flag,                     ! True if we find any matches
;  987          1111  2         head_ptr: REF define$header,    ! A pointer to a define list header
;  988          1112  2         name: REF VECTOR[,BYTE],        ! Points to a symbol name
;  989          1113  2         sym_ptr: REF define$entry;      ! A pointer to a define list entry
;  990          1114  2
;  991          1115  2     ! Initialize found_flag
;  992          1116  2     !
;  993          1117  2     found_flag = FALSE;
;  994          1118  2
;  995          1119  2     ! Set up a pointer to either the head of the global define table or
;  996          1120  2     ! the head of the local define table.
;  997          1121  2     !
```

```
;  998          1122  2      IF .global_flag
;  999          1123  2      THEN
; 1000          1124  2          head_ptr = .dbg$gl_global_define_ptr
; 1001          1125  2      ELSE
; 1002          1126  2          head_ptr = .dbg$gl_local_define_ptr;
; 1003          1127  2
; 1004          1128  2      ! Loop through the define tables.
; 1005          1129  2      !
; 1006          1130  2      WHILE .head_ptr NEQ 0 DO
; 1007          1131  3          BEGIN
; 1008          1132  3
; 1009          1133  3          ! Loop through the symbols in the define table.
; 1010          1134  3          !
; 1011          1135  3          sym_ptr = .head_ptr [def$a_define_list];
; 1012          1136  3          WHILE .sym_ptr NEQ 0 DO
; 1013          1137  4              BEGIN
; 1014          1138  4
; 1015          1139  4              ! Pick up the name and determine if it is a match.
; 1016          1140  4              !
; 1017          1141  4              name = .sym_ptr [def$a_name];
; 1018          1142  4              IF dbg$wildcard_name_match (.string, .name)
; 1019          1143  4              THEN
; 1020          1144  5                  BEGIN
; 1021          1145  5                  LABEL
; 1022          1146  5                      search_block;
; 1023          1147  5                  LOCAL
; 1024          1148  5                      print_flag,
; 1025          1149  5                      temp_head_ptr: REF define$header,
; 1026          1150  5                      temp_name,
; 1027          1151  5                      temp_sym_ptr: REF define$entry;
; 1028          1152  5
; 1029          1153  5                  found_flag = TRUE;
; 1030          1154  5
; 1031          1155  6 search_block:    BEGIN
; 1032          1156  6
; 1033          1157  6                  ! Check whether we have already printed the name.
; 1034          1158  6                  !
; 1035          1159  6                  IF .global_flag
; 1036          1160  6                  THEN
; 1037          1161  6                      temp_head_ptr = .dbg$gl_global_define_ptr
; 1038          1162  6                  ELSE
; 1039          1163  6                      temp_head_ptr = .dbg$gl_local_define_ptr;
; 1040          1164  6                  WHILE .temp_head_ptr NEQ 0 DO
; 1041          1165  7                      BEGIN
; 1042          1166  7                      temp_sym_ptr = .temp_head_ptr [def$a_define_list];
; 1043          1167  7                      WHILE .temp_sym_ptr NEQ 0 DO
; 1044          1168  8                          BEGIN
; 1045          1169  8                          IF .temp_sym_ptr EQL .sym_ptr
; 1046          1170  8                          THEN
; 1047          1171  9                              BEGIN
; 1048          1172  9                              print_flag = TRUE;
; 1049          1173  9                              LEAVE search_block;
; 1050          1174  8                              END;
; 1051          1175  8                          temp_name = .temp_sym_ptr[def$a_name];
; 1052          1176  8                          IF name_match (.temp_name, .name)
; 1053          1177  8                          THEN
; 1054          1178  9                              BEGIN
```

```
;  1055          1179  9                        print_flag = FALSE;
;  1056          1180  9                        LEAVE search_block;
;  1057          1181  8                        END;
;  1058          1182  8                    temp_sym_ptr = .temp_sym_ptr [def$a_entry_next_link];
;  1059          1183  7                    END;
;  1060          1184  7                temp_head_ptr = .temp_head_ptr [def$a_next_link];
;  1061          1185  6                END;
;  1062          1186  5            END; ! search_block
;  1063          1187  5
;  1064          1188  5            IF .print_flag
;  1065          1189  5            THEN
;  1066          1190  5                dump_entry (.sym_ptr, .addr_flag, .type_flag, .message_vect);
;  1067          1191  4            END;
;  1068          1192  4
;  1069          1193  4        ! Set up for next time around loop.
;  1070          1194  4        !
;  1071          1195  4        sym_ptr = .sym_ptr [def$a_entry_next_link];
;  1072          1196  3        END; ! inner while loop
;  1073          1197  3
;  1074          1198  3    ! Set up for next time around loop.
;  1075          1199  3    !
;  1076          1200  3    head_ptr = .head_ptr [def$a_next_link];
;  1077          1201  2    END; ! outer while loop
;  1078          1202  2
;  1079          1203  2! If we did not find any matches, signal an informational to that effect.
;  1080          1204  2!
;  1081          1205  3IF (NOT .found_status) AND (NOT .found_flag)
;  1082          1206  2THEN
;  1083          1207  2    SIGNAL (dbg$_symnotfnd, 1, .string);
;  1084          1208  2
;  1085          1209  2RETURN sts$k_success;
;  1086          1210  1END; ! of DBG$DUMP_DEFINE


                                07FC 00000          .ENTRY  DBG$DUMP_DEFINE, Save R2,R3,R4,R5,R6,R7,R8,-; 1065
                                                            R9,R10
                    5A 00000000'  EF  9E 00002      MOVAB   DBG$GL_GLOBAL_DEFINE_PTR, R10                 ;
                                  59  D4 00009      CLRL    FOUND_FLAG                                    ; 1117
                    05        0C  AC  E9 0000B      BLBC    GLOBAL_FLAG, 1$                               ; 1122
                    55            6A  D0 0000F      MOVL    DBG$GL_GLOBAL_DEFINE_PTR, HEAD_PTR            ; 1124
                                  04  11 00012      BRB     2$                                            ;
                    55        04  AA  D0 00014 1$:  MOVL    DBG$GL_LOCAL_DEFINE_PTR, HEAD_PTR             ; 1126
                                  74  13 00018 2$:  BEQL    13$                                           ; 1130
                    54        08  A5  D0 0001A      MOVL    8(HEAD_PTR), SYM_PTR                          ; 1135
                                  69  13 0001E 3$:  BEQL    12$                                           ; 1136
                    58        08  A4  D0 00020      MOVL    8(SYM_PTR), NAME                              ; 1141
                                  58  DD 00024      PUSHL   NAME                                          ; 1142
                              04  AC  DD 00026      PUSHL   STRING                                        ;
              0000V CF            02  FB 00029      CALLS   #2, DBG$WILDCARD_NAME_MATCH                   ;
                    53            50  E9 0002E      BLBC    R0, 11$                                       ;
                    59            01  D0 00031      MOVL    #1, FOUND_FLAG                                ; 1153
                    05        0C  AC  E9 00034      BLBC    GLOBAL_FLAG, 4$                               ; 1159
                    53            6A  D0 00038      MOVL    DBG$GL_GLOBAL_DEFINE_PTR, TEMP_HEAD_PTR       ; 1161
                                  04  11 0003B      BRB     5$                                            ;
```

```
                    53        04     AA  D0 0003D 4$:     MOVL    DBG$GL_LOCAL_DEFINE_PTR, TEMP_HEAD_PTR      ; 1163
                                     2E  13 00041 5$:     BEQL    10$                                         ;  1164
                    52        08     A3  D0 00043         MOVL    8(TEMP_HEAD_PTR), TEMP_SYM_PTR              ;  1166
                                     23  13 00047 6$:     BEQL    9$                                          ;  1167
                    54              52  D1 00049         CMPL    TEMP_SYM_PTR, SYM_PTR                       ;  1169
                                    05  12 0004C          BNEQ    7$                                          ;
                    57              01  D0 0004E         MOVL    #1, PRINT_FLAG                              ;  1172
                                    1E  11 00051          BRB     10$                                         ;  1173
                    56        08    A2  D0 00053 7$:     MOVL    8(TEMP_SYM_PTR), TEMP_NAME                  ;  1175
                            0140    8F  BB 00057         PUSHR   #^M<R6,R8>                                  ;  1176
              0000V CF              02  FB 0005B         CALLS   #2, NAME_MATCH                              ;
                    04              50  E9 00060         BLBC    R0, 8$                                      ;
                                    57  D4 00063         CLRL    PRINT_FLAG                                  ;  1179
                                    0A  11 00065         BRB     10$                                         ;  1180
                    52              62  D0 00067 8$:     MOVL    (TEMP_SYM_PTR), TEMP_SYM_PTR                ;  1182
                                    DB  11 0006A         BRB     6$                                          ;  1167
                    53              63  D0 0006C 9$:     MOVL    (TEMP_HEAD_PTR), TEMP_HEAD_PTR              ;  1184
                                    D0  11 0006F         BRB     5$                                          ;  1164
                    10              57  E9 00071 10$:    BLBC    PRINT_FLAG, 11$                             ;  1188
                              18    AC  DD 00074         PUSHL   MESSAGE_VECT                                ;  1190
                              10    AC  DD 00077         PUSHL   TYPE_FLAG                                   ;
                              08    AC  DD 0007A         PUSHL   ADDR_FLAG                                   ;
                                    54  DD 0007D         PUSHL   SYM_PTR                                     ;
              0000V CF              04  FB 0007F         CALLS   #4, DUMP_ENTRY                              ;
                    54              64  D0 00084 11$:    MOVL    (SYM_PTR), SYM_PTR                          ;  1195
                                    95  11 00087         BRB     3$                                          ;  1136
                    55              65  D0 00089 12$:    MOVL    (HEAD_PTR), HEAD_PTR                        ;  1200
                                    8A  11 0008C         BRB     2$                                          ;  1130
                    15        14    AC  E8 0008E 13$:    BLBS    FOUND_STATUS, 14$                           ;  1205
                    12              59  E8 00092         BLBS    FOUND_FLAG, 14$                             ;
                              04    AC  DD 00095         PUSHL   STRING                                      ;  1207
                                    01  DD 00098         PUSHL   #1                                          ;
                          000286BB  8F  DD 0009A         PUSHL   #165563                                     ;
          00000000G   00            03  FB 000A0         CALLS   #3, LIB$SIGNAL                              ;
                    50              01  D0 000A7 14$:    MOVL    #1, R0                                      ;  1209
                                        04 000AA         RET                                                 ;  1210

; Routine Size:  171 bytes,    Routine Base:  DBG$CODE + 035D
```

```
;  1088          1211  1 GLOBAL ROUTINE dbg$nparse_define (input_desc, verb_node, message_vect) =
;  1089          1212  1 !++
;  1090          1213  1 ! Functional Description
;  1091          1214  1 !
;  1092          1215  1 !       This is the top-level parse network for the DEFINE command.
;  1093          1216  1 !
;  1094          1217  1 ! Routine Inputs
;  1095          1218  1 !
;  1096          1219  1 !       input_desc -    A string descriptor for the remaining input.
;  1097          1220  1 !       verb_node -     A pointer to the verb node for DEFINE, which
;  1098          1221  1 !                           will be the top-level node in the command
;  1099          1222  1 !                           execution tree.
;  1100          1223  1 !       message_vect -  An error message vector
;  1101          1224  1 !
;  1102          1225  1 ! Routine Outputs
;  1103          1226  1 !
;  1104          1227  1 !       A command execution tree is constructed starting at the verb
;  1105          1228  1 !       node:
;  1106          1229  1 !
;  1107          1230  1 !        ------      ------      ------
;  1108          1231  1 !       | VERB | -> | NOUN | -> | NOUN | -> ...
;  1109          1232  1 !        ------      ------      ------
;  1110          1233  1 !
;  1111          1234  1 !       The DBG$B_VERB_COMPOSITE field contains an indication of the
;  1112          1235  1 !       "kind" of DEFINE.
;  1113          1236  1 !       In each noun node, all three fields contain information
;  1114          1237  1 !       about the symbol being defined:
;  1115          1238  1 !       DBG$L_NOUN_VALUE -      Points to a counted string with the name of
;  1116          1239  1 !                               the symbol (the left-hand-side of the
;  1117          1240  1 !                               definition).
;  1118          1241  1 !       DBG$L_NOUN_VALUE2 -     Points to some kind of descriptor or
;  1119          1242  1 !                               counted string with the right-hand-side
;  1120          1243  1 !                               of the definition.
;  1121          1244  1 !       DBG$L_ADJECTIVE_PTR -   Contains an indication of whether the
;  1122          1245  1 !                               definition was local (=) or global (==).
;  1123          1246  1 !       The string descriptor is updated to point past the
;  1124          1247  1 !       input that has been parsed. A completion code is returned:
;  1125          1248  1 !       STS$K_SUCCESS - The input was successfully parsed.
;  1126          1249  1 !       STS$K_SEVERE -  There were errors during the parse. An error
;  1127          1250  1 !                       message vector is constructed and returned
;  1128          1251  1 !                       in message_vect.
;  1129          1252  1 !--
;  1130          1253  2     BEGIN
;  1131          1254  2
;  1132          1255  2     MAP
;  1133          1256  2         input_desc : REF BLOCK [,BYTE], ! The string descriptor for the
;  1134          1257  2                                         !    remaining input.
;  1135          1258  2         verb_node : REF dbg$verb_node;  ! The verb node with the DEFINE verb.
;  1136          1259  2
;  1137          1260  2     BIND
;  1138          1261  2         dbg$cs_address =        UPLIT BYTE (7, 'ADDRESS'),
;  1139          1262  2         dbg$cs_command =        UPLIT BYTE (7, 'COMMAND'),
;  1140          1263  2         dbg$cs_key =            UPLIT BYTE (3, 'KEY'),
;  1141          1264  2         dbg$cs_local =          UPLIT BYTE (5, 'LOCAL'),
;  1142          1265  2         dbg$cs_procedure =      UPLIT BYTE (9, 'PROCEDURE'),
;  1143          1266  2         dbg$cs_string =         UPLIT BYTE (6, 'STRING'),
;  1144          1267  2         dbg$cs_value =          UPLIT BYTE (5, 'VALUE'),
```

```
;  1145          1268  2          dbg$cs_comma =          UPLIT BYTE (1, dbg$k_comma),
;  1146          1269  2          dbg$cs_cr =             UPLIT BYTE (1, dbg$k_car_return),
;  1147          1270  2          dbg$cs_dblquote =       UPLIT BYTE (1, dbg$k_dblquote),
;  1148          1271  2          dbg$cs_equal =          UPLIT BYTE (1, dbg$k_equal),
;  1149          1272  2          dbg$cs_left_paren =     UPLIT BYTE (1, dbg$k_left_parenthesis),
;  1150          1273  2          dbg$cs_slash =          UPLIT BYTE (1, dbg$k_slash);
;  1151          1274  2
;  1152          1275  2
;  1153          1276  2      LOCAL
;  1154          1277  2          addr_exp_desc: REF dbg$aed,                 ! Points to an address
;  1155          1278  2                                                      !    expression descriptor
;  1156          1279  2          define_kind,                                ! Holds "kind" of symbol
;  1157          1280  2          first_time,                                 ! True during parsing of first
;  1158          1281  2                                                      !    element of comma list.
;  1159          1282  2          global_flag,                                ! Will be true on ==
;  1160          1283  2          new_noun_node    : REF dbg$noun_node,       ! Another pointer to a noun node
;  1161          1284  2          noun_node        : REF dbg$noun_node,       ! Pointer to a noun node
;  1162          1285  2          ptr: REF VECTOR[,BYTE],                     ! Points into input string
;  1163          1286  2          status,
;  1164          1287  2          stg_desc: dbg$stg_desc,                     ! String descriptor
;  1165          1288  2          symid_list;                                 ! Pointer to symid list
;  1166          1289  2
;  1167          1290  2      ! Special case check for DEFINE/KEY. And save the input descriptor.
;  1168          1291  2      !
;  1169          1292  2
;  1170          1293  2      ch$move(8, .input_desc, stg_desc);
;  1171          1294  2      IF dbg$nmatch (stg_desc, dbg$cs_slash, 1)
;  1172          1295  2      THEN
;  1173          1296  2          IF dbg$nmatch (stg_desc, dbg$cs_key, 1)
;  1174          1297  2          THEN
;  1175          1298  3              BEGIN
;  1176          1299  3              CH$MOVE(8, stg_desc, .input_desc);
;  1177          1300  3              status = dbg$nparse_def_key(.input_desc, .verb_node, .message_vect);
;  1178          1301  3              IF NOT .status
;  1179          1302  3              THEN
;  1180          1303  3                  RETURN .status;
;  1181          1304  3              RETURN sts$k_success;
;  1182          1305  2              END;
;  1183          1306  2
;  1184          1307  2
;  1185          1308  2      ! Initialize global flag. The default is /GLOBAL, so the flag is
;  1186          1309  2      ! initially TRUE.
;  1187          1310  2      !
;  1188          1311  2      global_flag = TRUE;
;  1189          1312  2
;  1190          1313  2      ! First look for the qualifier on the DEFINE command.
;  1191          1314  2      !
;  1192          1315  2      WHILE dbg$nmatch (.input_desc, dbg$cs_slash, 1) DO
;  1193          1316  3          BEGIN
;  1194          1317  3          SELECTONE TRUE OF
;  1195          1318  3              SET
;  1196          1319  3
;  1197          1320  3              [dbg$nmatch (.input_desc, dbg$cs_address, 1)] :
;  1198          1321  4                  BEGIN
;  1199          1322  4                  dbg$set_define_lvl (override_define);
;  1200          1323  4                  dbg$gb_define_ptr [define_only] = define_address;
;  1201          1324  3                  END;
```

```
;  1202          1325  3                [dbg$nmatch (.input_desc, dbg$cs_command, 1)] :
;  1203          1326  3                    BEGIN
;  1204          1327  4                    dbg$set_define_lvl (override_define);
;  1205          1328  4                    dbg$gb_define_ptr [define_only] = define_command;
;  1206          1329  4                    END;
;  1207          1330  4
;  1208          1331
;  1209          1332
;  1210          1333                   ! /GLOBAL is not allowed since it is a no-op.
;  1211          1334                   !
;  1212          1335                   ! [dbg$nmatch (.input_desc, dbg$cs_global, 1)] :
;  1213          1336                   !     global_flag = TRUE;
;  1214          1337
;  1215          1338                   [dbg$nmatch (.input_desc, dbg$cs_local, 1)]:
;  1216          1339                       global_flag = FALSE;
;  1217          1340
;  1218          1341                   ! [dbg$nmatch (.input_desc, dbg$cs_procedure, 1)] :
;  1219          1342                   !   dbg$gb_define_ptr [define_only] = define_procedure;
;  1220          1343
;  1221          1344                   ! [dbg$nmatch (.input_desc, dbg$cs_string, 1)] :
;  1222          1345                   !   dbg$gb_define_ptr [define_only] = define_string;
;  1223          1346
;  1224          1347                   [dbg$nmatch (.input_desc, dbg$cs_value, 1)] :
;  1225          1348  4                    BEGIN
;  1226          1349  4                    dbg$set_define_lvl (override_define);
;  1227          1350  4                    dbg$gb_define_ptr [define_only] = define_value;
;  1228          1351  4                    END;
;  1229          1352
;  1230          1353  3                [OTHERWISE] :
;  1231          1354  3                    report_error;
;  1232          1355  3
;  1233          1356  3                TES;
;  1234          1357  2            END;
;  1235          1358  2
;  1236          1359  2        define_kind = .dbg$gb_define_ptr [define_only];
;  1237          1360  2
;  1238          1361  2
;  1239          1362  2        ! Now that we have decided what kind of DEFINE this is,
;  1240          1363  2        ! put that piece of information in the verb node.
;  1241          1364  2        !
;  1242          1365  2        verb_node [dbg$b_verb_composite] = .define_kind;
;  1243          1366  2
;  1244          1367  2
;  1245          1368  2        ! Now that we have collected the qualifier, we loop through
;  1246          1369  2        ! the list of definitions (e.g,
;  1247          1370  2        ! DEFINE A=B,C=D,E=F
;  1248          1371  2        ! Most of the time there will only be one in the list, but
;  1249          1372  2        ! we are prepared to handle a list here.)
;  1250          1373  2        !
;  1251          1374  2        first_time = TRUE;
;  1252          1375  2        WHILE TRUE DO
;  1253          1376  3            BEGIN
;  1254          1377  3
;  1255          1378  3            ! Check for end-of-line.
;  1256          1379  3            ! It is an error at this point.
;  1257          1380  3            !
;  1258          1381  3            IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
```

```
;  1259     1382  3          THEN
;  1260     1383  4              BEGIN
;  1261     1384  4              .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  1262     1385  4              RETURN sts$k_severe;
;  1263     1386  4              END;
;  1264     1387
;  1265     1388              ! Allocate a noun node to hold the definition
;  1266     1389              !
;  1267     1390              new_noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  1268     1391
;  1269     1392              ! We must link in the noun node. If this is the first definition
;  1270     1393              ! in the list, it is attached to the verb node; otherwise it
;  1271     1394              ! is attached to the previous noun node.
;  1272     1395              !
;  1273     1396              IF .first_time
;  1274     1397              THEN
;  1275     1398  4               BEGIN
;  1276     1399  4               first_time = FALSE;
;  1277     1400  4               verb_node [dbg$l_verb_object_ptr] = .new_noun_node;
;  1278     1401  4               noun_node = .new_noun_node;
;  1279     1402  4               noun_node [dbg$l_noun_link] = 0
;  1280     1403  4               END
;  1281     1404  4
;  1282     1405  3          ELSE
;  1283     1406  4               BEGIN
;  1284     1407  4               noun_node [dbg$l_noun_link] = .new_noun_node;
;  1285     1408  4               noun_node = .new_noun_node;
;  1286     1409  4               noun_node [dbg$l_noun_link] = 0
;  1287     1410  4               END;
;  1288     1411
;  1289     1412
;  1290     1413              ! Now we read the name that is being defined and store a
;  1291     1414              ! pointer to the counted string in the value field of
;  1292     1415              ! the noun node.
;  1293     1416              !
;  1294     1417              IF NOT dbg$nread_name (.input_desc,
;  1295     1418                     noun_node [dbg$l_noun_value],
;  1296     1419                     .message_vect)
;  1297     1420              THEN
;  1298     1421                  RETURN sts$k_severe;
;  1299     1422
;  1300     1423              ! Look for =.
;  1301     1424              !
;  1302     1425              IF NOT dbg$nmatch (.input_desc, dbg$cs_equal, 1)
;  1303     1426              THEN
;  1304     1427                  report_error;
;  1305     1428
;  1306     1429              ! We use the "adjective" field to keep an indication
;  1307     1430              ! of whether the user specified /GLOBAL.
;  1308     1431              !
;  1309     1432              IF .global_flag
;  1310     1433              THEN
;  1311     1434                  noun_node [dbg$l_adjective_ptr] = define_global
;  1312     1435
;  1313     1436              ELSE
;  1314     1437                  noun_node [dbg$l_adjective_ptr] = define_local;
;  1315     1438
```

```
;  1316          1439  3
;  1317          1440  3          ! Now we collect the right-hand-side of the definition.
;  1318          1441  3          ! What we are looking for depends on the qualifier that
;  1319          1442  3          ! was specified.
;  1320          1443  3          !
;  1321          1444  3          CASE .define_kind FROM define_lowest TO define_highest OF
;  1322          1445  3              SET
;  1323          1446  3
;  1324          1447  3              [define_address] :
;  1325          1448  4                  BEGIN
;  1326          1449  4                  LOCAL
;  1327          1450  4                      status,
;  1328          1451  4                      temp_addr_exp_desc;         ! Filled in by AEI
;  1329          1452  4
;  1330          1453  4                  ! Call the address expression interpreter.
;  1331          1454  4                  !
;  1332          1455  4                  status = dbg$nparse_address (.input_desc, temp_addr_exp_desc,
;  1333          1456  4                          .dbg$gb_radix[dbg$b_radix_input],
;  1334          1457  4                          token$k_term_comma, .message_vect);
;  1335          1458  4                  IF .status NEQ sts$k_success AND .status NEQ sts$k_warning
;  1336          1459  4                  THEN
;  1337          1460  4                      RETURN sts$k_severe;
;  1338          1461  4
;  1339          1462  4                  ! Copy the descriptor.
;  1340          1463  4                  !
;  1341          1464  4                  dbg$nget_symid (.temp_addr_exp_desc, symid_list, .message_vect);
;  1342          1465  4                  dbg$ncopy_desc (.temp_addr_exp_desc, addr_exp_desc, .message_vect);
;  1343          1466  4                  dbg$sta_lock_symid (.symid_list);
;  1344          1467  4
;  1345          1468  4                  ! Fill in the "value2" field of the noun node with
;  1346          1469  4                  ! a pointer to the address expression descriptor.
;  1347          1470  4                  !
;  1348          1471  4                  noun_node [dbg$l_noun_value2] = .addr_exp_desc;
;  1349          1472  3                  END;
;  1350          1473  3
;  1351          1474  3              [define_command, define_string] :
;  1352          1475  4                  BEGIN
;  1353          1476  4
;  1354          1477  4                  ! In these cases we want to pick up a quoted string.
;  1355          1478  4                  ! First we pick up the leading quote.
;  1356          1479  4                  !
;  1357          1480  4                  IF NOT dbg$nmatch (.input_desc, dbg$cs_dblquote, 1)
;  1358          1481  4                  THEN
;  1359          1482  4                      report_error;
;  1360          1483  4
;  1361          1484  4                  ! Now call a routine to accept the string.
;  1362          1485  4                  !
;  1363          1486  4                  IF NOT dbg$naccept_string (.input_desc,
;  1364          1487  4                          noun_node [dbg$l_noun_value2],
;  1365          1488  4                          dbg$k_dblquote,
;  1366          1489  4                          TRUE,
;  1367          1490  4                          .message_vect,
;  1368          1491  4                          TRUE)
;  1369          1492  4                  THEN
;  1370          1493  4                      RETURN sts$k_severe;
;  1371          1494  4
;  1372          1495  3                  END;
```

```
;  1373     1496  3
;  1374     1497  3          [define_procedure] :
;  1375     1498  4              BEGIN
;  1376     1499  4
;  1377     1500  4              ! For this case we want to pick up a sequence of DEBUG
;  1378     1501  4              ! commands inside of parenthesis.
;  1379     1502  4              ! First we eat the left paren.
;  1380     1503  4              !
;  1381     1504  4              IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  1382     1505  4              THEN
;  1383     1506  4                  report_error;
;  1384     1507  4
;  1385     1508  4              ! Call the routine which picks up a sequence of
;  1386     1509  4              ! DEBUG commands.
;  1387     1510  4              !
;  1388     1511  4              IF NOT dbg$nsave_break_buffer (.input_desc,
;  1389     1512  4                        noun_node [dbg$l_noun_value2],
;  1390     1513  4                        .message_vect)
;  1391     1514  4              THEN
;  1392     1515  4                  RETURN sts$k_severe;
;  1393     1516  4
;  1394     1517  3              END;
;  1395     1518  3
;  1396     1519  3          [define_value] :
;  1397     1520  4              BEGIN
;  1398     1521  4              LOCAL
;  1399     1522  4                  status,
;  1400     1523  4                  temp_desc: REF dbg$dhead; ! Variable to hold a
;  1401     1524  4                                            ! pointer to a value descriptor
;  1402     1525  4
;  1403     1526  4              ! For this case we just call the expression interpreter
;  1404     1527  4              ! to parse a language expression.
;  1405     1528  4              !
;  1406     1529  4              status = dbg$nparse_expression (.input_desc,
;  1407     1530  4                        .dbg$gb_radix[dbg$b_radix_input],
;  1408     1531  4                        temp_desc, token$k_term_comma, .message_vect);
;  1409     1532  4              IF .status NEQ sts$k_success AND .status NEQ sts$k_warning
;  1410     1533  4              THEN
;  1411     1534  4                  RETURN sts$k_severe;
;  1412     1535  4
;  1413     1536  4              ! If the descriptor is volatile, then we cannot save it.
;  1414     1537  4              ! First try to make it into an ordinary value descriptor.
;  1415     1538  4              ! If that attempt fails, then report an error message saying
;  1416     1539  4              ! that we cannot save the value (this should only happen
;  1417     1540  4              ! for very large values, e.g., long strings).
;  1418     1541  4              !
;  1419     1542  4              IF .temp_desc [dbg$b_dhead_type] EQL dbg$k_v_value_desc
;  1420     1543  4              THEN
;  1421     1544  4                  dbg$prim_to_val(.temp_desc, dbg$k_value_desc, temp_desc);
;  1422     1545  4              IF .temp_desc [dbg$b_dhead_type] EQL dbg$k_v_value_desc
;  1423     1546  4              THEN
;  1424     1547  5                  BEGIN
;  1425     1548  5                  .message_vect = dbg$nmake_arg_vect (dbg$_unasavval,
;  1426     1549  5                      1, .noun_node[dbg$l_noun_value]);
;  1427     1550  5                  RETURN sts$k_severe;
;  1428     1551  4                  END;
;  1429     1552  4
```

```
;  1430          1553  4              ! Copy the descriptor into permanent memory.
;  1431          1554  4              !
;  1432          1555  4              IF dbg$nget_symid(
;  1433          1556  4                              .temp_desc,
;  1434          1557  4                              symid_list,
;  1435          1558  4                              .message_vect)
;  1436          1559  4              THEN
;  1437          1560  4                  IF dbg$ncopy_desc(
;  1438          1561  4                                  .temp_desc,
;  1439          1562  4                                  noun_node [dbg$l_noun_value2],
;  1440          1563  4                                  .message_vect)
;  1441          1564  4                  THEN
;  1442          1565  4                      dbg$sta_lock_symid (.symid_list)
;  1443          1566  4                  ELSE
;  1444          1567  4                      RETURN sts$k_severe
;  1445          1568  4              ELSE
;  1446          1569  4                  RETURN sts$k_severe;
;  1447          1570  4
;  1448          1571  3                  END;
;  1449          1572  3
;  1450          1573  3              [INRANGE,OUTRANGE] :
;  1451          1574  4                  BEGIN
;  1452          1575  4                  report_error;
;  1453          1576  3                  END;
;  1454          1577  3
;  1455          1578  3              TES;
;  1456          1579  3
;  1457          1580  3
;  1458          1581  3          ! Check for exhausted input. If so, exit the loop.
;  1459          1582  3          !
;  1460          1583  3          IF .input_desc [dsc$w_length] EQL 0
;  1461          1584  3          THEN
;  1462          1585  3              EXITLOOP;
;  1463          1586  3
;  1464          1587  3          ! Now check for comma, indicating there are further elements in
;  1465          1588  3          ! the list
;  1466          1589  3          !
;  1467          1590  3          IF NOT dbg$nmatch (.input_desc, dbg$cs_comma, 1)
;  1468          1591  3          THEN
;  1469          1592  3              EXITLOOP;
;  1470          1593  3
;  1471          1594  2          END;                            ! End of WHILE loop
;  1472          1595  2
;  1473          1596  2      RETURN sts$k_success;
;  1474          1597  2
;  1475          1598  1      END;
; INFO#250                 L1:1407
; Referenced LOCAL symbol NOUN_NODE is probably not initialized


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

                                                  07  00058 P.AAJ:  .BYTE   7
                                53  53  45  52  44  44  41  00059          .ASCII  \ADDRESS\                                  ;
                                                  07  00060 P.AAK:  .BYTE   7                                                ;
                                44  4E  41  4D  4D  4F  43  00061          .ASCII  \COMMAND\                                  ;
```

```
                                                03  00068 P.AAL:   .BYTE   3
                                             59 45  4B  00069           .ASCII  \KEY\
                                                05  0006C P.AAM:   .BYTE   5
                                          4C 41 43 4F  4C  0006D           .ASCII  \LOCAL\
                                                09  00072 P.AAN:   .BYTE   9
                                45 52 55 44 45 43 4F 52  50  00073           .ASCII  \PROCEDURE\
                                                06  0007C P.AAO:   .BYTE   6
                                       47 4E 49 52 54  53  0007D           .ASCII  \STRING\
                                                05  00083 P.AAP:   .BYTE   5
                                          45 55 4C 41  56  00084           .ASCII  \VALUE\
                                                2C  01  00089 P.AAQ:   .BYTE   1, 44
                                                0D  01  0008B P.AAR:   .BYTE   1, 13
                                                22  01  0008D P.AAS:   .BYTE   1, 34
                                                3D  01  0008F P.AAT:   .BYTE   1, 61
                                                28  01  00091 P.AAU:   .BYTE   1, 40
                                                2F  01  00093 P.AAV:   .BYTE   1, 47

                                                                DBG$CS_ADDRESS=     P.AAJ
                                                                DBG$CS_COMMAND=     P.AAK
                                                                DBG$CS_KEY=         P.AAL
                                                                DBG$CS_LOCAL=       P.AAM
                                                                DBG$CS_PROCEDURE=   P.AAN
                                                                DBG$CS_STRING=      P.AAO
                                                                DBG$CS_VALUE=       P.AAP
                                                                DBG$CS_COMMA=       P.AAQ
                                                                DBG$CS_CR=          P.AAR
                                                                DBG$CS_DBLQUOTE=    P.AAS
                                                                DBG$CS_EQUAL=       P.AAT
                                                                DBG$CS_LEFT_PAREN=  P.AAU
                                                                DBG$CS_SLASH=       P.AAV


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

                                                OFFC  00000     .ENTRY  DBG$NPARSE_DEFINE, Save R2,R3,R4,R5,R6,R7,- ; 1211
                                                                        R8,R9,R10,R11
                                5B 00000000'  EF   9E  00002    MOVAB   DBG$CS_CR, R11
                                5A 00000000G  00   9E  00009    MOVAB   DBG$NMATCH, R10
                                5E            1C   C2  00010    SUBL2   #28, SP
                                56        04  AC   D0  00013    MOVL    INPUT_DESC, R6                          ; 1293
                          10 AE             66   08   28  00017    MOVC3   #8, (R6), STG_DESC
                                              01   DD  0001C    PUSHL   #1                                      ; 1294
                                          08  AB   9F  0001E    PUSHAB  DBG$CS_SLASH
                                          18  AE   9F  00021    PUSHAB  STG_DESC
                                6A            03   FB  00024    CALLS   #3, DBG$NMATCH
                                25            50   E9  00027    BLBC    R0, 2$
                                              01   DD  0002A    PUSHL   #1                                      ; 1296
                                          DD  AB   9F  0002C    PUSHAB  DBG$CS_KEY
                                          18  AE   9F  0002F    PUSHAB  STG_DESC
                                6A            03   FB  00032    CALLS   #3, DBG$NMATCH
                                17            50   E9  00035    BLBC    R0, 2$
                          66        10  AE   08   28  00038    MOVC3   #8, STG_DESC, (R6)                      ; 1299
                                7E        08  AC   7D  0003D    MOVQ    VERB_NODE, -(SP)                        ; 1300
                                              56   DD  00041    PUSHL   R6
                             0000V  CF        03   FB  00043    CALLS   #3, DBG$NPARSE_DEF_KEY
                                03            50   E9  00048    BLBC    STATUS, 1$                              ; 1301
                                          02C0  31   00  0004B    BRW     41$
```

```
                               04 0004E 1$:     RET                                              ; 1304
              59           01  D0 0004F 2$:     MOVL    #1, GLOBAL_FLAG                          ; 1311
                           01  DD 00052 3$:     PUSHL   #1                                       ; 1315
                     08    AB  9F 00054         PUSHAB  DBG$CS_SLASH                             ;
                           56  DD 00057         PUSHL   R6                                       ;
              6A           03  FB 00059         CALLS   #3, DBG$NMATCH                           ;
              03           50  E8 0005C         BLBS    R0, 4$                                   ;
                   009F        31 0005F         BRW     10$                                      ;
                           01  DD 00062 4$:     PUSHL   #1                                       ; 1320
                     CD    AB  9F 00064         PUSHAB  DBG$CS_ADDRESS                           ;
                           56  DD 00067         PUSHL   R6                                       ;
              6A           03  FB 00069         CALLS   #3, DBG$NMATCH                           ;
              01           50  D1 0006C         CMPL    R0, #1                                   ;
                           15  12 0006F         BNEQ    6$                                       ;
                           02  DD 00071         PUSHL   #2                                       ; 1322
     00000000G  00         01  FB 00073         CALLS   #1, DBG$SET_DEFINE_LVL                   ;
              50 00000000G 00  D0 0007A         MOVL    DBG$GB_DEFINE_PTR, R0                    ; 1323
              60           01  90 00081         MOVB    #1, (R0)                                 ;
                           CC  11 00084 5$:     BRB     3$                                       ; 1317
                           01  DD 00086 6$:     PUSHL   #1                                       ; 1326
                     D5    AB  9F 00088         PUSHAB  DBG$CS_COMMAND                           ;
                           56  DD 0008B         PUSHL   R6                                       ;
              6A           03  FB 0008D         CALLS   #3, DBG$NMATCH                           ;
              01           50  D1 00090         CMPL    R0, #1                                   ;
                           15  12 00093         BNEQ    7$                                       ;
                           02  DD 00095         PUSHL   #2                                       ; 1328
     00000000G  00         01  FB 00097         CALLS   #1, DBG$SET_DEFINE_LVL                   ;
              50 00000000G 00  D0 0009E         MOVL    DBG$GB_DEFINE_PTR, R0                    ; 1329
              60           02  90 000A5         MOVB    #2, (R0)                                 ;
                           A8  11 000A8         BRB     3$                                       ; 1317
                           01  DD 000AA 7$:     PUSHL   #1                                       ; 1338
                     E1    AB  9F 000AC         PUSHAB  DBG$CS_LOCAL                             ;
                           56  DD 000AF         PUSHL   R6                                       ;
              6A           03  FB 000B1         CALLS   #3, DBG$NMATCH                           ;
              01           50  D1 000B4         CMPL    R0, #1                                   ;
                           04  12 000B7         BNEQ    8$                                       ;
                           59  D4 000B9         CLRL    GLOBAL_FLAG                              ; 1339
                           95  11 000BB         BRB     3$                                       ;
                           01  DD 000BD 8$:     PUSHL   #1                                       ; 1347
                     F8    AB  9F 000BF         PUSHAB  DBG$CS_VALUE                             ;
                           56  DD 000C2         PUSHL   R6                                       ;
              6A           03  FB 000C4         CALLS   #3, DBG$NMATCH                           ;
              01           50  D1 000C7         CMPL    R0, #1                                   ;
                           15  12 000CA         BNEQ    9$                                       ;
                           02  DD 000CC         PUSHL   #2                                       ; 1349
     00000000G  00         01  FB 000CE         CALLS   #1, DBG$SET_DEFINE_LVL                   ;
              50 00000000G 00  D0 000D5         MOVL    DBG$GB_DEFINE_PTR, R0                    ; 1350
              60           05  90 000DC         MOVB    #5, (R0)                                 ;
                           A3  11 000DF         BRB     5$                                       ; 1317
                           01  DD 000E1 9$:     PUSHL   #1                                       ; 1353
                   0840    8F  BB 000E3         PUSHR   #^M<R6,R11>                              ;
              6A           03  FB 000E7         CALLS   #3, DBG$NMATCH                           ;
              39           50  E8 000EA         BLBS    R0, 12$                                  ;
                           56  DD 000ED         PUSHL   R6                                       ;
     00000000G  00         01  FB 000EF         CALLS   #1, DBG$NNEXT_WORD                       ;
                           50  DD 000F6         PUSHL   R0                                       ;
     00000000G  00         01  FB 000F8         CALLS   #1, DBG$NSYNTAX_ERROR                    ;
```

```
                                    32  11 000FF          BRB     13$
                          50 00000000G  00  D0 00101 10$:  MOVL    DBG$GB_DEFINE_PTR, R0                   ;  1359
                                    58  60  9A 00108          MOVZBL  (R0), DEFINE_KIND
                                    55  08  AC  D0 0010B      MOVL    VERB_NODE, R5                           ;  1365
                                01  A5  58  90 0010F          MOVB    DEFINE_KIND, 1(R5)
                                    57  01  D0 00113          MOVL    #1, FIRST_TIME                          ;  1374
                                    53  0C  AC  D0 00116      MOVL    MESSAGE_VECT, R3                        ;  1419
                                        01  DD 0011A 11$:     PUSHL   #1                                      ;  1381
                                  0840  8F  BB 0011C          PUSHR   #^M<R6,R11>
                                    6A  03  FB 00120          CALLS   #3, DBG$NMATCH
                                    13  50  E9 00123          BLBC    R0, 14$
                                  000280D0  8F  DD 00126 12$: PUSHL   #164048                                 ;  1384
                          00000000G  00  01  FB 0012C          CALLS   #1, DBG$NMAKE_ARG_VECT
                                0C  BC  50  D0 00133 13$:     MOVL    R0, @MESSAGE_VECT
                                    7C  11 00137              BRB     23$                                     ;  1385
                                        04  DD 00139 14$:     PUSHL   #4                                      ;  1390
                          00000000G  00  01  FB 0013B          CALLS   #1, DBG$GET_TEMPMEM
                                    54  50  D0 00142          MOVL    R0, NEW_NOUN_NODE
                                    08  57  E9 00145          BLBC    FIRST_TIME, 15$                         ;  1396
                                        57  D4 00148          CLRL    FIRST_TIME                              ;  1399
                                08  A5  54  D0 0014A          MOVL    NEW_NOUN_NODE, 8(R5)                    ;  1400
                                        04  11 0014E          BRB     16$                                     ;  1401
                                08  A2  54  D0 00150 15$:     MOVL    NEW_NOUN_NODE, 8(NOUN_NODE)             ;  1407
                                    52  54  D0 00154 16$:     MOVL    NEW_NOUN_NODE, NOUN_NODE                ;  1408
                                  08  A2  D4 00157          CLRL    8(NOUN_NODE)                            ;  1409
                                        0C  BB 0015A          PUSHR   #^M<R2,R3>                              ;  1418
                                        56  DD 0015C          PUSHL   R6
                                0000V   CF  03  FB 0015E      CALLS   #3, DBG$NREAD_NAME
                                    4F  50  E9 00163          BLBC    R0, 23$
                                        01  DD 00166          PUSHL   #1                                      ;  1425
                                    04  AB  9F 00168          PUSHAB  DBG$CS_EQUAL
                                        56  DD 0016B          PUSHL   R6
                                    6A  03  FB 0016D          CALLS   #3, DBG$NMATCH
                                    1E  50  E9 00170          BLBC    R0, 20$
                                    06  59  E9 00173          BLBC    GLOBAL_FLAG, 17$                        ;  1434
                                04  A2  01  D0 00176          MOVL    #1, 4(NOUN_NODE)
                                    03  11 0017A              BRB     18$
                                04  A2  D4 0017C 17$:         CLRL    4(NOUN_NODE)                            ;  1437
                          06            01  58  CF 0017F 18$: CASEL   DEFINE_KIND, #1, #6                    ;  1444
                0084      00AB        0084      0034   00183 19$: .WORD   24$-19$,-
                          000E        000E      00E6   0018B          27$-19$,-
                                                                    30$-19$,-
                                                                    27$-19$,-
                                                                    34$-19$,-
                                                                    20$-19$,-
                                                                    20$-19$
                                        01  DD 00191 20$:     PUSHL   #1                                      ;  1574
                                  0840  8F  BB 00193          PUSHR   #^M<R6,R11>
                                    6A  03  FB 00197          CALLS   #3, DBG$NMATCH
                                    03  50  E9 0019A          BLBC    R0, 21$
                                      00AA  31 0019D          BRW     32$
                                        56  DD 001A0 21$:     PUSHL   R6
                          00000000G  00  01  FB 001A2          CALLS   #1, DBG$NNEXT_WORD
                                        50  DD 001A9          PUSHL   R0
                          00000000G  00  01  FB 001AB          CALLS   #1, DBG$NSYNTAX_ERROR
                                    63  50  D0 001B2 22$:     MOVL    R0, (R3)
                                        74  11 001B5 23$:     BRB     29$
```

```
                                53   DD 001B7 24$:    PUSHL   R3                                      ; 1457
                                01   DD 001B9         PUSHL   #1                                      ; 1455
                          7E 00000000G  00   9A 001BB         MOVZBL  DBG$GB_RADIX, -(SP)             ; 1456
                                0C   AE   9F 001C2         PUSHAB  TEMP_ADDR_EXP_DESC                  ; 1455
                                56   DD 001C5         PUSHL   R6
                00000000G  00        05   FB 001C7         CALLS   #5, DBG$NPARSE_ADDRESS
                           01        50   D1 001CE         CMPL    STATUS, #1                          ; 1458
                                     04   13 001D1         BEQL    25$
                                     50   D5 001D3         TSTL    STATUS
                                     54   12 001D5         BNEQ    29$
                                     53   DD 001D7 25$:    PUSHL   R3                                  ; 1464
                                10   AE   9F 001D9         PUSHAB  SYMID_LIST
                                08   AE   DD 001DC         PUSHL   TEMP_ADDR_EXP_DESC
                00000000G  00        03   FB 001DF         CALLS   #3, DBG$NGET_SYMID
                                     53   DD 001E6         PUSHL   R3                                  ; 1465
                                08   AE   9F 001E8         PUSHAB  ADDR_EXP_DESC
                                08   AE   DD 001EB         PUSHL   TEMP_ADDR_EXP_DESC
                00000000G  00        03   FB 001EE         CALLS   #3, DBG$NCOPY_DESC
                                0C   AE   DD 001F5         PUSHL   SYMID_LIST                          ; 1466
                00000000G  00        01   FB 001F8         CALLS   #1, DBG$STA_LOCK_SYMID
                      0C   A2        04   AE   D0 001FF         MOVL    ADDR_EXP_DESC, 12(NOUN_NODE)    ; 1471
                                00F3   31   00204 26$:    BRW     40$                                 ; 1444
                                     01   DD 00207 27$:    PUSHL   #1                                  ; 1480
                                02   AB   9F 00209         PUSHAB  DBG$CS_DBLQUOTE
                                     56   DD 0020C         PUSHL   R6
                           6A        03   FB 0020E         CALLS   #3, DBG$NMATCH
                           27        50   E9 00211         BLBC    R0, 31$
                                     01   DD 00214         PUSHL   #1                                  ; 1487
                                     53   DD 00216         PUSHL   R3                                  ; 1490
                                     01   DD 00218         PUSHL   #1                                  ; 1487
                                     22   DD 0021A         PUSHL   #34
                                0C   A2   9F 0021C         PUSHAB  12(NOUN_NODE)
                                     56   DD 0021F         PUSHL   R6
                00000000G  00        06   FB 00221         CALLS   #6, DBG$NACCEPT_STRING
                           D9        50   E8 00228 28$:    BLBS    R0, 26$
                                00C8   31   0022B 29$:    BRW     39$                                 ; 1493
                                     01   DD 0022E 30$:    PUSHL   #1                                  ; 1504
                                06   AB   9F 00230         PUSHAB  DBG$CS_LEFT_PAREN
                                     56   DD 00233         PUSHL   R6
                           6A        03   FB 00235         CALLS   #3, DBG$NMATCH
                           1E        50   E8 00238         BLBS    R0, 33$
                                     01   DD 0023B 31$:    PUSHL   #1                                  ; 1505
                                0840   8F   BB 0023D         PUSHR   #^M<R6,R11>
                           6A        03   FB 00241         CALLS   #3, DBG$NMATCH
                           03        50   E8 00244         BLBS    R0, 32$
                                FF56   31   00247         BRW     21$
                      000280D0   8F   DD 0024A 32$:    PUSHL   #164048
                00000000G  00        01   FB 00250         CALLS   #1, DBG$NMAKE_ARG_VECT
                                     6A   11 00257         BRB     37$
                                     53   DD 00259 33$:    PUSHL   R3                                  ; 1513
                                0C   A2   9F 0025B         PUSHAB  12(NOUN_NODE)                       ; 1512
                                     56   DD 0025E         PUSHL   R6
                00000000G  00        03   FB 00260         CALLS   #3, DBG$NSAVE_BREAK_BUFFER
                                     BF   11 00267         BRB     28$
                                     53   DD 00269 34$:    PUSHL   R3                                  ; 1531
                                     01   DD 0026B         PUSHL   #1                                  ; 1529
                                10   AE   9F 0026D         PUSHAB  TEMP_DESC
```

```
                                 7E 00000000G   00   9A 00270          MOVZBL  DBG$GB_RADIX, -(SP)                             ; 1530
                                                56   DD 00277          PUSHL   R6                                              ; 1529
                          00000000G  00          05   FB 00279          CALLS   #5, DBG$NPARSE_EXPRESSION                       ;
                                     01          50   D1 00280          CMPL    STATUS, #1                                      ; 1532
                                                 04   13 00283          BEQL    35$                                             ;
                                                 50   D5 00285          TSTL    STATUS                                          ;
                                                 6D   12 00287          BNEQ    39$                                             ;
00000083   8F       08   BE                     10   ED 00289 35$:      CMPZV   #16, #8, @TEMP_DESC, #131                       ; 1542
                                                 11   12 00293          BNEQ    36$                                             ;
                                           08    AE   9F 00295          PUSHAB  TEMP_DESC                                       ; 1544
                                     7E    7A    8F   9A 00298          MOVZBL  #122, -(SP)                                     ;
                                           10    AE   DD 0029C          PUSHL   TEMP_DESC                                       ;
                          00000000G  00          03   FB 0029F          CALLS   #3, DBG$PRIM_TO_VAL                             ;
00000083   8F       08   BE                     10   ED 002A6 36$:      CMPZV   #16, #8, @TEMP_DESC, #131                       ; 1545
                                                 14   12 002B0          BNEQ    38$                                             ;
                                                 62   DD 002B2          PUSHL   (NOUN_NODE)                                     ; 1549
                                                 01   DD 002B4          PUSHL   #1                                              ; 1548
                                     00028E68    8F   DD 002B6          PUSHL   #167528                                         ;
                          00000000G  00          03   FB 002BC          CALLS   #3, DBG$NMAKE_ARG_VECT                          ;
                                               FEEC   31 002C3 37$:      BRW     22$                                             ;
                                                 53   DD 002C6 38$:      PUSHL   R3                                              ; 1558
                                           10    AE   9F 002C8          PUSHAB  SYMID_LIST                                      ; 1555
                                           10    AE   DD 002CB          PUSHL   TEMP_DESC                                       ; 1556
                          00000000G  00          03   FB 002CE          CALLS   #3, DBG$NGET_SYMID                              ;
                                     1E          50   E9 002D5          BLBC    R0, 39$                                         ;
                                                 53   DD 002D8          PUSHL   R3                                              ; 1563
                                           0C    A2   9F 002DA          PUSHAB  12(NOUN_NODE)                                   ; 1562
                                           10    AE   DD 002DD          PUSHL   TEMP_DESC                                       ;
                          00000000G  00          03   FB 002E0          CALLS   #3, DBG$NCOPY_DESC                              ;
                                     0C          50   E9 002E7          BLBC    R0, 39$                                         ;
                                           0C    AE   DD 002EA          PUSHL   SYMID_LIST                                      ; 1565
                                00000000G  00    01   FB 002ED          CALLS   #1, DBG$STA_LOCK_SYMID                          ;
                                                 04   11 002F4          BRB     40$                                             ;
                                     50          04   D0 002F6 39$:      MOVL    #4, R0                                          ; 1569
                                                 04   002F9             RET                                                     ;
                                                 66   B5 002FA 40$:      TSTW    (R6)                                            ; 1583
                                                 10   13 002FC          BEQL    41$                                             ;
                                                 01   DD 002FE          PUSHL   #1                                              ; 1590
                                           FE    AB   9F 00300          PUSHAB  DBG$CS_COMMA                                    ;
                                                 56   DD 00303          PUSHL   R6                                              ;
                                     6A          03   FB 00305          CALLS   #3, DBG$NMATCH                                  ;
                                     03          50   E9 00308          BLBC    R0, 41$                                         ;
                                               FE0C   31 0030B          BRW     11$                                             ;
                                     50          01   D0 0030E 41$:      MOVL    #1, R0                                          ; 1596
                                                 04   00311             RET                                                     ; 1598

; Routine Size:  786 bytes,    Routine Base:  DBG$CODE + 0408
```

```
;  1477          1599  1 ROUTINE dbg$nparse_def_key (input_desc, verb_node, message_vect) =
;  1478          1600  1 !++
;  1479          1601  1 ! Functional Description
;  1480          1602  1 !
;  1481          1603  1 !       This is the parse network for the DEFINE/KEY command.
;  1482          1604  1 !
;  1483          1605  1 ! Routine Inputs
;  1484          1606  1 !
;  1485          1607  1 !       input_desc      -       A pointer to a string descriptor for the
;  1486          1608  1 !                                 remaining input.
;  1487          1609  1 !       a_verb_node     -       The address of a pointer to the verb node
;  1488          1610  1 !                                 for DEFINE, which will be the top-level
;  1489          1611  1 !                                 node in the command execution tree.
;  1490          1612  1 !       message_vect    -       A pointer to an error message vector.
;  1491          1613  1 !
;  1492          1614  1 ! Routine Outputs
;  1493          1615  1 !
;  1494          1616  1 !       A command execution tree is constructed starting at the verb
;  1495          1617  1 !       node:
;  1496          1618  1 !
;  1497          1619  1 !        ------      ------
;  1498          1620  1 !       ! VERB ! -> ! NOUN !
;  1499          1621  1 !        ------      ------
;  1500          1622  1 !          !                    The DBG$B_VERB_COMPOSITE field contains a
;  1501          1623  1 !        -------                value for the DEFINE/KEY command.
;  1502          1624  1 !       !ADVERB0!
;  1503          1625  1 !        -------               The noun node has the following information:
;  1504          1626  1 !          !
;  1505          1627  1 !        -------               DBG$L_NOUN_VALUE - A pointer to a descriptor
;  1506          1628  1 !       !ADVERB1!                      that contains the key-name.
;  1507          1629  1 !        -------               DBG$L_ADJECTIVE_PTR - A pointer to a
;  1508          1630  1 !          !                          descriptor that contains the
;  1509          1631  1 !          :                          equilalence string for the key.
;  1510          1632  1 !          :
;  1511          1633  1 !        -------               The adverb nodes appear as follows:
;  1512          1634  1 !       !ADVERB5!
;  1513          1635  1 !        -------               DBG$B_ADVERB_LITERAL - Qualifier Code.
;  1514          1636  1 !                              DBG$L_ADVERB_VALUE - Value or location of data
;  1515          1637  1 !                                    for this qualifier.
;  1516          1638  1 !                              DBG$L_ADVERB_LINK - Link to next Adverb-node.
;  1517          1639  1 !
;  1518          1640  1 !       For Adverb1, which contains the qualifier information for the IF_STATE
;  1519          1641  1 !       qualifier, there exists a list of state names with DBG$L_ADVERB_VALUE
;  1520          1642  1 !       pointing at a state_name_node.  This node is defined above, but you
;  1521          1643  1 !       should note that the node consists of 2 fields. 1) A pointer to a
;  1522          1644  1 !       descriptor of the state name and 2) A link field to the next node in
;  1523          1645  1 !       the list.
;  1524          1646  1 !
;  1525          1647  1 !
;  1526          1648  1 !       The string descriptor is updated to point past the
;  1527          1649  1 !       input that has been parsed. A completion code is returned:
;  1528          1650  1 !
;  1529          1651  1 !       STS$K_SUCCESS - The input was successfully parsed.
;  1530          1652  1 !       STS$K_SEVERE -  There were errors during the parse. An error
;  1531          1653  1 !                       message vector is constructed and returned
;  1532          1654  1 !                       in message_vect.
;  1533          1655  1 !--
```

```
;  1534          1656  2     BEGIN
;  1535          1657  2
;  1536          1658  2     MAP
;  1537          1659  2         input_desc              : REF BLOCK [,BYTE],    ! String descriptor
;  1538          1660  2         verb_node               : REF dbg$verb_node;
;  1539          1661  2
;  1540          1662  2     BIND
;  1541          1663  2         dbg$cs_echo             = UPLIT BYTE (4, 'ECHO'),
;  1542          1664  2         dbg$cs_if_state         = UPLIT BYTE (8, 'IF_STATE'),
;  1543          1665  2         dbg$cs_lock_state       = UPLIT BYTE (10, 'LOCK_STATE'),
;  1544          1666  2         dbg$cs_log              = UPLIT BYTE (3, 'LOG'),
;  1545          1667  2         dbg$cs_set_state        = UPLIT BYTE (9, 'SET_STATE'),
;  1546          1668  2         dbg$cs_terminate        = UPLIT BYTE (9, 'TERMINATE'),
;  1547          1669  2         dbg$cs_NO               = UPLIT BYTE ('NO'),
;  1548          1670  2         dbg$cs_left_paren       = UPLIT BYTE (1, dbg$k_left_parenthesis),
;  1549          1671  2         dbg$cs_right_paren      = UPLIT BYTE (1, dbg$k_right_parenthesis),
;  1550          1672  2         dbg$cs_comma            = UPLIT BYTE (1, dbg$k_comma),
;  1551          1673  2         dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
;  1552          1674  2         dbg$cs_dblquote         = UPLIT BYTE (1, dbg$k_dblquote),
;  1553          1675  2         dbg$cs_equal            = UPLIT BYTE (1, dbg$k_equal),
;  1554          1676  2         dbg$cs_slash            = UPLIT BYTE (1, dbg$k_slash);
;  1555          1677  2
;  1556          1678  2     LITERAL
;  1557          1679  2         dbg$k_lowest_qualifier  = 0,    ! These literals correspond to the
;  1558          1680  2         dbg$k_echo              = 0,    ! adverb-nodes in the structure
;  1559          1681  2         dbg$k_if_state          = 1,    ! that gets built.
;  1560          1682  2         dbg$k_lock_state        = 2,
;  1561          1683  2         dbg$k_log               = 3,
;  1562          1684  2         dbg$k_set_state         = 4,
;  1563          1685  2         dbg$k_terminate         = 5,
;  1564          1686  2         dbg$k_highest_qualifier = 5;
;  1565          1687  2
;  1566          1688  2     LOCAL
;  1567          1689  2         define_kind             : INITIAL(0),           ! Value of DEFINE/KEY qualifier
;  1568          1690  2         noun_node               : REF dbg$noun_node,    ! Pointer to a noun node
;  1569          1691  2         new_noun_node           : REF dbg$noun_node,    ! Another pointer to a noun node
;  1570          1692  2         adverb_node             : REF dbg$adverb_node,  ! Pointer to a noun node
;  1571          1693  2         new_adverb_node         : REF dbg$adverb_node,  ! Another pointer to a adverb node
;  1572          1694  2         state_name_node         : REF dbg$state_name_node,      ! Pointer to a state-name node
;  1573          1695  2         new_state_name_node     : REF dbg$state_name_node,      ! Another pointer to a state-name node
;  1574          1696  2         ptr                     : REF VECTOR[,BYTE],    ! Points into input string
;  1575          1697  2         status,
;  1576          1698  2         temp_key_desc           : REF dbg$stg_desc,     ! String desc. for DEFINE/KEY symbols
;  1577          1699  2         d_key_no,                                       ! Flag for NOxxx qualifier
;  1578          1700  2         define_key_value;                               ! Value for the qualifier
;  1579          1701  2
;  1580          1702  2
;  1581          1703  2     ! Check whether we are on a system that allows keypad input.
;  1582          1704  2     !
;  1583          1705  2     IF NOT .dbg$gb_keypad_input
;  1584          1706  2     THEN
;  1585          1707  2         SIGNAL(dbg$_nokeydef);
;  1586          1708  2
;  1587          1709  2     ! Fill in the fact that this is a DEFINE/KEY command in the verb node.
;  1588          1710  2     ! And clear the noun link value.
;  1589          1711  2     !
;  1590          1712  2     verb_node [dbg$b_verb_composite] = define_key;
```

```
;  1591          1713  2      verb_node [dbg$l_verb_object_ptr] = 0;
;  1592          1714  2
;  1593          1715  2      ! Build adverb list with defaults.
;  1594          1716  2      !
;  1595          1717  2
;  1596          1718  2      new_adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);  ! Get first node
;  1597          1719  2
;  1598          1720  2      verb_node [dbg$l_verb_adverb_ptr] = .new_adverb_node;
;  1599          1721  2      adverb_node = .new_adverb_node;
;  1600          1722  2
;  1601          1723  2      adverb_node [dbg$b_adverb_literal] = dbg$k_lowest_qualifier;        ! Initialize first node
;  1602          1724  2      adverb_node [dbg$l_adverb_value]   = 0;
;  1603          1725  2      adverb_node [dbg$l_adverb_link] = 0;
;  1604          1726  2
;  1605          1727  2      define_kind = dbg$k_lowest_qualifier + 1;
;  1606          1728  2      WHILE .define_kind [EQ dbg$k_highest_qualifier DO           ! Build rest of adverb list
;  1607          1729  3          BEGIN
;  1608          1730  3          new_adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);
;  1609          1731  3          adverb_node [dbg$l_adverb_link] = .new_adverb_node;
;  1610          1732  3          adverb_node = .new_adverb_node;
;  1611          1733  3
;  1612          1734  3          adverb_node [dbg$b_adverb_literal] = .define_kind;
;  1613          1735  3          IF .define_kind EQ[ dbg$k_if_state
;  1614          1736  3          THEN
;  1615          1737  4              BEGIN
;  1616          1738  4              ! Initialize the if-state node to point to a state name node that
;  1617          1739  4              ! points to a descriptor that has the current state-name.
;  1618          1740  4              !
;  1619          1741  4              temp_key_desc = dbg$get_tempmem(2);
;  1620          1742  4              temp_key_desc [dsc$w_length]  = 0;
;  1621          1743  4              temp_key_desc [dsc$b_dtype]   = dsc$k_dtype_t;
;  1622          1744  4              temp_key_desc [dsc$b_class]   = dsc$k_class_d;
;  1623          1745  4              temp_key_desc [dsc$a_pointer] = 0;
;  1624          1746  4
;  1625          1747  4              state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  1626          1748  4
;  1627          1749  4              ! Get the current state-name
;  1628          1750  4              !
;  1629          1751  4              smg$set_default_state(dbg$gl_key_table_id, 0, .temp_key_desc);
;  1630          1752  4              state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  1631          1753  4              state_name_node [dbg$l_state_name_link] = 0;
;  1632          1754  4
;  1633          1755  4              adverb_node [dbg$l_adverb_value] = .state_name_node;
;  1634          1756  4              END
;  1635          1757  3          ELSE
;  1636          1758  3              ! Let zero be the default for all the other adverb nodes
;  1637          1759  3              !
;  1638          1760  3              adverb_node [dbg$l_adverb_value] = 0;
;  1639          1761  3
;  1640          1762  3          define_kind = .define_kind + 1;
;  1641          1763  2          END;
;  1642          1764  2      adverb_node [dbg$l_adverb_link] = 0;
;  1643          1765  2
;  1644          1766  2      WHILE (NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)) AND
;  1645          1767  2            (.input_desc [dsc$w_length] GTR 0) DO
;  1646          1768  2
;  1647          1769  3          BEGIN
```

```
;  1648          1770  3          IF dbg$nmatch(.input_desc, dbg$cs_slash, 1)
;  1649          1771  3          THEN
;  1650          1772  4              BEGIN
;  1651          1773  4
;  1652          1774  4              ! Find out what kind of qualifier it is
;  1653          1775  4              !
;  1654          1776  4              ! Initialize value
;  1655          1777  4
;  1656          1778  4              define_key_value = 0;
;  1657          1779  4              d_key_NO = FALSE;
;  1658          1780  4
;  1659          1781  4              ! Check for a NO qualifier
;  1660          1782  4
;  1661          1783  4              WHILE CH$EQL(1, UPLIT(' '), 1, CH$PTR(.input_desc [dsc$a_pointer])) DO
;  1662          1784  5                  BEGIN
;  1663          1785  5                  input_desc [dsc$a_pointer] = CH$PLUS(.input_desc [dsc$a_pointer], 1);
;  1664          1786  5                  input_desc [dsc$w_length]  = .input_desc [dsc$w_length] - 1;
;  1665          1787  4                  END;
;  1666          1788  4
;  1667          1789  4              IF CH$EQL(2, CH$PTR(dbg$cs_NO), 2, CH$PTR(.input_desc [dsc$a_pointer]))
;  1668          1790  4              THEN
;  1669          1791  5                  BEGIN
;  1670          1792  5                  d_key_NO = TRUE;
;  1671          1793  5                  input_desc [dsc$a_pointer] = CH$PLUS(.input_desc [dsc$a_pointer], 2);
;  1672          1794  5                  input_desc [dsc$w_length]  = .input_desc [dsc$w_length] - 2;
;  1673          1795  4                  END;
;  1674          1796  4
;  1675          1797  4              ! Set Define_key with qualifier code, and get value of define_key_value.
;  1676          1798  4
;  1677          1799  4              SELECTONE TRUE  OF
;  1678          1800  4                  SET
;  1679          1801  4
;  1680          1802  4                  [dbg$nmatch(.input_desc, dbg$cs_echo, 1)] :
;  1681          1803  5                      BEGIN
;  1682          1804  5                      define_key_value = 0;
;  1683          1805  5                      define_kind = dbg$k_echo;
;  1684          1806  5                      IF .d_key_NO THEN define_key_value = 1;
;  1685          1807  4                      END;
;  1686          1808  4
;  1687          1809  4                  [dbg$nmatch(.input_desc, dbg$cs_if_state, 1)] :
;  1688          1810  5                      BEGIN
;  1689          1811  5                      define_key_value = 0;
;  1690          1812  5                      define_kind = dbg$k_if_state;
;  1691          1813  5                      IF NOT .d_key_NO
;  1692          1814  5                      THEN
;  1693          1815  6                          BEGIN
;  1694          1816  6                          temp_key_desc = dbg$get_tempmem(2);
;  1695          1817  6                          temp_key_desc[dsc$w_length]  = 0;
;  1696          1818  6                          temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1697          1819  6                          temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  1698          1820  6                          temp_key_desc[dsc$a_pointer] = 0;
;  1699          1821  6
;  1700          1822  6                          ! Look for =
;  1701          1823  6                          !
;  1702          1824  6
;  1703          1825  6                          IF NOT dbg$nmatch (.input_desc, dbg$cs_equal, 1)
;  1704          1826  6                          THEN
```

```
;  1705          1827  6                    report_error;
;  1706          1828  6
;  1707          1829  6                ! Look for a left paren
;  1708          1830  6                !
;  1709          1831  6
;  1710          1832  6                IF dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  1711          1833  6                THEN
;  1712          1834  7                    BEGIN
;  1713          1835  7
;  1714          1836  7                    ! Pick up the first state name
;  1715          1837  7                    !
;  1716          1838  7                    status = dbg$read_key_info (.input_desc,
;  1717          1839  7                                                .temp_key_desc,
;  1718          1840  7                                                .message_vect);
;  1719          1841  7                    IF NOT .status
;  1720          1842  7                    THEN
;  1721          1843  7                        RETURN sts$k_severe;
;  1722          1844  7
;  1723          1845  7                    new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  1724          1846  7                    state_name_node = .new_state_name_node;
;  1725          1847  7                    state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  1726          1848  7                    state_name_node [dbg$l_state_name_link] = 0;
;  1727          1849  7                    define_key_value = .state_name_node;
;  1728          1850  7
;  1729          1851  7                    WHILE dbg$nmatch (.input_desc, dbg$cs_comma, 1) DO
;  1730          1852  8                        BEGIN
;  1731          1853  8                        temp_key_desc = dbg$get_tempmem(2);
;  1732          1854  8                        temp_key_desc[dsc$w_length]  = 0;
;  1733          1855  8                        temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1734          1856  8                        temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  1735          1857  8                        temp_key_desc[dsc$a_pointer] = 0;
;  1736          1858  8
;  1737          1859  8                        ! Pick up the next state name
;  1738          1860  8                        !
;  1739          1861  8                        status = dbg$read_key_info (.input_desc,
;  1740          1862  8                                                    .temp_key_desc,
;  1741          1863  8                                                    .message_vect);
;  1742          1864  8                        IF NOT .status
;  1743          1865  8                        THEN
;  1744          1866  8                            RETURN sts$k_severe;
;  1745          1867  8
;  1746          1868  8                        new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  1747          1869  8                        state_name_node [dbg$l_state_name_link] = .new_state_name_node;
;  1748          1870  8                        state_name_node = .new_state_name_node;
;  1749          1871  8                        state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  1750          1872  8                        state_name_node [dbg$l_state_name_link] = 0;
;  1751          1873  8
;  1752          1874  7                        END;
;  1753          1875  7
;  1754          1876  7                ! Eat right paren
;  1755          1877  7                !
;  1756          1878  7
;  1757          1879  7                IF NOT dbg$nmatch (.input_desc, dbg$cs_right_paren, 1)
;  1758          1880  7                THEN
;  1759          1881  7                    report_error;
;  1760          1882  7
;  1761          1883  7                END
```

```
;  1762          1884  7                    ELSE
;  1763          1885  6                        BEGIN
;  1764          1886  7
;  1765          1887  7                        ! Pick up the only state name
;  1766          1888  7                        !
;  1767          1889  7
;  1768          1890  7                        status = dbg$read_key_info (.input_desc,
;  1769          1891  7                                                    .temp_key_desc,
;  1770          1892  7                                                    .message_vect);
;  1771          1893  7                        IF NOT .status
;  1772          1894  7                        THEN
;  1773          1895  7                            RETURN sts$k_severe;
;  1774          1896  7
;  1775          1897  7                        new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  1776          1898  7                        state_name_node = .new_state_name_node;
;  1777          1899  7                        state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  1778          1900  7                        state_name_node [dbg$l_state_name_link] = 0;
;  1779          1901  7                        define_key_value = .state_name_node;
;  1780          1902  7
;  1781          1903  7                        END;
;  1782          1904  6
;  1783          1905  6                    END;
;  1784          1906  5                END;
;  1785          1907  4
;  1786          1908  4
;  1787          1909  4            [dbg$nmatch(.input_desc, dbg$cs_lock_state, 3)] :
;  1788          1910  5                BEGIN
;  1789          1911  5                define_key_value = 1;
;  1790          1912  5                define_kind = dbg$k_lock_state;
;  1791          1913  5                IF .d_key_NO THEN define_key_value = 0;
;  1792          1914  4                END;
;  1793          1915  4
;  1794          1916  4            [dbg$nmatch(.input_desc, dbg$cs_log, 3)] :
;  1795          1917  5                BEGIN
;  1796          1918  5                define_key_value = 0;
;  1797          1919  5                define_kind = dbg$k_log;
;  1798          1920  5                IF .d_key_NO THEN define_key_value = 1;
;  1799          1921  4                END;
;  1800          1922  4
;  1801          1923  4            [dbg$nmatch(.input_desc, dbg$cs_set_state, 1)] :
;  1802          1924  5                BEGIN
;  1803          1925  5                define_kind = dbg$k_set_state;
;  1804          1926  5                define_key_value = 0;
;  1805          1927  5                IF NOT .d_key_NO
;  1806          1928  5                THEN
;  1807          1929  6                    BEGIN
;  1808          1930  6                    temp_key_desc = dbg$get_tempmem(2);
;  1809          1931  6                    temp_key_desc[dsc$w_length]  = 0;
;  1810          1932  6                    temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1811          1933  6                    temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  1812          1934  6                    temp_key_desc[dsc$a_pointer] = 0;
;  1813          1935  6
;  1814          1936  6                    ! Look for =
;  1815          1937  6                    !
;  1816          1938  6
;  1817          1939  6                    IF NOT dbg$nmatch (.input_desc, dbg$cs_equal, 1)
;  1818          1940  6                    THEN
```

```
;  1819          1941  6                    report_error;
;  1820          1942  6
;  1821          1943  6                ! Pick up the state name
;  1822          1944  6                !
;  1823          1945  6
;  1824          1946  6                status = dbg$read_key_info (.input_desc,
;  1825          1947  6                                            .temp_key_desc,
;  1826          1948  6                                            .message_vect);
;  1827          1949  6                IF NOT .status
;  1828          1950  6                THEN
;  1829          1951  6                    RETURN sts$k_severe;
;  1830          1952  6
;  1831          1953  6                define_key_value = .temp_key_desc;
;  1832          1954  5                END;
;  1833          1955  4            END;
;  1834          1956  4
;  1835          1957  4        [dbg$nmatch(.input_desc, dbg$cs_terminate, 1)] :
;  1836          1958  5            BEGIN
;  1837          1959  5            define_key_value = 1;
;  1838          1960  5            define_kind = dbg$k_terminate;
;  1839          1961  5            IF .d_key_NO THEN define_key_value = 0;
;  1840          1962  5
;  1841          1963  5            ! Since we may have to reset the default from 1 to 0
;  1842          1964  5            ! It is best to do this here and allow it to skip
;  1843          1965  5            ! over the similar code down below.
;  1844          1966  5            ! ( This is hacked up because the /NOECHO and /NOTERM are
;  1845          1967  5            !   mutually exlusive qualifiers. ) This bothers me too.
;  1846          1968  5            !
;  1847          1969  5            adverb_node = .verb_node [dbg$l_verb_adverb_ptr];
;  1848          1970  5            WHILE .adverb_node [dbg$b_adverb_literal] NEQ dbg$k_terminate DO
;  1849          1971  5                adverb_node = .adverb_node [dbg$l_adverb_link];
;  1850          1972  5            adverb_node [dbg$l_adverb_value] = .define_key_value;
;  1851          1973  5            define_key_value = 0;       ! This is to jump over the code below.
;  1852          1974  4            END;
;  1853          1975  4
;  1854          1976  4        [OTHERWISE] :
;  1855          1977  4            report_error;
;  1856          1978  4        TES;
;  1857          1979  4
;  1858          1980  4    ! Process the qualifier if it changes the default
;  1859          1981  4
;  1860          1982  4    IF .define_key_value NEQ 0
;  1861          1983  4    THEN
;  1862          1984  5        BEGIN
;  1863          1985  5        adverb_node = .verb_node [dbg$l_verb_adverb_ptr];
;  1864          1986  5        WHILE (.adverb_node [dbg$b_adverb_literal] NEQ .define_kind) DO
;  1865          1987  5            adverb_node = .adverb_node [dbg$l_adverb_link];
;  1866          1988  5        adverb_node [dbg$l_adverb_value] = .define_key_value;
;  1867          1989  5
;  1868          1990  5        ! If the qualifier is /NOECHO then the default of the terminate
;  1869          1991  5        ! qualifier changes to /TERMINATE
;  1870          1992  5        IF .adverb_node [dbg$b_adverb_literal] EQL dbg$k_echo
;  1871          1993  5        THEN
;  1872          1994  6            BEGIN
;  1873          1995  6            WHILE .adverb_node [dbg$b_adverb_literal] NEQ dbg$k_terminate DO
;  1874          1996  6                adverb_node = .adverb_node [dbg$l_adverb_link];
;  1875          1997  6            adverb_node [dbg$l_adverb_value] = 1;
```

```
;  1876          1998  5                END;
;  1877          1999  4            END;
;  1878          2000  4
;  1879          2001  4        END ! End of picking up qualifier after slash
;  1880          2002  4
;  1881          2003  3    ELSE
;  1882          2004  4        BEGIN
;  1883          2005  4
;  1884          2006  4        ! Process key name or equivalence string
;  1885          2007  4
;  1886          2008  4        IF .verb_node [dbg$l_verb_object_ptr] EQL 0
;  1887          2009  4        THEN
;  1888          2010  5            BEGIN
;  1889          2011  5
;  1890          2012  5            ! Get key name
;  1891          2013  5            !
;  1892          2014  5
;  1893          2015  5            temp_key_desc = dbg$get_tempmem(2);
;  1894          2016  5            temp_key_desc[dsc$w_length]  = 0;
;  1895          2017  5            temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1896          2018  5            temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  1897          2019  5            temp_key_desc[dsc$a_pointer] = 0;
;  1898          2020  5            status = dbg$read_key_info (.input_desc,
;  1899          2021  5                                        .temp_key_desc,
;  1900          2022  5                                        .message_vect);
;  1901          2023  5            IF NOT .status
;  1902          2024  5            THEN
;  1903          2025  5                RETURN sts$k_severe;
;  1904          2026  5
;  1905          2027  5            ! Make noun node for key-name and equivalence string
;  1906          2028  5            !
;  1907          2029  5
;  1908          2030  5            new_noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  1909          2031  5
;  1910          2032  5            verb_node [dbg$l_verb_object_ptr] = .new_noun_node;
;  1911          2033  5            noun_node = .new_noun_node;
;  1912          2034  5            noun_node [dbg$l_noun_value] = .temp_key_desc;
;  1913          2035  5            noun_node [dbg$l_adjective_ptr] = 0;
;  1914          2036  5            noun_node [dbg$l_noun_link] = 0;
;  1915          2037  5            END
;  1916          2038  5
;  1917          2039  5    ! If we got the key-name ok, go after the equivalence string
;  1918          2040  5    ! Provided there is still something left to get.
;  1919          2041  5    !
;  1920          2042  5
;  1921          2043  4    ELSE
;  1922          2044  4        IF .noun_node [dbg$l_adjective_ptr] EQL 0
;  1923          2045  4        THEN
;  1924          2046  5            BEGIN
;  1925          2047  5            temp_key_desc = dbg$get_tempmem(2);
;  1926          2048  5            temp_key_desc[dsc$w_length]  = 0;
;  1927          2049  5            temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1928          2050  5            temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  1929          2051  5            temp_key_desc[dsc$a_pointer] = 0;
;  1930          2052  5
;  1931          2053  5            ! Pick up leading quote
;  1932          2054  5            !
```

```
;  1933          2055  5
;  1934          2056  5                    IF NOT dbg$nmatch (.input_desc, dbg$cs_dblquote, 1)
;  1935          2057  5                    THEN
;  1936          2058  6                        BEGIN
;  1937          2059  6                        status = dbg$read_key_info (.input_desc,
;  1938          2060  6                                                    .temp_key_desc,
;  1939          2061  6                                                    .message_vect);
;  1940          2062  6                        IF NOT .status
;  1941          2063  6                        THEN
;  1942          2064  6                            report_error;
;  1943          2065  6                        END
;  1944          2066  5                    ELSE
;  1945          2067  6                        BEGIN
;  1946          2068  6                        ! Pick up the quoted string
;  1947          2069  6                        !
;  1948          2070  6
;  1949          2071  6                        status = dbg$naccept_string (.input_desc,
;  1950          2072  6                                                     temp_key_desc [dsc$a_pointer],
;  1951          2073  6                                                     dbg$k_dblquote,
;  1952          2074  6                                                     FALSE,
;  1953          2075  6                                                     .message_vect,
;  1954          2076  6                                                     FALSE);
;  1955          2077  6
;  1956          2078  6                        IF NOT .status
;  1957          2079  6                        THEN
;  1958          2080  6                            RETURN sts$k_severe;
;  1959          2081  6
;  1960          2082  6                        ! Grab the length of the string and move the pointer.
;  1961          2083  6                        !
;  1962          2084  6                        temp_key_desc [dsc$w_length] = CH$RCHAR_A(temp_key_desc [dsc$a_pointer]);
;  1963          2085  6
;  1964          2086  5                        END;
;  1965          2087  5
;  1966          2088  5                    noun_node [dbg$l_adjective_ptr] = .temp_key_desc;
;  1967          2089  5
;  1968          2090  5                    END
;  1969          2091  4                ELSE
;  1970          2092  4                    report_error;
;  1971          2093  3            END;
;  1972          2094  3
;  1973          2095  2        END;    ! End While
;  1974          2096  2
;  1975          2097  2        ! Check to see if key-name and the equivalence string have been entered.
;  1976          2098  2        ! If not, return a need more message and error status.
;  1977          2099  2        !
;  1978          2100  2
;  1979          2101  2        IF .verb_node [dbg$l_verb_object_ptr] EQL 0
;  1980          2102  2        THEN
;  1981          2103  3            BEGIN
;  1982          2104  3            .message_vect = dbg$nmake_arg_vect(dbg$_needmore);
;  1983          2105  3            RETURN sts$k_severe;
;  1984          2106  3            END;
;  1985          2107  2        IF .noun_node [dbg$l_adjective_ptr] EQL 0
;  1986          2108  2        THEN
;  1987          2109  3            BEGIN
;  1988          2110  3            .message_vect = dbg$nmake_arg_vect(dbg$_needmore);
;  1989          2111  3            RETURN sts$k_severe;
```

```
;  1990          2112  2          END;
;  1991          2113  2
;  1992          2114  2      RETURN sts$k_success;
;  1993          2115  1      END;
;  INFO#250                  L1:2044
;  Referenced LOCAL symbol NOUN_NODE is probably not initialized


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

                                                              04    00095 P.AAW:    .BYTE   4                                       ;
                                                        4F    48    43  45    00096           .ASCII  \ECHO\                          ;
                                                              08    0009A P.AAX:    .BYTE   8                                       ;
                                                  45    54    41  54  53  5F    46  49    0009B           .ASCII  \IF_STATE\              ;
                                                              0A    000A3 P.AAY:    .BYTE   10                                      ;
                                              45  54    41    54  53  5F    4B  43  4F    4C    000A4           .ASCII  \LOCK_STATE\    ;
                                                              03    000AE P.AAZ:    .BYTE   3                                       ;
                                                        47    4F    4C  000AF           .ASCII  \LOG\                           ;
                                                              09    000B2 P.ABA:    .BYTE   9                                       ;
                                                  45    54    41  54  53  5F    54  45  53    000B3           .ASCII  \SET_STATE\     ;
                                                              09    000BC P.ABB:    .BYTE   9                                       ;
                                                  45    54    41  4E  49  4D    52  45  54    000BD           .ASCII  \TERMINATE\     ;
                                                              4F    4E    000C6 P.ABC:    .ASCII  \NO\                            ;
                                                              28    01    000C8 P.ABD:    .BYTE   1, 40                           ;
                                                              29    01    000CA P.ABE:    .BYTE   1, 41                           ;
                                                              2C    01    000CC P.ABF:    .BYTE   1, 44                           ;
                                                              0D    01    000CE P.ABG:    .BYTE   1, 13                           ;
                                                              22    01    000D0 P.ABH:    .BYTE   1, 34                           ;
                                                              3D    01    000D2 P.ABI:    .BYTE   1, 61                           ;
                                                              2F    01    000D4 P.ABJ:    .BYTE   1, 47                           ;
                                                                    000D6           .BLKB   2                                       ;
                                                  00  00    00    20  000D8 P.ABK:    .ASCII  \ \<0><0><0>                    ;

                                                                    DBG$CS_ECHO=            P.AAW
                                                                    DBG$CS_IF_STATE=        P.AAX
                                                                    DBG$CS_LOCK_STATE=      P.AAY
                                                                    DBG$CS_LOG=             P.AAZ
                                                                    DBG$CS_SET_STATE=       P.ABA
                                                                    DBG$CS_TERMINATE=       P.ABB
                                                                    DBG$CS_NO=              P.ABC
                                                                    DBG$CS_LEFT_PAREN=      P.ABD
                                                                    DBG$CS_RIGHT_PAREN=     P.ABE
                                                                    DBG$CS_COMMA=           P.ABF
                                                                    DBG$CS_CR=              P.ABG
                                                                    DBG$CS_DBLQUOTE=        P.ABH
                                                                    DBG$CS_EQUAL=           P.ABI
                                                                    DBG$CS_SLASH=           P.ABJ


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

                                                        0FFC 00000 DBG$NPARSE_DEF_KEY:
                                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11    ; 1599
                                                  5E          08  C2 00002           SUBL2   #8, SP                  ;
                                                              7E  D4 00005           CLRL    DEFINE_KIND             ; 1656
                                                  0D 00000000G  00 E8 00007           BLBS    DBG$GB_KEYPAD_INPUT, 1$ ; 1705
```

```
                00028D18   8F   DD 0000E          PUSHL   #167192                                  ; 1707
00000000G   00             01   FB 00014          CALLS   #1, LIB$SIGNAL
                57      08 AC   D0 0001B 1$:      MOVL    VERB_NODE, R7                          ; 1712
             01 A7          07   90 0001F          MOVB    #7, T(R7)
                      08 A7   D4 00023          CLRL    8(R7)                                  ; 1713
                         03   DD 00026          PUSHL   #3                                     ; 1718
00000000G   00             01   FB 00028          CALLS   #1, DBG$GET_TEMPMEM
                53           50   D0 0002F          MOVL    R0, NEW_ADVERB_NODE
             04 A7           53   D0 00032          MOVL    NEW_ADVERB_NODE, 4(R7)                 ; 1720
                54           53   D0 00036          MOVL    NEW_ADVERB_NODE, ADVERB_NODE           ; 1721
                             64   94 00039          CLRB    (ADVERB_NODE)                          ; 1723
                      04 A4   7C 0003B          CLRQ    4(ADVERB_NODE)                         ; 1724
                6E           01   D0 0003E          MOVL    #1, DEFINE_KIND                        ; 1727
                05           6E   D1 00041 2$:      CMPL    DEFINE_KIND, #5                        ; 1728
                             61   14 00044          BGTR    5$
                             03   DD 00046          PUSHL   #3                                     ; 1730
00000000G   00             01   FB 00048          CALLS   #1, DBG$GET_TEMPMEM
                53           50   D0 0004F          MOVL    R0, NEW_ADVERB_NODE
             08 A4           53   D0 00052          MOVL    NEW_ADVERB_NODE, 8(ADVERB_NODE)        ; 1731
                54           53   D0 00056          MOVL    NEW_ADVERB_NODE, ADVERB_NODE           ; 1732
                64           6E   90 00059          MOVB    DEFINE_KIND, (ADVERB_NODE)             ; 1734
                01           6E   D1 0005C          CMPL    DEFINE_KIND, #1                        ; 1735
                             3F   12 0005F          BNEQ    3$
                             02   DD 00061          PUSHL   #2                                     ; 1741
00000000G   00             01   FB 00063          CALLS   #1, DBG$GET_TEMPMEM
                52           50   D0 0006A          MOVL    R0, TEMP_KEY_DESC
                62 020E0000  8F   D0 0006D          MOVL    #34471936, (TEMP_KEY_DESC)             ; 1742
                      04 A2   D4 00074          CLRL    4(TEMP_KEY_DESC)                       ; 1745
                             02   DD 00077          PUSHL   #2                                     ; 1747
00000000G   00             01   FB 00079          CALLS   #1, DBG$GET_TEMPMEM
                59           50   D0 00080          MOVL    R0, STATE_NAME_NODE
                             52   DD 00083          PUSHL   TEMP_KEY_DESC                          ; 1751
                             7E   D4 00085          CLRL    -(SP)
                00000000G    00   9F 00087          PUSHAB  DBG$GL_KEY_TABLE_ID
00000000G   00             03   FB 0008D          CALLS   #3, SMG$SET_DEFAULT_STATE
                69           52   D0 00094          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)       ; 1752
                      04 A9   D4 00097          CLRL    4(STATE_NAME_NODE)                     ; 1753
             04 A4           59   D0 0009A          MOVL    STATE_NAME_NODE, 4(ADVERB_NODE)        ; 1755
                             03   11 0009E          BRB     4$                                     ; 1735
                      04 A4   D4 000A0 3$:      CLRL    4(ADVERB_NODE)                         ; 1760
                             6E   D6 000A3 4$:      INCL    DEFINE_KIND                            ; 1762
                             9A   11 000A5          BRB     2$                                     ; 1728
                      08 A4   D4 000A7 5$:      CLRL    8(ADVERB_NODE)                         ; 1764
                53      04 AC   D0 000AA          MOVL    INPUT_DESC, R3                         ; 1766
                             01   DD 000AE 6$:      PUSHL   #1
                00000000'    EF   9F 000B0          PUSHAB  DBG$CS_CR
                             53   DD 000B6          PUSHL   R3
00000000G   00             03   FB 000B8          CALLS   #3, DBG$NMATCH
                03           50   E9 000BF          BLBC    R0, 8$
                          03C9   31 000C2 7$:      BRW     54$
                             63   B5 000C5 8$:      TSTW    (R3)                                   ; 1767
                             F9   13 000C7          BEQL    7$
                             01   DD 000C9          PUSHL   #1                                     ; 1770
                00000000'    EF   9F 000CB          PUSHAB  DBG$CS_SLASH
                             53   DD 000D1          PUSHL   R3
00000000G   00             03   FB 000D3          CALLS   #3, DBG$NMATCH
                03           50   E8 000DA          BLBS    R0, 9$
```

```
                        02BB  31 000DD         BRW     45$                                             ;
                          56  D4 000E0 9$:     CLRL    DEFINE_KEY_VALUE                                ; 1778
                          5A  D4 000E2         CLRL    D_KEY_NO                                        ; 1779
                50        04  A3 9E 000E4     MOVAB   4(R3), R0                                       ; 1783
             00 B0 00000000'  EF 91 000E8 10$:  CMPB    P.ABK, @0(R0)                                   ;
                          06  12 000F0         BNEQ    11$                                             ;
                          60  D6 000F2         INCL    (R0)                                            ; 1785
                          63  B7 000F4         DECW    (R3)                                            ; 1786
                          F0  11 000F6         BRB     10$                                             ; 1783
             00 B0 00000000'  EF B1 000F8 11$:  CMPW    DBG$CS_NO, @0(R0)                               ; 1789
                          09  12 00100         BNEQ    12$                                             ;
                5A            01 D0 00102     MOVL    #1, D_KEY_NO                                    ; 1792
                60            02 C0 00105     ADDL2   #2, (R0)                                        ; 1793
                63            02 A2 00108     SUBW2   #2, (R3)                                        ; 1794
                              01 DD 0010B 12$:  PUSHL   #1                                              ; 1802
                   00000000'  EF 9F 0010D     PUSHAB  DBG$CS_ECHO                                     ;
                              53 DD 00113     PUSHL   R3                                              ;
00000000G       00            03 FB 00115     CALLS   #3, DBG$NMATCH                                  ;
                01            50 D1 0011C     CMPL    R0, #1                                          ;
                              07 12 0011F     BNEQ    13$                                             ;
                              56 D4 00121     CLRL    DEFINE_KEY_VALUE                                ; 1804
                              6E D4 00123     CLRL    DEFINE_KIND                                     ; 1805
                            01A2 31 00125     BRW     28$                                             ; 1806
                              01 DD 00128 13$:  PUSHL   #1                                              ; 1809
                   00000000'  EF 9F 0012A     PUSHAB  DBG$CS_IF_STATE                                 ;
                              53 DD 00130     PUSHL   R3                                              ;
00000000G       00            03 FB 00132     CALLS   #3, DBG$NMATCH                                  ;
                01            50 D1 00139     CMPL    R0, #1                                          ;
                              03 13 0013C     BEQL    14$                                             ;
                            014C 31 0013E     BRW     26$                                             ;
                              56 D4 00141 14$:  CLRL    DEFINE_KEY_VALUE                                ; 1811
                6E            01 D0 00143     MOVL    #1, DEFINE_KIND                                 ; 1812
                03            5A E9 00146     BLBC    D_KEY_NO, T5$                                   ; 1813
                            0221 31 00149     BRW     40$                                             ;
                              02 DD 0014C 15$:  PUSHL   #2                                              ; 1816
00000000G       00            01 FB 0014E     CALLS   #1, DBG$GET_TEMPMEM                             ;
                52            50 D0 00155     MOVL    R0, TEMP_KEY_DESC                               ;
                62 020E0000   8F D0 00158     MOVL    #34471936, (TEMP_KEY_DESC)                      ; 1817
                        04    A2 D4 0015F     CLRL    4(TEMP_KEY_DESC)                                ; 1820
                              01 DD 00162     PUSHL   #1                                              ; 1825
                   00000000'  EF 9F 00164     PUSHAB  DBG$CS_EQUAL                                    ;
                              53 DD 0016A     PUSHL   R3                                              ;
00000000G       00            03 FB 0016C     CALLS   #3, DBG$NMATCH                                  ;
                2C            50 E8 00173     BLBS    R0, 18$                                         ;
                              01 DD 00176 16$:  PUSHL   #1                                              ; 1826
                   00000000'  EF 9F 00178     PUSHAB  DBG$CS_CR                                       ;
                              53 DD 0017E     PUSHL   R3                                              ;
00000000G       00            03 FB 00180     CALLS   #3, DBG$NMATCH                                  ;
                03            50 E9 00187     BLBC    R0, 17$                                         ;
                            030B 31 0018A     BRW     55$                                             ;
                              53 DD 0018D 17$:  PUSHL   R3                                              ;
00000000G       00            01 FB 0018F     CALLS   #1, DBG$NNEXT_WORD                              ;
                              50 DD 00196     PUSHL   R0                                              ;
00000000G       00            01 FB 00198     CALLS   #1, DBG$NSYNTAX_ERROR                           ;
                            0303 31 0019F     BRW     56$                                             ;
                              01 DD 001A2 18$:  PUSHL   #1                                              ; 1832
                   00000000'  EF 9F 001A4     PUSHAB  DBG$CS_LEFT_PAREN                               ;
```

```
                        53   DD 001AA          PUSHL   R3                                                          ;
00000000G   00          03   FB 001AC          CALLS   #3, DBG$NMATCH                                              ;
            03          50   E8 001B3          BLBS    R0, 19$                                                     ;
                  00A3  31   001B6          BRW     23$                                                         ;
                    0C  AC   DD 001B9 19$:    PUSHL   MESSAGE_VECT                                                ; 1840
                        52   DD 001BC          PUSHL   TEMP_KEY_DESC                                               ; 1839
                        53   DD 001BE          PUSHL   R3                                                          ; 1838
          0000V     CF        03   FB 001C0          CALLS   #3, DBG$READ_KEY_INFO                                       ;
                    5B        50   D0 001C5          MOVL    R0, STATUS                                                  ;
                    57        5B   E9 001C8          BLBC    STATUS, 21$                                                 ; 1841
                              02   DD 001CB          PUSHL   #2                                                          ; 1845
00000000G   00                01   FB 001CD          CALLS   #1, DBG$GET_TEMPMEM                                         ;
         04   AE              50   D0 001D4          MOVL    R0, NEW_STATE_NAME_NODE                                     ;
                    59        04   AE   D0 001D8          MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE                        ; 1846
                    69        52   D0 001DC          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)                            ; 1847
                              04   A9   D4 001DF          CLRL    4(STATE_NAME_NODE)                                          ; 1848
                    56        59   D0 001E2          MOVL    STATE_NAME_NODE, DEFINE_KEY_VALUE                           ; 1849
                    58        04   A9   9E 001E5          MOVAB   4(R9), R8                                           ; 1869
                              01   DD 001E9 20$:    PUSHL   #1                                                          ; 1851
                00000000'     EF   9F 001EB          PUSHAB  DBG$CS_COMMA                                                ;
                              53   DD 001F1          PUSHL   R3                                                          ;
00000000G   00                03   FB 001F3          CALLS   #3, DBG$NMATCH                                              ;
            48                50   E9 001FA          BLBC    R0, 22$                                                     ;
                              02   DD 001FD          PUSHL   #2                                                          ; 1853
00000000G   00                01   FB 001FF          CALLS   #1, DBG$GET_TEMPMEM                                         ;
            52                50   D0 00206          MOVL    R0, TEMP_KEY_DESC                                           ;
            62  020E0000     8F   D0 00209          MOVL    #34471936, (TEMP_KEY_DESC)                                  ; 1854
                        04    A2   D4 00210          CLRL    4(TEMP_KEY_DESC)                                            ; 1857
                        0C    AC   DD 00213          PUSHL   MESSAGE_VECT                                                ; 1863
                              52   DD 00216          PUSHL   TEMP_KEY_DESC                                               ; 1862
                              53   DD 00218          PUSHL   R3                                                          ; 1861
          0000V     CF        03   FB 0021A          CALLS   #3, DBG$READ_KEY_INFO                                       ;
                    5B        50   D0 0021F          MOVL    R0, STATUS                                                  ;
                    46        5B   E9 00222 21$:    BLBC    STATUS, 24$                                                 ; 1864
                              02   DD 00225          PUSHL   #2                                                          ; 1868
00000000G   00                01   FB 00227          CALLS   #1, DBG$GET_TEMPMEM                                         ;
         04   AE              50   D0 0022E          MOVL    R0, NEW_STATE_NAME_NODE                                     ;
                    68        04   AE   D0 00232          MOVL    NEW_STATE_NAME_NODE, (R8)                                   ; 1869
                    59        04   AE   D0 00236          MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE                        ; 1870
                    69        52   D0 0023A          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)                            ; 1871
                    58        04   A9   9E 0023D          MOVAB   4(R9), R8                                           ; 1872
                              68   D4 00241          CLRL    (R8)                                                        ;
                              A4   11 00243          BRB     20$                                                         ; 1851
                              01   DD 00245 22$:    PUSHL   #1                                                          ; 1879
                00000000'     EF   9F 00247          PUSHAB  DBG$CS_RIGHT_PAREN                                          ;
                              53   DD 0024D          PUSHL   R3                                                          ;
00000000G   00                03   FB 0024F          CALLS   #3, DBG$NMATCH                                              ;
            77                50   E8 00256          BLBS    R0, 29$                                                     ;
                        FF1A  31   00259          BRW     16$                                                         ; 1880
                        0C    AC   DD 0025C 23$:    PUSHL   MESSAGE_VECT                                                ; 1893
                              52   DD 0025F          PUSHL   TEMP_KEY_DESC                                               ; 1892
                              53   DD 00261          PUSHL   R3                                                          ; 1891
          0000V     CF        03   FB 00263          CALLS   #3, DBG$READ_KEY_INFO                                       ;
                    5B        50   D0 00268          MOVL    R0, STATUS                                                  ;
                    03        5B   E8 0026B 24$:    BLBS    STATUS, 25$                                                 ; 1894
                        0238  31   0026E          BRW     57$                                                         ;
                              02   DD 00271 25$:    PUSHL   #2                                                          ; 1898
```

```
        00000000G  00           01  FB 00273          CALLS   #1, DBG$GET_TEMPMEM                      ;
               04  AE           50  D0 0027A          MOVL    R0, NEW_STATE_NAME_NODE                  ;
                   59       04  AE  D0 0027E          MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE     ; 1899
                   69           52  D0 00282          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)         ; 1900
                            04  A9  D4 00285          CLRL    4(STATE_NAME_NODE)                       ; 1901
                   56           59  D0 00288          MOVL    STATE_NAME_NODE, DEFINE_KEY_VALUE        ; 1902
                                43  11 0028B          BRB     29$                                      ; 1813
                                03  DD 0028D  26$:    PUSHL   #3                                       ; 1909
                       00000000'  EF  9F 0028F          PUSHAB  DBG$CS_LOCK_STATE                        ;
                                53  DD 00295          PUSHL   R3                                       ;
        00000000G  00           03  FB 00297          CALLS   #3, DBG$NMATCH                           ;
                   01           50  D1 0029E          CMPL    R0, #1                                   ;
                                0C  12 002A1          BNEQ    27$                                      ;
                   56           01  D0 002A3          MOVL    #1, DEFINE_KEY_VALUE                     ; 1911
                   6E           02  D0 002A6          MOVL    #2, DEFINE_KIND                          ; 1912
                   24           5A  E9 002A9          BLBC    D_KEY_NO, 29$                            ; 1913
                              00BC  31 002AC          BRW     39$                                      ;
                                03  DD 002AF  27$:    PUSHL   #3                                       ; 1916
                       00000000'  EF  9F 002B1          PUSHAB  DBG$CS_LOG                               ;
                                53  DD 002B7          PUSHL   R3                                       ;
        00000000G  00           03  FB 002B9          CALLS   #3, DBG$NMATCH                           ;
                   01           50  D1 002C0          CMPL    R0, #1                                   ;
                                0D  12 002C3          BNEQ    30$                                      ;
                                56  D4 002C5          CLRL    DEFINE_KEY_VALUE                         ; 1918
                   6E           03  D0 002C7          MOVL    #3, DEFINE_KIND                          ; 1919
                   68           5A  E9 002CA  28$:    BLBC    D_KEY_NO, 34$                            ; 1920
                   56           01  D0 002CD          MOVL    #1, DEFINE_KEY_VALUE                     ;
                                63  11 002D0  29$:    BRB     34$                                      ; 1799
                                01  DD 002D2  30$:    PUSHL   #1                                       ; 1923
                       00000000'  EF  9F 002D4          PUSHAB  DBG$CS_SET_STATE                         ;
                                53  DD 002DA          PUSHL   R3                                       ;
        00000000G  00           03  FB 002DC          CALLS   #3, DBG$NMATCH                           ;
                   01           50  D1 002E3          CMPL    R0, #1                                   ;
                                4F  12 002E6          BNEQ    35$                                      ;
                   6E           04  D0 002E8          MOVL    #4, DEFINE_KIND                          ; 1925
                                56  D4 002EB          CLRL    DEFINE_KEY_VALUE                         ; 1926
                   7D           5A  E8 002ED          BLBS    D_KEY_NO, 40$                            ; 1927
                                02  DD 002F0          PUSHL   #2                                       ; 1930
        00000000G  00           01  FB 002F2          CALLS   #1, DBG$GET_TEMPMEM                      ;
                   52           50  D0 002F9          MOVL    R0, TEMP_KEY_DESC                        ;
                   62 020E0000  8F  D0 002FC          MOVL    #34471936, (TEMP_KEY_DESC)               ; 1931
                            04  A2  D4 00303          CLRL    4(TEMP_KEY_DESC)                         ; 1934
                                01  DD 00306          PUSHL   #1                                       ; 1939
                       00000000'  EF  9F 00308          PUSHAB  DBG$CS_EQUAL                             ;
                                53  DD 0030E          PUSHL   R3                                       ;
        00000000G  00           03  FB 00310          CALLS   #3, DBG$NMATCH                           ;
                   03           50  E8 00317          BLBS    R0, 32$                                  ;
                              FE59  31 0031A  31$:    BRW     16$                                      ;
                            0C  AC  DD 0031D  32$:    PUSHL   MESSAGE_VECT                             ; 1948
                                52  DD 00320          PUSHL   TEMP_KEY_DESC                            ; 1947
                                53  DD 00322          PUSHL   R3                                       ; 1946
            0000V  CF           03  FB 00324          CALLS   #3, DBG$READ_KEY_INFO                    ;
                   5B           50  D0 00329          MOVL    R0, STATUS                               ;
                   03           5B  E8 0032C          BLBS    STATUS, 33$                              ; 1949
                              0177  31 0032F          BRW     57$                                      ;
                   56           52  D0 00332  33$:    MOVL    TEMP_KEY_DESC, DEFINE_KEY_VALUE          ; 1953
                                36  11 00335  34$:    BRB     40$                                      ; 1799
```

```
                                                01 DD 00337 35$:    PUSHL   #1                                      ; 1957
                                00000000'       EF 9F 00339         PUSHAB  DBG$CS_TERMINATE                        ;
                                                53 DD 0033F         PUSHL   R3                                      ;
                        00000000G   00          03 FB 00341         CALLS   #3, DBG$NMATCH                          ;
                                    01          50 D1 00348         CMPL    R0, #1                                  ;
                                                CD 12 0034B         BNEQ    31$                                     ;
                                    56          01 D0 0034D         MOVL    #1, DEFINE_KEY_VALUE                    ; 1959
                                    6E          05 D0 00350         MOVL    #5, DEFINE_KIND                         ; 1960
                                    02          5A E9 00353         BLBC    D_KEY_NO, 36$                           ; 1961
                                                56 D4 00356         CLRL    DEFINE_KEY_VALUE                        ;
                                    54      04  A7 D0 00358 36$:    MOVL    4(R7), ADVERB_NODE                      ; 1969
                                    05          64 91 0035C 37$:    CMPB    (ADVERB_NODE), #5                       ; 1970
                                                06 13 0035F         BEQL    38$                                     ;
                                    54      08  A4 D0 00361         MOVL    8(ADVERB_NODE), ADVERB_NODE             ; 1971
                                                F5 11 00365         BRB     37$                                     ;
                                04  A4              56 D0 00367 38$:    MOVL    DEFINE_KEY_VALUE, 4(ADVERB_NODE)        ; 1972
                                                56 D4 0036B 39$:    CLRL    DEFINE_KEY_VALUE                        ; 1973
                                                56 D5 0036D 40$:    TSTL    DEFINE_KEY_VALUE                        ; 1982
                                                79 13 0036F         BEQL    47$                                     ;
                                    54      04  A7 D0 00371         MOVL    4(R7), ADVERB_NODE                      ; 1985
                6E          64      08          00 ED 00375 41$:    CMPZV   #0, #8, (ADVERB_NODE), DEFINE_KIND      ; 1986
                                                06 13 0037A         BEQL    42$                                     ;
                                    54      08  A4 D0 0037C         MOVL    8(ADVERB_NODE), ADVERB_NODE             ; 1987
                                                F3 11 00380         BRB     41$                                     ;
                                04  A4              56 D0 00382 42$:    MOVL    DEFINE_KEY_VALUE, 4(ADVERB_NODE)        ; 1988
                                                64 95 00386         TSTB    (ADVERB_NODE)                           ; 1992
                                                60 12 00388         BNEQ    47$                                     ;
                                    05          64 91 0038A 43$:    CMPB    (ADVERB_NODE), #5                       ; 1995
                                                06 13 0038D         BEQL    44$                                     ;
                                    54      08  A4 D0 0038F         MOVL    8(ADVERB_NODE), ADVERB_NODE             ; 1996
                                                F5 11 00393         BRB     43$                                     ;
                                04  A4              01 D0 00395 44$:    MOVL    #1, 4(ADVERB_NODE)                      ; 1997
                                                4F 11 00399         BRB     47$                                     ; 1770
                                    58      0C  AC D0 0039B 45$:    MOVL    MESSAGE_VECT, R8                        ; 2022
                                            08  A7 D5 0039F         TSTL    8(R7)                                   ; 2008
                                                49 12 003A2         BNEQ    48$                                     ;
                                                02 DD 003A4         PUSHL   #2                                      ; 2015
                        00000000G   00          01 FB 003A6         CALLS   #1, DBG$GET_TEMPMEM                     ;
                                    52          50 D0 003AD         MOVL    R0, TEMP_KEY_DESC                       ;
                                    62  020E0000  8F D0 003B0         MOVL    #34471936, (TEMP_KEY_DESC)              ; 2016
                                            04  A2 D4 003B7         CLRL    4(TEMP_KEY_DESC)                        ; 2019
                                        0104    8F BB 003BA         PUSHR   #^M<R2,R8>                              ; 2021
                                                53 DD 003BE         PUSHL   R3                                      ; 2020
                            0000V   CF          03 FB 003C0         CALLS   #3, DBG$READ_KEY_INFO                   ;
                                    5B          50 D0 003C5         MOVL    R0, STATUS                              ;
                                    03          5B E8 003C8         BLBS    STATUS, 46$                             ; 2023
                                                0ODB 31 003CB       BRW     57$                                     ;
                                                04 DD 003CE 46$:    PUSHL   #4                                      ; 2030
                        00000000G   00          01 FB 003D0         CALLS   #1, DBG$GET_TEMPMEM                     ;
                                08  AE          50 D0 003D7         MOVL    R0, NEW_NOUN_NODE                       ;
                                08  A7      08  AE D0 003DB         MOVL    NEW_NOUN_NODE, 8(R7)                    ; 2032
                                    55      08  AE D0 003E0         MOVL    NEW_NOUN_NODE, NOUN_NODE                ; 2033
                                    65          52 D0 003E4         MOVL    TEMP_KEY_DESC, (NOUN_NODE)              ; 2034
                                            04  A5 7C 003E7         CLRQ    4(NOUN_NODE)                            ; 2035
                                                FCC1 31 003EA 47$:  BRW     6$                                      ; 2008
                                            04  A5 D5 003ED 48$:    TSTL    4(NOUN_NODE)                            ; 2044
                                                3B 12 003F0         BNEQ    49$                                     ;
```

```
                                02  DD 003F2          PUSHL   #2                                      ; 2047
00000000G   00                  01  FB 003F4          CALLS   #1, DBG$GET_TEMPMEM
                52              50  D0 003FB          MOVL    R0, TEMP_KEY_DESC
                62 020E0000     8F  D0 003FE          MOVL    #34471936, (TEMP_KEY_DESC)              ; 2048
                          04    A2  D4 00405          CLRL    4(TEMP_KEY_DESC)                        ; 2051
                                01  DD 00408          PUSHL   #1                                      ; 2056
                   00000000'    EF  9F 0040A          PUSHAB  DBG$CS_DBLQUOTE
                                53  DD 00410          PUSHL   R3
00000000G   00                  03  FB 00412          CALLS   #3, DBG$NMATCH
                4B              50  E8 00419          BLBS    R0, 52$
                        0104    8F  BB 0041C          PUSHR   #^M<R2,R8>                              ; 2060
                                53  DD 00420          PUSHL   R3                                      ; 2059
      0000V     CF              03  FB 00422          CALLS   #3, DBG$READ_KEY_INFO
                5B              50  D0 00427          MOVL    R0, STATUS
                5A              5B  E8 0042A          BLBS    STATUS, 53$                             ; 2062
                                01  DD 0042D 49$:     PUSHL   #1                                      ; 2063
                   00000000'    EF  9F 0042F          PUSHAB  DBG$CS_CR
                                53  DD 00435          PUSHL   R3
00000000G   00                  03  FB 00437          CALLS   #3, DBG$NMATCH
                0F              50  E9 0043E          BLBC    R0, 50$
                   000280D0     8F  DD 00441          PUSHL   #164048
00000000G   00                  01  FB 00447          CALLS   #1, DBG$NMAKE_ARG_VECT
                                12  11 0044E          BRB     51$
                                53  DD 00450 50$:     PUSHL   R3
00000000G   00                  01  FB 00452          CALLS   #1, DBG$NNEXT_WORD
                                50  DD 00459          PUSHL   R0
00000000G   00                  01  FB 0045B          CALLS   #1, DBG$NSYNTAX_ERROR
                68              50  D0 00462 51$:     MOVL    R0, (R8)
                                42  11 00465          BRB     57$
                                7E  D4 00467 52$:     CLRL    -(SP)                                   ; 2072
                                58  DD 00469          PUSHL   R8                                      ; 2075
                7E                  22  7D 0046B          MOVQ    #34, -(SP)                          ; 2072
                          04    A2  9F 0046E          PUSHAB  4(TEMP_KEY_DESC)
                                53  DD 00471          PUSHL   R3
00000000G   00                  06  FB 00473          CALLS   #6, DBG$NACCEPT_STRING
                5B              50  D0 0047A          MOVL    R0, STATUS
                29              5B  E9 0047D          BLBC    STATUS, 57$                             ; 2078
                62          04  B2  9B 00480          MOVZBW  @4(TEMP_KEY_DESC), (TEMP_KEY_DESC)      ; 2084
                          04    A2  D6 00484          INCL    4(TEMP_KEY_DESC)
                04  A5              52  D0 00487 53$:     MOVL    TEMP_KEY_DESC, 4(NOUN_NODE)         ; 2088
                            FC20    31  00488          BRW     6$                                      ; 2044
                          08    A7  D5 0048E 54$:     TSTL    8(R7)                                   ; 2101
                                05  13 00491          BEQL    55$
                          04    A5  D5 00493          TSTL    4(NOUN_NODE)                            ; 2107
                                15  12 00496          BNEQ    58$
                   000280D0     8F  DD 00498 55$:     PUSHL   #164048                                 ; 2110
00000000G   00                  01  FB 0049E          CALLS   #1, DBG$NMAKE_ARG_VECT
          0C    BC              50  D0 004A5 56$:     MOVL    R0, @MESSAGE_VECT
                50                  04  D0 004A9 57$:     MOVL    #4, R0                              ; 2111
                                    04  004AC          RET
                50                  01  D0 004AD 58$:     MOVL    #1, R0                              ; 2114
                                    04  004B0          RET                                             ; 2115

; Routine Size:  1201 bytes,    Routine Base:  DBG$CODE + 071A

; 1994          2116  1
```

```
;  1996          2117  1 GLOBAL ROUTINE dbg$nparse_delete (input_desc, verb_node, message_vect) =
;  1997          2118  1 !++
;  1998          2119  1 ! Functional Description
;  1999          2120  1 !
;  2000          2121  1 !       This is the top-level parse network for the DELETE command.
;  2001          2122  1 !       DELETE is the same as UNDEFINE so we just call that parse
;  2002          2123  1 !       network.
;  2003          2124  1 !
;  2004          2125  1 ! Inputs
;  2005          2126  1 !       input_desc      - String descriptor for the remaining input.
;  2006          2127  1 !       verb_node       - Top-level node in the command tree.
;  2007          2128  1 !       message_vect    - error message vector.
;  2008          2129  1 !
;  2009          2130  1 ! Routine Outputs
;  2010          2131  1 !
;  2011          2132  1 !       A command execution tree is constructed starting at the verb
;  2012          2133  1 !       node:
;  2013          2134  1 !
;  2014          2135  1 !        ------      ------      ------
;  2015          2136  1 !       ! VERB ! -> ! NOUN ! -> ! NOUN ! -> ...
;  2016          2137  1 !        ------      ------      ------
;  2017          2138  1 !
;  2018          2139  1 !       In each noun node, all the following fields contain information
;  2019          2140  1 !       about the symbol being undefined:
;  2020          2141  1 !       DBG$L_NOUN_VALUE -      Points to a counted string with the name of
;  2021          2142  1 !                               the symbol (the left-hand-side of the
;  2022          2143  1 !                               definition).
;  2023          2144  1 !       DBG$L_ADJECTIVE_PTR -   Contains an indication of whether the
;  2024          2145  1 !                               definition was local (=) or global (==).
;  2025          2146  1 !       The string descriptor is updated to point past the
;  2026          2147  1 !       input that has been parsed. A completion code is returned:
;  2027          2148  1 !       STS$K_SUCCESS - The input was successfully parsed.
;  2028          2149  1 !       STS$K_SEVERE -  There were errors during the parse. An error
;  2029          2150  1 !                       message vector is constructed and returned
;  2030          2151  1 !                       in message_vect.
;  2031          2152  1 !--
;  2032          2153  2     BEGIN
;  2033          2154  2     RETURN dbg$nparse_undefine(.input_desc, .verb_node, .message_vect);
;  2034          2155  1     END;


                                        0000 00000          .ENTRY  DBG$NPARSE_DELETE, Save nothing                 ; 2117
                              7E    08  AC  7D 00002          MOVQ    VERB_NODE, -(SP)                                ; 2154
                                    04  AC  DD 00006          PUSHL   INPUT_DESC                                      ;
                        0000V       CF      03  FB 00009          CALLS   #3, DBG$NPARSE_UNDEFINE                     ;
                                            04 0000E            RET                                                 ; 2155

; Routine Size:  15 bytes,    Routine Base:  DBG$CODE + 0BCB
```

```
;  2036     2156  1 GLOBAL ROUTINE dbg$nparse_undefine (input_desc, verb_node, message_vect) =
;  2037     2157  1 !++
;  2038     2158  1 ! Functional Description
;  2039     2159  1 !
;  2040     2160  1 !       This is the top-level parse network for the UNDEFINE command.
;  2041     2161  1 !
;  2042     2162  1 ! Routine Inputs
;  2043     2163  1 !
;  2044     2164  1 !       input_desc -    A string descriptor for the remaining input.
;  2045     2165  1 !       verb_node -     A pointer to the verb node for UNDEFINE, which
;  2046     2166  1 !                           will be the top-level node in the command
;  2047     2167  1 !                           execution tree.
;  2048     2168  1 !       message_vect -  An error message vector
;  2049     2169  1 !
;  2050     2170  1 ! Routine Outputs
;  2051     2171  1 !
;  2052     2172  1 !       A command execution tree is constructed starting at the verb
;  2053     2173  1 !       node:
;  2054     2174  1 !
;  2055     2175  1 !        ------      ------      ------
;  2056     2176  1 !       | VERB | -> | NOUN | -> | NOUN | -> ...
;  2057     2177  1 !        ------      ------      ------
;  2058     2178  1 !
;  2059     2179  1 !       In each noun node, all the following fields contain information
;  2060     2180  1 !       about the symbol being undefined:
;  2061     2181  1 !       DBG$L_NOUN_VALUE -      Points to a counted string with the name of
;  2062     2182  1 !                               the symbol (the left-hand-side of the
;  2063     2183  1 !                               definition).
;  2064     2184  1 !       DBG$L_ADJECTIVE_PTR -   Contains an indication of whether the
;  2065     2185  1 !                               definition was local (=) or global (==).
;  2066     2186  1 !       The string descriptor is updated to point past the
;  2067     2187  1 !       input that has been parsed. A completion code is returned:
;  2068     2188  1 !       STS$K_SUCCESS - The input was successfully parsed.
;  2069     2189  1 !       STS$K_SEVERE -  There were errors during the parse. An error
;  2070     2190  1 !                       message vector is constructed and returned
;  2071     2191  1 !                       in message_vect.
;  2072     2192  1 !--
;  2073     2193  2     BEGIN
;  2074     2194  2
;  2075     2195  2     MAP
;  2076     2196  2         input_desc : REF BLOCK [,BYTE], ! The string descriptor for the
;  2077     2197  2                                         !    remaining input.
;  2078     2198  2         verb_node : REF dbg$verb_node;  ! The verb node with the UNDEFINE verb.
;  2079     2199  2
;  2080     2200  2     BIND
;  2081     2201  2         dbg$cs_all      = UPLIT BYTE (3, 'ALL'),
;  2082     2202  2         dbg$cs_global   = UPLIT BYTE (6, 'GLOBAL'),
;  2083     2203  2         dbg$cs_key      = UPLIT BYTE (3, 'KEY'),
;  2084     2204  2         dbg$cs_local    = UPLIT BYTE (5, 'LOCAL'),
;  2085     2205  2         dbg$cs_comma    = UPLIT BYTE (1, dbg$k_comma),
;  2086     2206  2         dbg$cs_cr       = UPLIT BYTE (1, dbg$k_car_return),
;  2087     2207  2         dbg$cs_slash    = UPLIT BYTE (1, dbg$k_slash);
;  2088     2208  2
;  2089     2209  2     LOCAL
;  2090     2210  2         all_flag,                       ! Flag for UNDEFINE/ALL
;  2091     2211  2         first_time,                     ! Flag for first time around the main loop
;  2092     2212  2         global_flag,                    ! Flag for UNDEFINE/GLOBAL
```

```
;  2093          2213  2          new_noun_node: REF dbg$noun_node,   ! Pointer to a noun node
;  2094          2214  2          noun_node: REF dbg$noun_node,       ! Pointer to a noun node
;  2095          2215  2          saved_input_desc: dbg$str_desc;     ! Copy of input desc
;  2096          2216  2
;  2097          2217  2
;  2098          2218  2      ! Check for UNDEFINE/KEY.
;  2099          2219  2      !
;  2100          2220  2
;  2101          2221  2      ch$move(8, .input_desc, saved_input_desc);
;  2102          2222  2      IF dbg$nmatch (saved_input_desc, dbg$cs_slash, 1)
;  2103          2223  2      THEN
;  2104          2224  2          IF dbg$nmatch (saved_input_desc, dbg$cs_key, 1)
;  2105          2225  2          THEN
;  2106          2226  3              BEGIN
;  2107          2227  3              LOCAL
;  2108          2228  3                  status;
;  2109          2229  3
;  2110          2230  3              ch$move(8, saved_input_desc, .input_desc);
;  2111          2231  3              status = dbg$nparse_del_key(.input_desc, .verb_node, .message_vect);
;  2112          2232  3              IF NOT .status
;  2113          2233  3              THEN
;  2114          2234  3                  RETURN sts$k_severe;
;  2115          2235  3              RETURN sts$k_success;
;  2116          2236  3              END;
;  2117          2237  2
;  2118          2238  2      ! Initialize flags.
;  2119          2239  2      !
;  2120          2240  2      all_flag = FALSE;
;  2121          2241  2      first_time = TRUE;
;  2122          2242  2      global_flag = TRUE;
;  2123          2243  2      verb_node [dbg$b_verb_composite] = 0;
;  2124          2244  2
;  2125          2245  2      ! Look for qualifiers.
;  2126          2246  2      !
;  2127          2247  2      WHILE dbg$nmatch (.input_desc, dbg$cs_slash, 1) DO
;  2128          2248  2          SELECTONE TRUE OF
;  2129          2249  2              SET
;  2130          2250  2
;  2131          2251  2              [dbg$nmatch (.input_desc, dbg$cs_all, 1)]:
;  2132          2252  2                  all_flag = TRUE;
;  2133          2253  2
;  2134          2254  2              ! /GLOBAL is not allowed since it is the default.
;  2135          2255  2              !
;  2136          2256  2              ! [dbg$nmatch (.input_desc, dbg$cs_global, 1)]:
;  2137          2257  2              !     global_flag = TRUE;
;  2138          2258  2
;  2139          2259  2              [dbg$nmatch (.input_desc, dbg$cs_local, 1)]:
;  2140          2260  2                  global_flag = FALSE;
;  2141          2261  2
;  2142          2262  2              [OTHERWISE] :
;  2143          2263  2                  report_error;
;  2144          2264  2
;  2145          2265  2              TES;
;  2146          2266  2
;  2147          2267  2      ! For UNDEFINE/ALL, we don't pick up a list of names
;  2148          2268  2      !
;  2149          2269  2      IF .all_flag
```

```
;  2150          2270  2      THEN
;  2151          2271                BEGIN
;  2152          2272                IF .global_flag
;  2153          2273                THEN
;  2154          2274                    verb_node [dbg$b_verb_composite] = undefine_all_global
;  2155          2275                ELSE
;  2156          2276                    verb_node [dbg$b_verb_composite] = undefine_all;
;  2157          2277                RETURN sts$k_success;
;  2158          2278                END;
;  2159          2279
;  2160          2280            ! Loop through the list of names to be undefined.
;  2161          2281            !
;  2162          2282            WHILE TRUE DO
;  2163          2283                BEGIN
;  2164          2284
;  2165          2285                ! Check for end of line. It is an error at this point.
;  2166          2286                !
;  2167          2287                IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  2168          2288                THEN
;  2169          2289  4                 BEGIN
;  2170          2290  4                 .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  2171          2291  4                 RETURN sts$k_severe;
;  2172          2292                    END;
;  2173          2293
;  2174          2294                ! Allocate a noun node to hold the name
;  2175          2295                !
;  2176          2296                new_noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  2177          2297
;  2178          2298                ! We must link in the noun node. If this is the first definition
;  2179          2299                ! in the list, it is attached to the verb node; otherwise it
;  2180          2300                ! is attached to the previous noun node.
;  2181          2301                !
;  2182          2302                IF .first_time
;  2183          2303                THEN
;  2184          2304  4                 BEGIN
;  2185          2305  4                 first_time = FALSE;
;  2186          2306  4                 verb_node [dbg$l_verb_object_ptr] = .new_noun_node;
;  2187          2307  4                 noun_node = .new_noun_node;
;  2188          2308  4                 noun_node [dbg$l_noun_link] = 0
;  2189          2309  4                 END
;  2190          2310  4
;  2191          2311  3                 ELSE
;  2192          2312  4                 BEGIN
;  2193          2313  4                 noun_node [dbg$l_noun_link] = .new_noun_node;
;  2194          2314  4                 noun_node = .new_noun_node;
;  2195          2315  4                 noun_node [dbg$l_noun_link] = 0
;  2196          2316                    END;
;  2197          2317
;  2198          2318                ! Now we read the name that is being undefined and store a
;  2199          2319                ! pointer to the counted string in the value field of
;  2200          2320                ! the noun node.
;  2201          2321                !
;  2202          2322                IF NOT dbg$nread_name (.input_desc,
;  2203          2323                       noun_node [dbg$l_noun_value],
;  2204          2324                       .message_vect)
;  2205          2325                THEN
;  2206          2326  3                 RETURN sts$k_severe;
```

```
;  2207          2327  3
;  2208          2328  3          ! Check for the global qualifier on the name.
;  2209          2329  3          !
;  2210          2330  3          IF dbg$nmatch (.input_desc, dbg$cs_slash, 1)
;  2211          2331  3          THEN
;  2212          2332  3              IF dbg$nmatch (.input_desc, dbg$cs_global, 1)
;  2213          2333  3              THEN
;  2214          2334  3                  noun_node [dbg$l_adjective_ptr] = define_global
;  2215          2335  3              ELSE
;  2216          2336  3                  report_error;
;  2217          2337  3          IF .global_flag
;  2218          2338  3          THEN
;  2219          2339  3              noun_node [dbg$l_adjective_ptr] = define_global;
;  2220          2340  3
;  2221          2341  3          ! Check for exhausted input. If so, exit the loop.
;  2222          2342  3          !
;  2223          2343  3          IF .input_desc [dsc$w_length] EQL 0
;  2224          2344  3          THEN
;  2225          2345  3              EXITLOOP;
;  2226          2346  3
;  2227          2347  3          ! Now check for comma, indicating more in the list.
;  2228          2348  3          !
;  2229          2349  3          IF NOT dbg$nmatch (.input_desc, dbg$cs_comma, 1)
;  2230          2350  3          THEN
;  2231          2351  3              EXITLOOP;
;  2232          2352  3          END;
;  2233          2353  2
;  2234          2354  2      ! We are all done. Return success.
;  2235          2355  2      !
;  2236          2356  2      RETURN sts$k_success;
;  2237          2357  1      END;
; INFO#250                  L1:2313
; Referenced LOCAL symbol NOUN_NODE is probably not initialized


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

                                                03  000DC P.ABL:   .BYTE   3                                ;
                                        4C  4C  41  000DD           .ASCII  \ALL\                           ;
                                                06  000E0 P.ABM:    .BYTE   6                               ;
                                4C  41  42  4F  4C  47  000E1       .ASCII  \GLOBAL\                        ;
                                                03  000E7 P.ABN:    .BYTE   3                               ;
                                        59  45  4B  000E8           .ASCII  \KEY\                           ;
                                                05  000EB P.ABO:    .BYTE   5                               ;
                                    4C  41  43  4F  4C  000EC       .ASCII  \LOCAL\                         ;
                                                2C  01  000F1 P.ABP:    .BYTE   1, 44                       ;
                                                0D  01  000F3 P.ABQ:    .BYTE   1, 13                       ;
                                                2F  01  000F5 P.ABR:    .BYTE   1, 47                       ;

                                                        DBG$CS_ALL=         P.ABL
                                                        DBG$CS_GLOBAL=      P.ABM
                                                        DBG$CS_KEY=         P.ABN
                                                        DBG$CS_LOCAL=       P.ABO
                                                        DBG$CS_COMMA=       P.ABP
                                                        DBG$CS_CR=          P.ABQ
                                                        DBG$CS_SLASH=       P.ABR
```

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

                                03FC 00000      .ENTRY  DBG$NPARSE_UNDEFINE, Save R2,R3,R4,R5,R6,-      ; 2156
                                                        R7,R8,R9
                59 00000000'    EF   9E 00002           MOVAB   DBG$CS_SLASH, R9                        ;
                58 00000000G    00   9E 00009           MOVAB   DBG$NMATCH, R8                          ;
                5E              0C   C2 00010           SUBL2   #12, SP                                 ;
                56        04    AC   D0 00013           MOVL    INPUT_DESC, R6                          ; 2221
             6E 66              08   28 00017           MOVC3   #8, (R6), SAVED_INPUT_DESC              ;
                                01   DD 0001B           PUSHL   #1                                      ; 2222
                                59   DD 0001D           PUSHL   R9                                      ;
                                08   AE 9F 0001F       PUSHAB  SAVED_INPUT_DESC                        ;
                68              03   FB 00022           CALLS   #3, DBG$NMATCH                          ;
                23              50   E9 00025           BLBC    R0, 1$                                  ;
                                01   DD 00028           PUSHL   #1                                      ; 2224
                                F2   A9 9F 0002A       PUSHAB  DBG$CS_KEY                              ;
                                08   AE 9F 0002D       PUSHAB  SAVED_INPUT_DESC                        ;
                68              03   FB 00030           CALLS   #3, DBG$NMATCH                          ;
                15              50   E9 00033           BLBC    R0, 1$                                  ;
             66 6E              08   28 00036           MOVC3   #8, SAVED_INPUT_DESC, (R6)              ; 2230
                7E        08    AC   7D 0003A           MOVQ    VERB_NODE, -(SP)                        ; 2231
                                56   DD 0003E           PUSHL   R6                                      ;
          0000V CF              03   FB 00040           CALLS   #3, DBG$NPARSE_DEL_KEY                  ;
                65              50   E8 00045           BLBS    STATUS, 7$                              ; 2232
                          00F3  31   00048           BRW     15$                                     ; 2234
                                52   D4 0004B 1$:       CLRL    ALL_FLAG                                ; 2240
                57              01   D0 0004D           MOVL    #1, FIRST_TIME                          ; 2241
                55              01   D0 00050           MOVL    #1, GLOBAL_FLAG                         ; 2242
                53        08    AC   D0 00053           MOVL    VERB_NODE, R3                           ; 2243
                          01    A3   94 00057           CLRB    1(R3)                                   ;
                                01   DD 0005A 2$:       PUSHL   #1                                      ; 2247
                        0240    8F   BB 0005C           PUSHR   #^M<R6,R9>                              ;
                68              03   FB 00060           CALLS   #3, DBG$NMATCH                          ;
                37              50   E9 00063           BLBC    R0, 5$                                  ;
                                01   DD 00066           PUSHL   #1                                      ; 2251
                          E7    A9   9F 00068           PUSHAB  DBG$CS_ALL                              ;
                                56   DD 0006B           PUSHL   R6                                      ;
                68              03   FB 0006D           CALLS   #3, DBG$NMATCH                          ;
                01              50   D1 00070           CMPL    R0, #1                                  ;
                                05   12 00073           BNEQ    3$                                      ;
                52              01   D0 00075           MOVL    #1, ALL_FLAG                            ; 2252
                                E0   11 00078           BRB     2$                                      ;
                                01   DD 0007A 3$:       PUSHL   #1                                      ; 2259
                          F6    A9   9F 0007C           PUSHAB  DBG$CS_LOCAL                            ;
                                56   DD 0007F           PUSHL   R6                                      ;
                68              03   FB 00081           CALLS   #3, DBG$NMATCH                          ;
                01              50   D1 00084           CMPL    R0, #1                                  ;
                                04   12 00087           BNEQ    4$                                      ;
                                55   D4 00089           CLRL    GLOBAL_FLAG                             ; 2260
                                CD   11 0008B           BRB     2$                                      ;
                                01   DD 0008D 4$:       PUSHL   #1                                      ; 2262
                          FE    A9   9F 0008F           PUSHAB  DBG$CS_CR                               ;
                                56   DD 00092           PUSHL   R6                                      ;
                68              03   FB 00094           CALLS   #3, DBG$NMATCH                          ;
                7F              50   E8 00097           BLBS    R0, 12$                                 ;
```

```
                        008B  31  0009A          BRW     13$                                             ;
                    10    52  E9  0009D  5$:     BLBC    ALL_FLAG, 8$                                    ; 2269
                    06    55  E9  000A0          BLBC    GLOBAL_FLAG, 6$                                 ; 2272
                01  A3    02  90  000A3          MOVB    #2, 1(R3)                                       ; 2274
                          04  11  000A7          BRB     7$                                              ;
                01  A3    01  90  000A9  6$:     MOVB    #1, 1(R3)                                       ; 2276
                        00AD  31  000AD  7$:     BRW     18$                                             ; 2277
                          01  DD  000B0  8$:     PUSHL   #1                                              ; 2287
                      FE  A9  9F  000B2          PUSHAB  DBG$CS_CR                                       ;
                          56  DD  000B5          PUSHL   R6                                              ;
                    68    03  FB  000B7          CALLS   #3, DBG$NMATCH                                  ;
                    5C    50  E8  000BA          BLBS    R0, 12$                                         ;
                          04  DD  000BD          PUSHL   #4                                              ; 2296
00000000G           00    01  FB  000BF          CALLS   #1, DBG$GET_TEMPMEM                             ;
                    54    50  D0  000C6          MOVL    R0, NEW_NOUN_NODE                               ;
                    08    57  E9  000C9          BLBC    FIRST_TIME, 9$                                  ; 2302
                          57  D4  000CC          CLRL    FIRST_TIME                                      ; 2305
                08  A3    54  D0  000CE          MOVL    NEW_NOUN_NODE, 8(R3)                            ; 2306
                          04  11  000D2          BRB     10$                                             ; 2307
                08  A2    54  D0  000D4  9$:     MOVL    NEW_NOUN_NODE, 8(NOUN_NODE)                     ; 2313
                    52    54  D0  000D8  10$:    MOVL    NEW_NOUN_NODE, NOUN_NODE                        ; 2314
                      08  A2  D4  000DB          CLRL    8(NOUN_NODE)                                    ; 2315
                      0C  AC  DD  000DE          PUSHL   MESSAGE_VECT                                    ; 2324
                          52  DD  000E1          PUSHL   NOUN_NODE                                       ; 2323
                          56  DD  000E3          PUSHL   R6                                              ;
              0000V       CF  03  FB  000E5          CALLS   #3, DBG$NREAD_NAME                          ;
                    51    50  E9  000EA          BLBC    R0, 15$                                         ;
                          01  DD  000ED          PUSHL   #1                                              ; 2330
                    0240  8F  BB  000EF          PUSHR   #^M<R6,R9>                                      ;
                    68    03  FB  000F3          CALLS   #3, DBG$NMATCH                                  ;
                    49    50  E9  000F6          BLBC    R0, 16$                                         ;
                          01  DD  000F9          PUSHL   #1                                              ; 2332
                      EB  A9  9F  000FB          PUSHAB  DBG$CS_GLOBAL                                   ;
                          56  DD  000FE          PUSHL   R6                                              ;
                    68    03  FB  00100          CALLS   #3, DBG$NMATCH                                  ;
                    06    50  E9  00103          BLBC    R0, 11$                                         ;
                04  A2    01  D0  00106          MOVL    #1, 4(NOUN_NODE)                                ; 2334
                          36  11  0010A          BRB     16$                                             ;
                          01  DD  0010C  11$:     PUSHL   #1                                              ; 2335
                      FE  A9  9F  0010E          PUSHAB  DBG$CS_CR                                       ;
                          56  DD  00111          PUSHL   R6                                              ;
                    68    03  FB  00113          CALLS   #3, DBG$NMATCH                                  ;
                    0F    50  E9  00116          BLBC    R0, 13$                                         ;
                      000280D0  8F  DD  00119  12$:     PUSHL   #164048                                 ;
00000000G           CC    01  FB  0011F          CALLS   #1, DBG$NMAKE_ARG_VECT                          ;
                          12  11  00126          BRB     14$                                             ;
                          56  DD  00128  13$:     PUSHL   R6                                             ;
00000000G           00    01  FB  0012A          CALLS   #1, DBG$NNEXT_WORD                              ;
                          50  DD  00131          PUSHL   R0                                              ;
00000000G           00    01  FB  00133          CALLS   #1, DBG$NSYNTAX_ERROR                           ;
                    0C  BC    50  D0  0013A  14$:     MOVL    R0, @MESSAGE_VECT                           ;
                    50    04  D0  0013E  15$:     MOVL    #4, R0                                          ;
                              04  00141          RET                                                     ;
                    04    55  E9  00142  16$:     BLBC    GLOBAL_FLAG, 17$                                ; 2337
                04  A2    01  D0  00145          MOVL    #1, 4(NOUN_NODE)                                ; 2339
                          66  B5  00149  17$:     TSTW    (R6)                                            ; 2343
                          10  13  0014B          BEQL    18$                                             ;
```

```
                        01  DD 0014D          PUSHL   #1                                      ; 2349
                    FC  A9  9F 0014F          PUSHAB  DBG$CS_COMMA                            ;
                        56  DD 00152          PUSHL   R6                                      ;
                68      03  FB 00154          CALLS   #3, DBG$NMATCH                          ;
                03      50  E9 00157          BLBC    R0, 18$                                 ;
                    FF53    31 0015A          BRW     8$                                      ;
                50      01  D0 0015D 18$:     MOVL    #1, R0                                  ; 2356
                            04 00160          RET                                             ; 2357

; Routine Size:  353 bytes,    Routine Base:  DBG$CODE + 0BDA
```

```
;  2239     2358  1 ROUTINE dbg$nparse_del_key (input_desc, verb_node, message_vect) =
;  2240     2359  1 !++
;  2241     2360  1 ! Functional Description
;  2242     2361  1 !
;  2243     2362  1 !       This is the parse network for the DELETE/KEY command.
;  2244     2363  1 !
;  2245     2364  1 ! Routine Inputs
;  2246     2365  1 !
;  2247     2366  1 !       input_desc      -       A pointer to a string descriptor for the
;  2248     2367  1 !                                 remaining input.
;  2249     2368  1 !       verb_node       -       A pointer to the verb node for the DELETE
;  2250     2369  1 !                                 command, which will be the top-level
;  2251     2370  1 !                                 node in the command execution tree.
;  2252     2371  1 !       message_vect    -       A pointer to an error message vector.
;  2253     2372  1 !
;  2254     2373  1 ! Routine Outputs
;  2255     2374  1 !
;  2256     2375  1 !       A command execution tree is constructed starting at the verb
;  2257     2376  1 !       node:
;  2258     2377  1 !
;  2259     2378  1 !        ------      ------
;  2260     2379  1 !       | VERB | -> | NOUN |
;  2261     2380  1 !        ------      ------
;  2262     2381  1 !          |                          The DBG$B_VERB_COMPOSITE field contains a
;  2263     2382  1 !        -------                      value for the DELETE/KEY command.
;  2264     2383  1 !       |ADVERB0|
;  2265     2384  1 !        -------                      The noun node has the following information:
;  2266     2385  1 !          |
;  2267     2386  1 !        -------                      DBG$L_NOUN_VALUE - A pointer to a descriptor
;  2268     2387  1 !       |ADVERB1|                           that contains the key-name.
;  2269     2388  1 !        -------
;  2270     2389  1 !          |
;  2271     2390  1 !        -------
;  2272     2391  1 !       |ADVERB2|                     The adverb nodes appear as follows:
;  2273     2392  1 !        -------
;  2274     2393  1 !                                    DBG$L_ADVERB_VALUE - Value or location of data
;  2275     2394  1 !                                           for this qualifier.
;  2276     2395  1 !                                    DBG$L_ADVERB_LINK - Link to next Adverb-node.
;  2277     2396  1 !
;  2278     2397  1 !       The string descriptor is updated to point past the
;  2279     2398  1 !       input that has been parsed. A completion code is returned:
;  2280     2399  1 !
;  2281     2400  1 !       STS$K_SUCCESS - The input was successfully parsed.
;  2282     2401  1 !       STS$K_SEVERE -  There were errors during the parse. An error
;  2283     2402  1 !                       message vector is constructed and returned
;  2284     2403  1 !                       in message_vect.
;  2285     2404  1 !--
;  2286     2405  2     BEGIN
;  2287     2406  2
;  2288     2407  2     MAP
;  2289     2408  2         input_desc              : REF BLOCK [,BYTE],    ! String descriptor
;  2290     2409  2         verb_node               : REF dbg$verb_node;
;  2291     2410  2
;  2292     2411  2     BIND
;  2293     2412  2         dbg$cs_all              = UPLIT BYTE (3, 'ALL'),
;  2294     2413  2         dbg$cs_log              = UPLIT BYTE (3, 'LOG'),
;  2295     2414  2         dbg$cs_state            = UPLIT BYTE (5, 'STATE'),
```

```
;  2296          2415  2          dbg$cs_NO                = UPLIT BYTE ('NO'),
;  2297          2416  2          dbg$cs_left_paren        = UPLIT BYTE (1, dbg$k_left_parenthesis),
;  2298          2417  2          dbg$cs_right_paren       = UPLIT BYTE (1, dbg$k_right_parenthesis),
;  2299          2418  2          dbg$cs_comma             = UPLIT BYTE (1, dbg$k_comma),
;  2300          2419  2          dbg$cs_cr                = UPLIT BYTE (1, dbg$k_car_return),
;  2301          2420  2          dbg$cs_equal             = UPLIT BYTE (1, dbg$k_equal),
;  2302          2421  2          dbg$cs_slash             = UPLIT BYTE (1, dbg$k_slash);
;  2303          2422  2
;  2304          2423  2      LITERAL
;  2305          2424  2          dbg$k_lowest_qualifier  = 0,    ! Corresponds to the adverb-nodes
;  2306          2425  2          dbg$k_all               = 0,    ! in the structure that is built.
;  2307          2426  2          dbg$k_log               = 1,
;  2308          2427  2          dbg$k_state             = 2,
;  2309          2428  2          dbg$k_highest_qualifier = 2;
;  2310          2429  2
;  2311          2430  2      LOCAL
;  2312          2431  2          all_flag                : INITIAL(FALSE),       ! True if /ALL
;  2313          2432  2          define_kind             : INITIAL(0),           ! Value of DELETE/KEY qualifier
;  2314          2433  2          noun_node               : REF dbg$noun_node,    ! Pointer to a noun node
;  2315          2434  2          new_noun_node           : REF dbg$noun_node,    ! Another pointer to a noun node
;  2316          2435  2          adverb_node             : REF dbg$adverb_node,  ! Pointer to a noun node
;  2317          2436  2          new_adverb_node         : REF dbg$adverb_node,  ! Another pointer to a adverb node
;  2318          2437  2          state_name_node         : REF dbg$state_name_node,      ! Pointer to a state-name node
;  2319          2438  2          new_state_name_node     : REF dbg$state_name_node,      ! Another pointer to a state-name node
;  2320          2439  2          ptr                     : REF VECTOR[,BYTE],    ! Points into input string
;  2321          2440  2          status,
;  2322          2441  2          temp_key_desc           : REF dbg$stg_desc,     ! String desc. for DELETE/KEY symbols
;  2323          2442  2          d_key_no,                                       ! Flag for NOxxx qualifier
;  2324          2443  2          define_key_value;                               ! Value for the qualifier
;  2325          2444  2
;  2326          2445  2
;  2327          2446  2      ! Check whether we are on a system that allows keypad input.
;  2328          2447  2      !
;  2329          2448  2      IF NOT .dbg$gb_keypad_input
;  2330          2449  2      THEN
;  2331          2450  2          SIGNAL(dbg$_nokeydef);
;  2332          2451  2
;  2333          2452  2      ! Fill in the fact that this is a DELETE/KEY command in the verb node.
;  2334          2453  2      ! And clear the noun link value.
;  2335          2454  2      !
;  2336          2455  2      verb_node [dbg$b_verb_composite] = undefine_key;
;  2337          2456  2      verb_node [dbg$l_verb_object_ptr] = 0;
;  2338          2457  2
;  2339          2458  2      ! Build adverb list with defaults.
;  2340          2459  2      !
;  2341          2460  2
;  2342          2461  2      new_adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);  ! Get first node
;  2343          2462  2
;  2344          2463  2      verb_node [dbg$l_verb_adverb_ptr] = .new_adverb_node;
;  2345          2464  2      adverb_node = .new_adverb_node;
;  2346          2465  2
;  2347          2466  2      adverb_node [dbg$b_adverb_literal] = dbg$k_lowest_qualifier;        ! Initialize first node
;  2348          2467  2      adverb_node [dbg$l_adverb_value]   = 0;
;  2349          2468  2      adverb_node [dbg$l_adverb_link]    = 0;
;  2350          2469  2
;  2351          2470  2      define_kind = dbg$k_lowest_qualifier + 1;
;  2352          2471  2      WHILE .define_kind [EQ dbg$k_highest_qualifier DO           ! Build rest of adverb list
```

```
;  2353          2472  3          BEGIN
;  2354          2473  3          new_adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);
;  2355          2474  3          adverb_node [dbg$l_adverb_link] = .new_adverb_node;
;  2356          2475  3          adverb_node = .new_adverb_node;
;  2357          2476  3
;  2358          2477  3          adverb_node [dbg$b_adverb_literal] = .define_kind;
;  2359          2478  3          IF .define_kind EQL dbg$k_state
;  2360          2479  3          THEN
;  2361          2480  4              BEGIN
;  2362          2481  4              ! If the adverb-node is for the state name, let the adverb-node-value
;  2363          2482  4              ! be a pointer to a state-name-node that points to a descriptor with
;  2364          2483  4              ! the current state name.
;  2365          2484  4              !
;  2366          2485  4              temp_key_desc = dbg$get_tempmem(2);
;  2367          2486  4              temp_key_desc [dsc$w_length]  = 0;
;  2368          2487  4              temp_key_desc [dsc$b_dtype]   = dsc$k_dtype_t;
;  2369          2488  4              temp_key_desc [dsc$b_class]   = dsc$k_class_d;
;  2370          2489  4              temp_key_desc [dsc$a_pointer] = 0;
;  2371          2490  4
;  2372          2491  4              ! This will return the current state name
;  2373          2492  4              !
;  2374          2493  4              smg$set_default_state(dbg$gl_key_table_id, 0, .temp_key_desc);
;  2375          2494  4
;  2376          2495  4              state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  2377          2496  4              state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  2378          2497  4              state_name_node [dbg$l_state_name_link] = 0;
;  2379          2498  4              adverb_node [dbg$l_adverb_value] = .state_name_node;
;  2380          2499  4              END
;  2381          2500  3          ELSE
;  2382          2501  3              ! In all other cases, let the default be zero.
;  2383          2502  3              !
;  2384          2503  3              adverb_node [dbg$l_adverb_value] = 0;
;  2385          2504  3
;  2386          2505  3          define_kind = .define_kind + 1;
;  2387          2506  2          END;
;  2388          2507  2      adverb_node [dbg$l_adverb_link] = 0;
;  2389          2508  2
;  2390          2509  2      WHILE (NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)) AND
;  2391          2510  2            (.input_desc [dsc$w_length] GTR 0) DO
;  2392          2511  2
;  2393          2512  3          BEGIN
;  2394          2513  3          IF dbg$nmatch(.input_desc, dbg$cs_slash, 1)
;  2395          2514  3          THEN
;  2396          2515  4              BEGIN
;  2397          2516  4
;  2398          2517  4              ! Find out what kind of qualifier it is
;  2399          2518  4              !
;  2400          2519  4              ! Initialize value
;  2401          2520  4
;  2402          2521  4              define_key_value = 0;
;  2403          2522  4              d_key_NO = FALSE;
;  2404          2523  4
;  2405          2524  4              ! Check for a NO qualifier
;  2406          2525  4
;  2407          2526  4              WHILE CH$EQL(1, UPLIT(' '), 1, CH$PTR(.input_desc [dsc$a_pointer])) DO
;  2408          2527  5                  BEGIN
;  2409          2528  5                  input_desc [dsc$a_pointer] = CH$PLUS(.input_desc [dsc$a_pointer], 1);
```

```
;  2410          2529  5                 input_desc [dsc$w_length]  = .input_desc [dsc$w_length] - 1;
;  2411          2530  4                 END;
;  2412          2531  4
;  2413          2532  4             IF CH$EQL(2, CH$PTR(dbg$cs_NO), 2, CH$PTR(.input_desc [dsc$a_pointer]))
;  2414          2533  4             THEN
;  2415          2534  5                 BEGIN
;  2416          2535  5                 d_key_NO = TRUE;
;  2417          2536  5                 input_desc [dsc$a_pointer] = CH$PLUS(.input_desc [dsc$a_pointer], 2);
;  2418          2537  5                 input_desc [dsc$w_length]  = .input_desc [dsc$w_length] - 2;
;  2419          2538  4                 END;
;  2420          2539  4
;  2421          2540  4             ! Set Define_key with qualifier code, and get value of define_key_value.
;  2422          2541  4
;  2423          2542  4             SELECTONE TRUE  OF
;  2424          2543  4                 SET
;  2425          2544  4
;  2426          2545  4                 [dbg$nmatch(.input_desc, dbg$cs_all, 1)] :
;  2427          2546  5                     BEGIN
;  2428          2547  5                     define_kind = dbg$k_all;
;  2429          2548  5                     define_key_value = 1;
;  2430          2549  5                     all_flag = TRUE;
;  2431          2550  5                     IF .d_key_NO
;  2432          2551  5                     THEN
;  2433          2552  5                         report_error;
;  2434          2553  5                     IF .verb_node [dbg$l_verb_object_ptr] NEQ 0
;  2435          2554  5                     THEN
;  2436          2555  5                         SIGNAL(dbg$_conflict);
;  2437          2556  4                     END;
;  2438          2557  4
;  2439          2558  4                 [dbg$nmatch(.input_desc, dbg$cs_state, 1)] :
;  2440          2559  5                     BEGIN
;  2441          2560  5                     define_key_value = 0;
;  2442          2561  5                     define_kind = dbg$k_state;
;  2443          2562  5                     IF NOT .d_key_NO
;  2444          2563  5                     THEN
;  2445          2564  6                         BEGIN
;  2446          2565  6                         temp_key_desc = dbg$get_tempmem(2);
;  2447          2566  6                         temp_key_desc[dsc$w_length]  = 0;
;  2448          2567  6                         temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  2449          2568  6                         temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  2450          2569  6                         temp_key_desc[dsc$a_pointer] = 0;
;  2451          2570  6
;  2452          2571  6                         ! Look for =
;  2453          2572  6                         !
;  2454          2573  6
;  2455          2574  6                         IF NOT dbg$nmatch (.input_desc, dbg$cs_equal, 1)
;  2456          2575  6                         THEN
;  2457          2576  6                             report_error;
;  2458          2577  6
;  2459          2578  6                         ! Look for a left paren
;  2460          2579  6                         !
;  2461          2580  6
;  2462          2581  6                         IF dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  2463          2582  6                         THEN
;  2464          2583  7                             BEGIN
;  2465          2584  7
;  2466          2585  7                             ! Pick up the first state name
```

```
;  2467          2586  7                !
;  2468          2587  7                status = dbg$read_key_info (.input_desc,
;  2469          2588  7                                            .temp_key_desc,
;  2470          2589  7                                            .message_vect);
;  2471          2590  7                IF NOT .status
;  2472          2591  7                THEN
;  2473          2592  7                    RETURN sts$k_severe;
;  2474          2593  7
;  2475          2594  7                new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  2476          2595  7                state_name_node = .new_state_name_node;
;  2477          2596  7                state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  2478          2597  7                state_name_node [dbg$l_state_name_link] = 0;
;  2479          2598  7                define_key_value = .state_name_node;
;  2480          2599  7
;  2481          2600  7                WHILE dbg$nmatch (.input_desc, dbg$cs_comma, 1) DO
;  2482          2601  8                    BEGIN
;  2483          2602  8                    temp_key_desc = dbg$get_tempmem(2);
;  2484          2603  8                    temp_key_desc[dsc$w_length]  = 0;
;  2485          2604  8                    temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  2486          2605  8                    temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  2487          2606  8                    temp_key_desc[dsc$a_pointer] = 0;
;  2488          2607  8
;  2489          2608  8                    ! Pick up the next state name
;  2490          2609  8                    !
;  2491          2610  8                    status = dbg$read_key_info (.input_desc,
;  2492          2611  8                                                .temp_key_desc,
;  2493          2612  8                                                .message_vect);
;  2494          2613  8                    IF NOT .status
;  2495          2614  8                    THEN
;  2496          2615  8                        RETURN sts$k_severe;
;  2497          2616  8
;  2498          2617  8                    new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  2499          2618  8                    state_name_node [dbg$l_state_name_link] = .new_state_name_node;
;  2500          2619  8                    state_name_node = .new_state_name_node;
;  2501          2620  8                    state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  2502          2621  8                    state_name_node [dbg$l_state_name_link] = 0;
;  2503          2622  8
;  2504          2623  7                    END;
;  2505          2624  7
;  2506          2625  7                ! Eat right paren
;  2507          2626  7                !
;  2508          2627  7
;  2509          2628  7                IF NOT dbg$nmatch (.input_desc, dbg$cs_right_paren, 1)
;  2510          2629  7                THEN
;  2511          2630  7                    report_error;
;  2512          2631  7
;  2513          2632  7                END
;  2514          2633  7
;  2515          2634  6            ELSE
;  2516          2635  7                BEGIN
;  2517          2636  7
;  2518          2637  7                ! Pick up the only state name
;  2519          2638  7                !
;  2520          2639  7
;  2521          2640  7                status = dbg$read_key_info (.input_desc,
;  2522          2641  7                                            .temp_key_desc,
;  2523          2642  7                                            .message_vect);
```

```
;  2524          2643  7                    IF NOT .status
;  2525          2644  7                    THEN
;  2526          2645  7                        RETURN sts$k_severe;
;  2527          2646  7
;  2528          2647  7                    new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  2529          2648  7                    state_name_node = .new_state_name_node;
;  2530          2649  7                    state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  2531          2650  7                    state_name_node [dbg$l_state_name_link] = 0;
;  2532          2651  7                    define_key_value = .state_name_node;
;  2533          2652  7
;  2534          2653  6                    END;
;  2535          2654  6
;  2536          2655  5                END;
;  2537          2656  4            END;
;  2538          2657  4
;  2539          2658  4            [dbg$nmatch(.input_desc, dbg$cs_log, 3)] :
;  2540          2659  5                BEGIN
;  2541          2660  5                define_key_value = 0;
;  2542          2661  5                define_kind = dbg$k_log;
;  2543          2662  5                IF .d_key_NO THEN define_key_value = 1;
;  2544          2663  4                END;
;  2545          2664  4
;  2546          2665  4            [OTHERWISE] :
;  2547          2666  4                report_error;
;  2548          2667  4            TES;
;  2549          2668  4
;  2550          2669  4        ! Process the qualifier
;  2551          2670  4
;  2552          2671  4        IF .define_key_value NEQ 0
;  2553          2672  4        THEN
;  2554          2673  5            BEGIN
;  2555          2674  5            adverb_node = .verb_node [dbg$l_verb_adverb_ptr];
;  2556          2675  5            WHILE (.adverb_node [dbg$b_adverb_literal] NEQ .define_kind) DO
;  2557          2676  5                adverb_node = .adverb_node [dbg$l_adverb_link];
;  2558          2677  5            adverb_node [dbg$l_adverb_value] = .define_key_value;
;  2559          2678  4            END;
;  2560          2679  4
;  2561          2680  4        END ! End of qualifier look up
;  2562          2681  4
;  2563          2682  3    ELSE
;  2564          2683  3
;  2565          2684  3        ! Process key name
;  2566          2685  3
;  2567          2686  4        IF (.verb_node [dbg$l_verb_object_ptr] EQL 0) AND (NOT .all_flag)
;  2568          2687  3        THEN
;  2569          2688  4            BEGIN
;  2570          2689  4
;  2571          2690  4            ! Get key name
;  2572          2691  4            !
;  2573          2692  4
;  2574          2693  4            temp_key_desc = dbg$get_tempmem(2);
;  2575          2694  4            temp_key_desc[dsc$w_length]  = 0;
;  2576          2695  4            temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  2577          2696  4            temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  2578          2697  4            temp_key_desc[dsc$a_pointer] = 0;
;  2579          2698  4            status = dbg$read_key_info (.input_desc,
;  2580          2699  4                                        .temp_key_desc,
```

```
;  2581          2700  4                                   .message_vect);
;  2582          2701  4                 IF NOT .status
;  2583          2702  4                 THEN
;  2584          2703  4                     RETURN .status;
;  2585          2704  4
;  2586          2705  4                 ! Make noun node for key-name string
;  2587          2706  4                 !
;  2588          2707  4
;  2589          2708  4                 new_noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  2590          2709  4
;  2591          2710  4                 verb_node [dbg$l_verb_object_ptr] = .new_noun_node;
;  2592          2711  4                 noun_node = .new_noun_node;
;  2593          2712  4                 noun_node [dbg$l_noun_value] = .temp_key_desc;
;  2594          2713  4                 noun_node [dbg$l_adjective_ptr] = 0;
;  2595          2714  4                 noun_node [dbg$l_noun_link] = 0;
;  2596          2715  4                 END
;  2597          2716  4
;  2598          2717  3             ELSE
;  2599          2718  3                 report_error;
;  2600          2719  3
;  2601          2720  2         END;    ! End While
;  2602          2721  2
;  2603          2722  2     ! Check to see if key-name string or /ALL has been entered.
;  2604          2723  2     ! If not, return a message and error status.
;  2605          2724  2     !
;  2606          2725  2
;  2607          2726  3     IF (.verb_node [dbg$l_verb_object_ptr] EQL 0) AND (NOT .all_flag)
;  2608          2727  2     THEN
;  2609          2728  3         BEGIN
;  2610          2729  3         .message_vect = dbg$nmake_arg_vect(dbg$_needmore);
;  2611          2730  3         RETURN sts$k_severe;
;  2612          2731  2         END;
;  2613          2732  2
;  2614          2733  2     RETURN sts$k_success;
;  2615          2734  1     END;


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

                                          03  000F7 P.ABS:  .BYTE   3                                    ;
                                    4C    4C  41  000F8        .ASCII  \ALL\                             ;
                                          03  000FB P.ABT:  .BYTE   3                                    ;
                                    47    4F  4C  000FC        .ASCII  \LOG\                             ;
                                          05  000FF P.ABU:  .BYTE   5                                    ;
                              45    54    41  54  53  00100        .ASCII  \STATE\                       ;
                                          4F  4E  00105 P.ABV:  .ASCII  \NO\                            ;
                                          28  01  00107 P.ABW:  .BYTE   1, 40                           ;
                                          29  01  00109 P.ABX:  .BYTE   1, 41                           ;
                                          2C  01  0010B P.ABY:  .BYTE   1, 44                           ;
                                          0D  01  0010D P.ABZ:  .BYTE   1, 13                           ;
                                          3D  01  0010F P.ACA:  .BYTE   1, 61                           ;
                                          2F  01  00111 P.ACB:  .BYTE   1, 47                           ;
                                              00113        .BLKB   1                                    ;
                                    00    00  00  20  00114 P.ACC:  .ASCII  \ \<0><0><0>                ;

                                                    DBG$CS_ALL=         P.ABS
```

```
                        DBG$CS_LOG=          P.ABT
                        DBG$CS_STATE=        P.ABU
                        DBG$CS_NO=           P.ABV
                        DBG$CS_LEFT_PAREN=   P.ABW
                        DBG$CS_RIGHT_PAREN=  P.ABX
                        DBG$CS_COMMA=        P.ABY
                        DBG$CS_CR=           P.ABZ
                        DBG$CS_EQUAL=        P.ACA
                        DBG$CS_SLASH=        P.ACB


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

                        0FFC 00000 DBG$NPARSE_DEL_KEY:
                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11            ; 2358
                 5E             0C  C2 00002            SUBL2   #12, SP                 ;
                                7E  7C 00005            CLRQ    DEFINE_KIND             ; 2405
                 0D 00000000G   00  E8 00007            BLBS    DBG$GB_KEYPAD_INPUT, 1$ ; 2448
                    00028D18    8F  DD 0000E            PUSHL   #167192                 ; 2450
00000000G        00             01  FB 00014            CALLS   #1, LIB$SIGNAL          ;
                 59         08  AC  D0 0001B 1$:       MOVL    VERB_NODE, R9           ; 2455
                01          A9  03  90 0001F            MOVB    #3, T(R9)               ;
                0C          AE  08  A9  9E 00023        MOVAB   8(R9), 12(SP)           ; 2456
                                0C  BE  D4 00028        CLRL    @12(SP)                 ;
                                    03  DD 0002B        PUSHL   #3                      ; 2461
00000000G        00             01  FB 0002D            CALLS   #1, DBG$GET_TEMPMEM     ;
                 52             50  D0 00034            MOVL    R0, NEW_ADVERB_NODE     ;
                04          A9  52  D0 00037            MOVL    NEW_ADVERB_NODE, 4(R9)  ; 2463
                 54             52  D0 0003B            MOVL    NEW_ADVERB_NODE, ADVERB_NODE ; 2464
                                64  94 0003E            CLRB    (ADVERB_NODE)           ; 2466
                            04  A4  7C 00040            CLRQ    4(ADVERB_NODE)          ; 2467
                 6E             01  D0 00043            MOVL    #1, DEFINE_KIND         ; 2470
                 02             6E  D1 00046 2$:        CMPL    DEFINE_KIND, #2         ; 2471
                                61  14 00049            BGTR    5$                      ;
                                    03  DD 0004B        PUSHL   #3                      ; 2473
00000000G        00             01  FB 0004D            CALLS   #1, DBG$GET_TEMPMEM     ;
                 52             50  D0 00054            MOVL    R0, NEW_ADVERB_NODE     ;
                08          A4  52  D0 00057            MOVL    NEW_ADVERB_NODE, 8(ADVERB_NODE) ; 2474
                 54             52  D0 0005B            MOVL    NEW_ADVERB_NODE, ADVERB_NODE ; 2475
                 64             6E  90 0005E            MOVB    DEFINE_KIND, (ADVERB_NODE) ; 2477
                 02             6E  D1 00061            CMPL    DEFINE_KIND, #2         ; 2478
                                3F  12 00064            BNEQ    3$                      ;
                                    02  DD 00066        PUSHL   #2                      ; 2485
00000000G        00             01  FB 00068            CALLS   #1, DBG$GET_TEMPMEM     ;
                 53             50  D0 0006F            MOVL    R0, TEMP_KEY_DESC       ;
                 63 020E0000        8F  D0 00072        MOVL    #34471936, (TEMP_KEY_DESC) ; 2486
                                04  A3  D4 00079        CLRL    4(TEMP_KEY_DESC)        ; 2489
                                    53  DD 0007C        PUSHL   TEMP_KEY_DESC           ; 2493
                                7E  D4 0007E            CLRL    -(SP)                   ;
                        00000000G   00  9F 00080        PUSHAB  DBG$GL_KEY_TABLE_ID     ;
00000000G        00             03  FB 00086            CALLS   #3, SMG$SET_DEFAULT_STATE ;
                                    02  DD 0008D        PUSHL   #2                      ; 2495
00000000G        00             01  FB 0008F            CALLS   #1, DBG$GET_TEMPMEM     ;
                 55             50  D0 00096            MOVL    R0, STATE_NAME_NODE     ;
                 65             53  D0 00099            MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE) ; 2496
                            04  A5  D4 0009C            CLRL    4(STATE_NAME_NODE)      ; 2497
                04          A4  55  D0 0009F            MOVL    STATE_NAME_NODE, 4(ADVERB_NODE) ; 2498
```

```
                                03  11 000A3          BRB     4$                                      ; 2478
                            04  A4  D4 000A5 3$:      CLRL    4(ADVERB_NODE)                          ; 2503
                                6E  D6 000A8 4$:      INCL    DEFINE_KIND                             ; 2505
                                9A  11 000AA          BRB     2$                                      ; 2471
                            08  A4  D4 000AC 5$:      CLRL    8(ADVERB_NODE)                          ; 2507
                        52      04  AC  D0 000AF      MOVL    INPUT_DESC, R2                          ; 2509
                                01  DD 000B3 6$:      PUSHL   #1                                      ;
                    00000000'       EF  9F 000B5      PUSHAB  DBG$CS_CR                               ;
                                52  DD 000BB          PUSHL   R2                                      ;
00000000G   00                      03  FB 000BD      CALLS   #3, DBG$NMATCH                          ;
                03                  50  E9 000C4      BLBC    R0, 8$                                  ;
                                0281    31 000C7 7$:      BRW     36$                                 ;
                                62  B5 000CA 8$:      TSTW    (R2)                                    ; 2510
                                F9  13 000CC          BEQL    7$                                      ;
                                01  DD 000CE          PUSHL   #1                                      ; 2513
                    00000000'       EF  9F 000D0      PUSHAB  DBG$CS_SLASH                            ;
                                52  DD 000D6          PUSHL   R2                                      ;
00000000G   00                      03  FB 000D8      CALLS   #3, DBG$NMATCH                          ;
                03                  50  E8 000DF      BLBS    R0, 9$                                  ;
                                01FD    31 000E2      BRW     32$                                     ;
                                57  D4 000E5 9$:      CLRL    DEFINE_KEY_VALUE                        ; 2521
                                5A  D4 000E7          CLRL    D_KEY_NO                                ; 2522
                        50      04  A2  9E 000E9      MOVAB   4(R2), R0                               ; 2526
                00      B0  00000000'   EF  91 000ED 10$:      CMPB    P.ACC, @0(R0)                   ;
                                06  12 000F5          BNEQ    11$                                     ;
                                60  D6 000F7          INCL    (R0)                                    ; 2528
                                62  B7 000F9          DECW    (R2)                                    ; 2529
                                F0  11 000FB          BRB     10$                                     ; 2526
                00      B0  00000000'   EF  B1 000FD 11$:      CMPW    DBG$CS_NO, @0(R0)               ; 2532
                                09  12 00105          BNEQ    12$                                     ;
                            5A      01  D0 00107      MOVL    #1, D_KEY_NO                            ; 2535
                            60      02  C0 0010A      ADDL2   #2, (R0)                                ; 2536
                            62      02  A2 0010D      SUBW2   #2, (R2)                                ; 2537
                                01  DD 00110 12$:      PUSHL   #1                                      ; 2545
                    00000000'       EF  9F 00112      PUSHAB  DBG$CS_ALL                              ;
                                52  DD 00118          PUSHL   R2                                      ;
00000000G   00                      03  FB 0011A      CALLS   #3, DBG$NMATCH                          ;
                01                  50  D1 00121      CMPL    R0, #1                                  ;
                                21  12 00124          BNEQ    14$                                     ;
                                6E  D4 00126          CLRL    DEFINE_KIND                             ; 2547
                            57      01  D0 00128      MOVL    #1, DEFINE_KEY_VALUE                    ; 2548
                        04  AE      01  D0 0012B      MOVL    #1, ALL_FLAG                            ; 2549
                            60      5A  E8 0012F      BLBS    D_KEY_NO, 16$                           ; 2550
                                0C  BE  D5 00132      TSTL    @12(SP)                                 ; 2553
                                0D  13 00135          BEQL    13$                                     ;
                    00028158        8F  DD 00137      PUSHL   #164184                                 ; 2555
00000000G   00                      01  FB 0013D      CALLS   #1, LIB$SIGNAL                          ;
                                0180    31 00144 13$:      BRW     29$                                 ; 2542
                                01  DD 00147 14$:      PUSHL   #1                                      ; 2558
                    00000000'       EF  9F 00149      PUSHAB  DBG$CS_STATE                            ;
                                52  DD 0014F          PUSHL   R2                                      ;
00000000G   00                      03  FB 00151      CALLS   #3, DBG$NMATCH                          ;
                01                  50  D1 00158      CMPL    R0, #1                                  ;
                                03  13 0015B          BEQL    15$                                     ;
                                0146    31 0015D      BRW     28$                                     ;
                                57  D4 00160 15$:      CLRL    DEFINE_KEY_VALUE                        ; 2560
                        6E      02  D0 00162      MOVL    #2, DEFINE_KIND                             ; 2561
```

```
                DC              5A   E8 00165          BLBS    D_KEY_NO, 13$                                   ; 2562
                                02   DD 00168          PUSHL   #2                                              ; 2565
00000000G       00              01   FB 0016A          CALLS   #1, DBG$GET_TEMPMEM                             ;
                53              50   D0 00171          MOVL    R0, TEMP_KEY_DESC                               ;
                63 020E0000     8F   D0 00174          MOVL    #34471936, (TEMP_KEY_DESC)                      ; 2566
                         04     A3   D4 0017B          CLRL    4(TEMP_KEY_DESC)                                ; 2569
                                01   DD 0017E          PUSHL   #1                                              ; 2574
                00000000'       EF   9F 00180          PUSHAB  DBG$CS_EQUAL                                    ;
                                52   DD 00186          PUSHL   R2                                              ;
00000000G       00              03   FB 00188          CALLS   #3, DBG$NMATCH                                  ;
                18              50   E8 0018F          BLBS    R0, 18$                                         ;
                             00C8    31 00192  16$:     BRW     23$                                             ; 2575
                                52   DD 00195  17$:     PUSHL   R2                                              ;
00000000G       00              01   FB 00197          CALLS   #1, DBG$NNEXT_WORD                              ;
                                50   DD 0019E          PUSHL   R0                                              ;
00000000G       00              01   FB 001A0          CALLS   #1, DBG$NSYNTAX_ERROR                           ;
                             01B7    31 001A7          BRW     38$                                             ;
                                01   DD 001AA  18$:     PUSHL   #1                                              ; 2581
                00000000'       EF   9F 001AC          PUSHAB  DBG$CS_LEFT_PAREN                               ;
                                52   DD 001B2          PUSHL   R2                                              ;
00000000G       00              03   FB 001B4          CALLS   #3, DBG$NMATCH                                  ;
                03              50   E8 001BB          BLBS    R0, 19$                                         ;
                             00B6    31 001BE          BRW     25$                                             ;
                         0C     AC   DD 001C1  19$:     PUSHL   MESSAGE_VECT                                    ; 2589
                                0C   BB 001C4          PUSHR   #^M<R2,R3>                                      ; 2587
          0000V CF              03   FB 001C6          CALLS   #3, DBG$READ_KEY_INFO                           ;
                5B              50   D0 001CB          MOVL    R0, STATUS                                      ;
                55              5B   E9 001CE          BLBC    STATUS, 21$                                     ; 2590
                                02   DD 001D1          PUSHL   #2                                              ; 2594
00000000G       00              01   FB 001D3          CALLS   #1, DBG$GET_TEMPMEM                             ;
             08 AE              50   D0 001DA          MOVL    R0, NEW_STATE_NAME_NODE                         ;
                55         08   AE   D0 001DE          MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE            ; 2595
                65              53   D0 001E2          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)                ; 2596
                         04     A5   D4 001E5          CLRL    4(STATE_NAME_NODE)                              ; 2597
                57              55   D0 001E8          MOVL    STATE_NAME_NODE, DEFINE_KEY_VALUE               ; 2598
                58         04   A5   9E 001EB          MOVAB   4(R5), R8                                       ; 2618
                                01   DD 001EF  20$:     PUSHL   #1                                              ; 2600
                00000000'       EF   9F 001F1          PUSHAB  DBG$CS_COMMA                                    ;
                                52   DD 001F7          PUSHL   R2                                              ;
00000000G       00              03   FB 001F9          CALLS   #3, DBG$NMATCH                                  ;
                46              50   E9 00200          BLBC    R0, 22$                                         ;
                                02   DD 00203          PUSHL   #2                                              ; 2602
00000000G       00              01   FB 00205          CALLS   #1, DBG$GET_TEMPMEM                             ;
                53              50   D0 0020C          MOVL    R0, TEMP_KEY_DESC                               ;
                63 020E0000     8F   D0 0020F          MOVL    #34471936, (TEMP_KEY_DESC)                      ; 2603
                         04     A3   D4 00216          CLRL    4(TEMP_KEY_DESC)                                ; 2606
                         0C     AC   DD 00219          PUSHL   MESSAGE_VECT                                    ; 2612
                                0C   BB 0021C          PUSHR   #^M<R2,R3>                                      ; 2610
          0000V CF              03   FB 0021E          CALLS   #3, DBG$READ_KEY_INFO                           ;
                5B              50   D0 00223          MOVL    R0, STATUS                                      ;
                5B              5B   E9 00226  21$:     BLBC    STATUS, 26$                                     ; 2613
                                02   DD 00229          PUSHL   #2                                              ; 2617
00000000G       00              01   FB 0022B          CALLS   #1, DBG$GET_TEMPMEM                             ;
             08 AE              50   D0 00232          MOVL    R0, NEW_STATE_NAME_NODE                         ;
                68              08   AE   D0 00236     MOVL    NEW_STATE_NAME_NODE, (R8)                       ; 2618
                55         08   AE   D0 0023A          MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE            ; 2619
                65              53   D0 0023E          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)                ; 2620
```

```
                58        04   A5  9E 00241          MOVAB   4(R5), R8                                    ;2621
                          68   D4  00245              CLRL    (R8)
                          A6   11  00247              BRB     20$                                         ;2600
                          01   DD  00249 22$:         PUSHL   #1                                          ;2628
                00000000'      EF  9F  0024B          PUSHAB  DBG$CS_RIGHT_PAREN
                          52   DD  00251              PUSHL   R2
00000000G       00             03   FB  00253         CALLS   #3, DBG$NMATCH
                6A             50   E8  00255         BLBS    R0, 29$
                          01   DD  00258 23$:         PUSHL   #1                                          ;2629
                00000000'      EF  9F  0025A          PUSHAB  DBG$CS_CR
                          52   DD  00260              PUSHL   R2
00000000G       00             03   FB  00262         CALLS   #3, DBG$NMATCH
                03             50   E9  00265         BLBC    R0, 24$
                         00E0  31   00268              BRW     37$
                         FF1E  31   0026B 24$:         BRW     17$
                      0C   AC   DD  0026E 25$:         PUSHL   MESSAGE_VECT                                ;2642
                           0C   BB  00271              PUSHR   #^M<R2,R3>                                  ;2640
          0000V CF             03   FB  00273         CALLS   #3, DBG$READ_KEY_INFO
                5B             50   D0  00278         MOVL    R0, STATUS
                03             5B   E8  0027B 26$:     BLBS    STATUS, 27$                                 ;2643
                         00DB  31   0027E              BRW     39$
                           02   DD  00281 27$:         PUSHL   #2                                          ;2647
00000000G       00             01   FB  00283         CALLS   #1, DBG$GET_TEMPMEM
              08   AE          50   D0  00287         MOVL    R0, NEW_STATE_NAME_NODE
                55        08   AE  D0  0028B          MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE        ;2648
                65             53   D0  00290         MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)            ;2649
                          04   A5  D4  00294          CLRL    4(STATE_NAME_NODE)                          ;2650
                57             55   D0  00296         MOVL    STATE_NAME_NODE, DEFINE_KEY_VALUE           ;2651
                               21   11  00299         BRB     29$                                         ;2562
                           03   DD  0029B 28$:         PUSHL   #3                                          ;2658
                00000000'      EF  9F  0029D          PUSHAB  DBG$CS_LOG
                          52   DD  002A3              PUSHL   R2
          00000000G       00             03   FB  002A5         CALLS   #3, DBG$NMATCH
                          01             50   D1  002AC         CMPL    R0, #1
                                         78   12  002AF         BNEQ    35$
                                         57   D4  002B1         CLRL    DEFINE_KEY_VALUE                    ;2660
                          6E             01   D0  002B3         MOVL    #1, DEFINE_KIND                     ;2661
                          03             5A   E9  002B6         BLBC    D_KEY_NO, 29$                       ;2662
                          57             01   D0  002B9         MOVL    #1, DEFINE_KEY_VALUE
                                         57   D5  002BC 29$:     TSTL    DEFINE_KEY_VALUE                    ;2671
                                         66   13  002BE         BEQL    34$
                          54        04   A9  D0  002C0         MOVL    4(R9), ADVERB_NODE                  ;2674
          6E        64             08         00  ED  002C4 30$:     CMPZV   #0, #8, (ADVERB_NODE), DEFINE_KIND  ;2675
                                               06  13  002C9         BEQL    31$
                          54        08   A4  D0  002CB         MOVL    8(ADVERB_NODE), ADVERB_NODE         ;2676
                                               F3  11  002CF         BRB     30$
                    04    A4                   57  D0  002D1 31$:     MOVL    DEFINE_KEY_VALUE, 4(ADVERB_NODE)    ;2677
                                               4F  11  002D5         BRB     34$                         ;2513
                           0C   BE  D5  002D7 32$:     TSTL    @12(SP)                                     ;2686
                                         4D  12  002DA         BNEQ    35$
                          49        04   AE  E8  002DC         BLBS    ALL_FLAG, 35$
                                         02  DD  002E0         PUSHL   #2                                  ;2693
          00000000G       00             01   FB  002E2         CALLS   #1, DBG$GET_TEMPMEM
                          53             50   D0  002E8         MOVL    R0, TEMP_KEY_DESC
                          63 020E0000    8F   D0  002EB         MOVL    #34471936, (TEMP_KEY_DESC)          ;2694
                                    04   A3  D4  002F2         CLRL    4(TEMP_KEY_DESC)                    ;2697
                                    0C   AC  DD  002F4         PUSHL   MESSAGE_VECT                        ;2700
```

```
                                 0C  BB 00304          PUSHR   #^M<R2,R3>                        ; 2698
                      0000V      CF      03  FB 00306          CALLS   #3, DBG$READ_KEY_INFO
                                 5B      50  D0 0030B          MOVL    R0, STATUS
                                 04      5B  E8 0030E          BLBS    STATUS, 33$                 ; 2701
                                 50      5B  D0 00311          MOVL    STATUS, R0                  ; 2703
                                             04 00314          RET
                                 04      DD 00315 33$:         PUSHL   #4                          ; 2708
                  00000000G      00      01  FB 00317          CALLS   #1, DBG$GET_TEMPMEM
                         10      AE      50  D0 0031E          MOVL    R0, NEW_NOUN_NODE
                         0C      BE  10  AE  D0 00322          MOVL    NEW_NOUN_NODE, @12(SP)      ; 2710
                                 56  10  AE  D0 00327          MOVL    NEW_NOUN_NODE, NOUN_NODE    ; 2711
                                 66      53  D0 0032B          MOVL    TEMP_KEY_DESC, (NOUN_NODE)  ; 2712
                                     04  A6  7C 0032E          CLRQ    4(NOUN_NODE)                ; 2713
                                   FD7F  31 00331 34$:         BRW     6$                          ; 2686
                                         01  DD 00334 35$:     PUSHL   #1                          ; 2717
                              00000000'  EF  9F 00336          PUSHAB  DBG$CS_CR
                                         52  DD 0033C          PUSHL   R2
                  00000000G      00      03  FB 0033E          CALLS   #3, DBG$NMATCH
                                 0C      50  E8 00345          BLBS    R0, 37$
                                   FE4A  31 00348          BRW     17$
                                     0C  BE  D5 0034B 36$:     TSTL    @12(SP)                     ; 2726
                                         19  12 0034E          BNEQ    40$
                                 15      04  AE  E8 00350      BLBS    ALL_FLAG, 40$
                              000280D0   8F  DD 00354 37$:     PUSHL   #164048                     ; 2729
                  00000000G      00          01  FB 0035A     CALLS   #1, DBG$NMAKE_ARG_VECT
                         0C      BC          50  D0 00361 38$: MOVL    R0, @MESSAGE_VECT
                                 50          04  D0 00365 39$: MOVL    #4, R0                      ; 2730
                                             04 00368          RET
                                 50          01  D0 00369 40$: MOVL    #1, R0                      ; 2733
                                             04 0036C          RET                                 ; 2734

; Routine Size:  877 bytes,    Routine Base:  DBG$CODE + 0D3B

; 2616          2735  1
```

```
;  2618          2736  1 GLOBAL ROUTINE dbg$nexecute_define (verb_node, message_vect) =
;  2619          2737  1 !++
;  2620          2738  1 ! Functional Description
;  2621          2739  1 !
;  2622          2740  1 !       This routine performs the action associated with the DEFINE command.
;  2623          2741  1 !
;  2624          2742  1 ! Routine Inputs
;  2625          2743  1 !
;  2626          2744  1 !       verb_node -     The head of a command execution tree. This is built
;  2627          2745  1 !                       by the routine DBG$NPARSE_DEFINE or by the routine
;  2628          2746  1 !                       DBG$NPARSE_DEF_KEY in the case of a DEFINE/KEY command,
;  2629          2747  1 !                       its structure is described in the header of the
;  2630          2748  1 !                       routine.
;  2631          2749  1 !       message_vect -  An error message vector.
;  2632          2750  1 !
;  2633          2751  1 ! Routine Outputs
;  2634          2752  1 !
;  2635          2753  1 !       New entries may be made to the DEFINE symbol table. (See DBGLIB.REQ
;  2636          2754  1 !       for documentation of the structure of the DEFINE symbol table.)
;  2637          2755  1 !       Or, new entries are made to the key-definition table for the DEFINE/KEY
;  2638          2756  1 !       command.  This table is external to the debugger structures.
;  2639          2757  1 !
;  2640          2758  1 !       The routine value is one of:
;  2641          2759  1 !       sts$k_success - Success code.
;  2642          2760  1 !       sts$k_severe -  Error. An error message vector is contructed.
;  2643          2761  1 !--
;  2644          2762  2     BEGIN
;  2645          2763  2
;  2646          2764  2     MAP
;  2647          2765  2         verb_node : REF dbg$verb_node;
;  2648          2766  2
;  2649          2767  2     LOCAL
;  2650          2768  2         first_time,                                     ! TRUE first time in loop
;  2651          2769  2         global_flag,                                    ! TRUE for symbols defined with ==
;  2652          2770  2         noun_node       : REF dbg$noun_node,            ! Points to a noun node
;  2653          2771  2         adverb_node     : REF dbg$adverb_node,          ! Points to an adverb node
;  2654          2772  2         replaced_flag;                                  ! TRUE if definition already existed
;  2655          2773  2
;  2656          2774  2     ! Check for DEFINE/KEY.
;  2657          2775  2     !
;  2658          2776  2
;  2659          2777  2     IF .verb_node[dbg$b_verb_composite] EQL define_key
;  2660          2778  2     THEN
;  2661          2779  3         BEGIN
;  2662          2780  3
;  2663          2781  3         LITERAL
;  2664          2782  3             v_key_noecho        = 0,
;  2665          2783  3             v_key_terminate     = 1,
;  2666          2784  3             v_key_lock          = 2;
;  2667          2785  3
;  2668          2786  3         LOCAL
;  2669          2787  3             attributes                  : BITVECTOR [32],
;  2670          2788  3             if_state_desc_address       : REF dbg$state_name_node,
;  2671          2789  3             set_state_desc_address      : REF dbg$state_name_node,
;  2672          2790  3             output_log,
;  2673          2791  3             add_status,
;  2674          2792  3             desc_ptr                    : REF dbg$stg_desc;
```

```
;  2675          2793  3
;  2676          2794  3          !+
;  2677          2795  3          !       We will set up the noun and verb pointers and proceed to walk
;  2678          2796  3          ! down the adverb list with the knowledge that the qualifier information
;  2679          2797  3          ! is in order.  After checking the qualifiers, a call is made to the
;  2680          2798  3          ! routine SMG$ADD_KEY_DEF to execute the command; if there is more than
;  2681          2799  3          ! one state noted in the if_state qualifier a call is made to
;  2682          2800  3          ! SMG$ADD_KEY_DEF to load each possible key definition.
;  2683          2801  3          !       Then exit successfully, unless some error was found on the way.
;  2684          2802  3          !-
;  2685          2803  3
;  2686          2804  3          noun_node   = .verb_node [dbg$l_verb_object_ptr];
;  2687          2805  3          adverb_node = .verb_node [dbg$l_verb_adverb_ptr];
;  2688          2806  3          attributes  = 0;
;  2689          2807  3
;  2690          2808  3          ! ECHO qualifier
;  2691          2809  3
;  2692          2810  3          attributes [v_key_noecho] = .adverb_node [dbg$l_adverb_value];
;  2693          2811  3          adverb_node = .adverb_node [dbg$l_adverb_link];
;  2694          2812  3
;  2695          2813  3          ! IF_STATE qualifier
;  2696          2814  3
;  2697          2815  3          if_state_desc_address = .adverb_node [dbg$l_adverb_value];
;  2698          2816  3          adverb_node = .adverb_node [dbg$l_adverb_link];
;  2699          2817  3
;  2700          2818  3          ! LOCK_STATE qualifier
;  2701          2819  3
;  2702          2820  3          attributes [v_key_lock] = .adverb_node [dbg$l_adverb_value];
;  2703          2821  3          adverb_node = .adverb_node [dbg$l_adverb_link];
;  2704          2822  3
;  2705          2823  3          ! LOG qualifier
;  2706          2824  3
;  2707          2825  3          output_log = NOT .adverb_node [dbg$l_adverb_value];
;  2708          2826  3          adverb_node = .adverb_node [dbg$l_adverb_link];
;  2709          2827  3
;  2710          2828  3          ! SET_STATE qualifier
;  2711          2829  3
;  2712          2830  3          set_state_desc_address = .adverb_node [dbg$l_adverb_value];
;  2713          2831  4          IF (.set_state_desc_address EQL 0) AND (.attributes [v_key_lock])
;  2714          2832  3          THEN
;  2715          2833  3              SIGNAL(dbg$_conflict);
;  2716          2834  3          adverb_node = .adverb_node [dbg$l_adverb_link];
;  2717          2835  3
;  2718          2836  3          ! TERMINATE qualifier
;  2719          2837  3
;  2720          2838  3          attributes [v_key_terminate] = .adverb_node [dbg$l_adverb_value];
;  2721          2839  4          IF (.attributes [v_key_noecho]) AND (NOT .attributes [v_key_terminate])
;  2722          2840  3          THEN
;  2723          2841  3              SIGNAL(dbg$_conflict);
;  2724          2842  3
;  2725          2843  3          ! Execute the define-key command by calling smg$add_key_def.
;  2726          2844  3          ! Loop until no more if_state names exist.
;  2727          2845  3
;  2728          2846  3          WHILE TRUE DO
;  2729          2847  4              BEGIN
;  2730          2848  4              ! Add the key definition
;  2731          2849  4
```

```
;  2732     2850  4              add_status = smg$add_key_def (dbg$gl_key_table_id,
;  2733     2851  4                                           .noun_node [dbg$l_noun_value],
;  2734     2852  4                                           .if_state_desc_address [dbg$l_state_name_ptr],
;  2735     2853  4                                           attributes,
;  2736     2854  4                                           .noun_node [dbg$l_adjective_ptr],
;  2737     2855  4                                           .set_state_desc_address);
;  2738     2856  4              IF NOT .add_status
;  2739     2857  4              THEN
;  2740     2858  4                  SIGNAL(dbg$_defkeyerr);
;  2741     2859  4              IF .output_log
;  2742     2860  4              THEN
;  2743     2861  4                  SIGNAL(dbg$_defkey, 2,
;  2744     2862  4                         .if_state_desc_address [dbg$l_state_name_ptr],
;  2745     2863  4                         .noun_node [dbg$l_noun_value]);
;  2746     2864  4
;  2747     2865  4              if_state_desc_address = .if_state_desc_address [dbg$l_state_name_link];
;  2748     2866  4              IF .if_state_desc_address EQL 0
;  2749     2867  4              THEN
;  2750     2868  4                  EXITLOOP;
;  2751     2869  3              END;
;  2752     2870  3
;  2753     2871  3          RETURN sts$k_success;
;  2754     2872  2          END;
;  2755     2873  2
;  2756     2874  2
;  2757     2875  2      ! Loop through the DEFINE list.
;  2758     2876  2      !
;  2759     2877  2      first_time = TRUE;
;  2760     2878  2      WHILE TRUE DO
;  2761     2879  3          BEGIN
;  2762     2880  3
;  2763     2881  3          ! For the first time around the loop, recover the noun node
;  2764     2882  3          ! by following the pointer in the verb node.
;  2765     2883  3          !
;  2766     2884  3          IF .first_time
;  2767     2885  3          THEN
;  2768     2886  4              BEGIN
;  2769     2887  4              noun_node = .verb_node [dbg$l_verb_object_ptr];
;  2770     2888  4
;  2771     2889  4              ! Set first_time to false for future times around the loop.
;  2772     2890  4              !
;  2773     2891  4              first_time = FALSE;
;  2774     2892  4              END
;  2775     2893  4
;  2776     2894  4          ! For subsequent times around the loop, get the next noun node
;  2777     2895  4          ! from the link in the current noun node.
;  2778     2896  4          ! Exit the loop when that link is zero.
;  2779     2897  4          !
;  2780     2898  3          ELSE
;  2781     2899  4              BEGIN
;  2782     2900  4              noun_node = .noun_node [dbg$l_noun_link];
;  2783     2901  4              IF .noun_node EQL 0
;  2784     2902  4              THEN
;  2785     2903  4                  EXITLOOP;
;  2786     2904  3              END;
;  2787     2905  3
;  2788     2906  3          ! Now determine whether the define was local or global.
```

```
;  2789          2907  3          !
;  2790          2908  3          IF .noun_node [dbg$l_adjective_ptr] EQL define_global
;  2791          2909  3          THEN
;  2792          2910  3              global_flag = TRUE
;  2793          2911  3          ELSE
;  2794          2912  3              global_flag = FALSE;
;  2795          2913  3
;  2796          2914  3          ! We just call the routine DBG$DEF_SYM_ADD to perform the action.
;  2797          2915  3          !
;  2798          2916  3          IF NOT dbg$def_sym_add (.noun_node [dbg$l_noun_value],
;  2799          2917  3                                  .verb_node [dbg$b_verb_composite],
;  2800          2918  3                                  .noun_node [dbg$l_noun_value2],
;  2801          2919  3                                  .global_flag,
;  2802          2920  3                                  replaced_flag,
;  2803          2921  3                                  .message_vect)
;  2804          2922  3          THEN
;  2805          2923  3              RETURN sts$k_severe;
;  2806          2924  3          END; ! While loop
;  2807          2925  3
;  2808          2926  2      RETURN sts$k_success;
;  2809          2927  1      END; ! dbg$nexecute_define


                                01FC 00000          .ENTRY  DBG$NEXECUTE_DEFINE, Save R2,R3,R4,R5,R6,-  ; 2736
                                                            R7,R8                                       ;
                    58 00000000G   00   9E 00002    MOVAB   LIB$SIGNAL, R8                              ;
                    5E             08   C2 00009    SUBL2   #8, SP                                      ;
                    55       04    AC   D0 0000C    MOVL    VERB_NODE, R5                               ; 2777
                    07       01    A5   91 00010    CMPB    1(R5), #7                                   ;
                                   03   13 00014    BEQL    1$                                          ;
                                 0099   31 00016    BRW     6$                                          ;
                    52       04    A5   7D 00019 1$: MOVQ    4(R5), ADVERB_NODE                          ; 2805
                                   6E   D4 0001D    CLRL    ATTRIBUTES                                  ; 2806
                6E       01        00   04 A2 F0 0001F    INSV    4(ADVERB_NODE), #0, #1, ATTRIBUTES      ; 2810
                    52       08    A2   D0 00025    MOVL    8(ADVERB_NODE), ADVERB_NODE                 ; 2811
                    54       04    A2   D0 00029    MOVL    4(ADVERB_NODE), IF_STATE_DESC_ADDRESS       ; 2815
                    52       08    A2   D0 0002D    MOVL    8(ADVERB_NODE), ADVERB_NODE                 ; 2816
                6E       01        02   04 A2 F0 00031    INSV    4(ADVERB_NODE), #2, #1, ATTRIBUTES      ; 2820
                    52       08    A2   D0 00037    MOVL    8(ADVERB_NODE), ADVERB_NODE                 ; 2821
                    57       04    A2   D2 0003B    MCOML   4(ADVERB_NODE), OUTPUT_LOG                  ; 2825
                    52       08    A2   D0 0003F    MOVL    8(ADVERB_NODE), ADVERB_NODE                 ; 2826
                    56       04    A2   D0 00043    MOVL    4(ADVERB_NODE), SET_STATE_DESC_ADDRESS      ; 2830
                                   0D   12 00047    BNEQ    2$                                          ; 2831
                             09    6E   02 E1 00049    BBC     #2, ATTRIBUTES, 2$                      ;
                          00028158   8F   DD 0004D    PUSHL   #164184                                 ; 2833
                                68   01   FB 00053    CALLS   #1, LIB$SIGNAL                          ;
                    52       08    A2   D0 00056 2$: MOVL    8(ADVERB_NODE), ADVERB_NODE                 ; 2834
                6E       01        01   04 A2 F0 0005A    INSV    4(ADVERB_NODE), #1, #1, ATTRIBUTES      ; 2838
                                   0D   6E E9 00060    BLBC    ATTRIBUTES, 3$                          ; 2839
                             09    6E   01 E0 00063    BBS     #1, ATTRIBUTES, 3$                      ;
                          00028158   8F   DD 00067    PUSHL   #164184                                 ; 2841
                                68   01   FB 0006D    CALLS   #1, LIB$SIGNAL                          ;
                                   56   DD 00070 3$: PUSHL   SET_STATE_DESC_ADDRESS                  ; 2855
                    04       04    A3   DD 00072    PUSHL   4(NOUN_NODE)                            ; 2854
```

```
                        08    AE  9F 00075         PUSHAB  ATTRIBUTES                              ; 2850
                              64  DD 00078         PUSHL   (IF_STATE_DESC_ADDRESS)                 ; 2852
                              63  DD 0007A         PUSHL   (NOUN_NODE)                             ; 2851
               00000000G      00  9F 0007C         PUSHAB  DBG$GL_KEY_TABLE_ID                     ; 2850
   00000000G   00             06  FB 00082         CALLS   #6, SMG$ADD_KEY_DEF                     ;
               52             50  D0 00089         MOVL    R0, ADD_STATUS                          ;
               09             52  E8 0008C         BLBS    ADD_STATUS, 4$                          ; 2856
               00028110       8F  DD 0008F         PUSHL   #164112                                 ; 2858
               68             01  FB 00095         CALLS   #1, LIB$SIGNAL                          ;
               0F             57  E9 00098  4$:    BLBC    OUTPUT_LOG, 5$                          ; 2859
                              63  DD 0009B         PUSHL   (NOUN_NODE)                             ; 2863
                              64  DD 0009D         PUSHL   (IF_STATE_DESC_ADDRESS)                 ; 2862
                              02  DD 0009F         PUSHL   #2                                      ; 2861
               000280B3       8F  DD 000A1         PUSHL   #164019                                 ;
               68             04  FB 000A7         CALLS   #4, LIB$SIGNAL                          ;
               54       04    A4  D0 000AA  5$:    MOVL    4(IF_STATE_DESC_ADDRESS), -             ; 2865
                                                           IF_STATE_DESC_ADDRESS                   ;
                              C0  12 000AE         BNEQ    3$                                      ; 2866
                              3E  11 000B0         BRB     12$                                     ; 2871
               54             01  D0 000B2  6$:    MOVL    #1, FIRST_TIME                          ; 2877
               08             54  E9 000B5  7$:    BLBC    FIRST_TIME, 8$                          ; 2884
               53       08    A5  D0 000B8         MOVL    8(R5), NOUN_NODE                        ; 2887
                              54  D4 000BC         CLRL    FIRST_TIME                              ; 2891
                              06  11 000BE         BRB     9$                                      ; 2884
               53       08    A3  D0 000C0  8$:    MOVL    8(NOUN_NODE), NOUN_NODE                 ; 2900
                              2A  13 000C4         BEQL    12$                                     ; 2901
               01       04    A3  D1 000C6  9$:    CMPL    4(NOUN_NODE), #1                        ; 2908
                              05  12 000CA         BNEQ    10$                                     ;
               52             01  D0 000CC         MOVL    #1, GLOBAL_FLAG                         ; 2910
                              02  11 000CF         BRB     11$                                     ;
                              52  D4 000D1  10$:   CLRL    GLOBAL_FLAG                             ; 2912
                        08    AC  DD 000D3  11$:   PUSHL   MESSAGE_VECT                            ; 2921
                        08    AE  9F 000D6         PUSHAB  REPLACED_FLAG                           ; 2916
                              52  DD 000D9         PUSHL   GLOBAL_FLAG                             ; 2919
                        0C    A3  DD 000DB         PUSHL   12(NOUN_NODE)                           ; 2918
               7E       01    A5  9A 000DE         MOVZBL  1(R5), -(SP)                            ; 2917
                              63  DD 000E2         PUSHL   (NOUN_NODE)                             ; 2916
        EE9F   CF             06  FB 000E4         CALLS   #6, DBG$DEF_SYM_ADD                     ;
               C9             50  E8 000E9         BLBS    R0, 7$                                  ;
               50             04  D0 000EC         MOVL    #4, R0                                  ; 2923
                                  04 000EF         RET                                             ;
               50             01  D0 000F0  12$:   MOVL    #1, R0                                  ; 2926
                                  04 000F3         RET                                             ; 2927

; Routine Size:  244 bytes,    Routine Base:  DBG$CODE + 10A8
```

```
;  2811          2928  1 GLOBAL ROUTINE dbg$nexecute_delete (verb_node, message_vect) =
;  2812          2929  1 !++
;  2813          2930  1 ! Functional Description
;  2814          2931  1 !
;  2815          2932  1 !       This routine performs the action associated with the DELETE command.
;  2816          2933  1 !       DELETE is the same as UNDEFINE so we just call that routine.
;  2817          2934  1 !
;  2818          2935  1 ! Routine Inputs
;  2819          2936  1 !
;  2820          2937  1 !       verb_node -     The head of a command execution tree. This is built
;  2821          2938  1 !                       by the routine DBG$NPARSE_UNDEFINE, and its structure
;  2822          2939  1 !                       is described in the header of that routine.
;  2823          2940  1 !       message_vect -  An error message vector.
;  2824          2941  1 !
;  2825          2942  1 ! Routine Outputs
;  2826          2943  1 !
;  2827          2944  1 !       Entries may be remove from the DEFINE symbol table. (See DBGLIB.REQ
;  2828          2945  1 !       for documentation of the structure of the DEFINE symbol table.
;  2829          2946  1 !
;  2830          2947  1 !       The routine value is one of:
;  2831          2948  1 !       sts$k_success - Success code.
;  2832          2949  1 !       sts$k_severe -  Error. An error message vector is contructed.
;  2833          2950  1 !--
;  2834          2951  2     BEGIN
;  2835          2952  2     RETURN dbg$nexecute_undefine(.verb_node, .message_vect);
;  2836          2953  1     END;


                                0000 00000          .ENTRY  DBG$NEXECUTE_DELETE, Save nothing       ; 2928
                          7E    04   AC  7D 00002    MOVQ    VERB_NODE, -(SP)                        ; 2952
                    0000V CF          02  FB 00006    CALLS   #2, DBG$NEXECUTE_UNDEFINE               ;
                                      04  0000B      RET                                             ; 2953

; Routine Size:  12 bytes,    Routine Base:  DBG$CODE + 119C
```

```
;  2838     2954  1 GLOBAL ROUTINE dbg$nexecute_undefine (verb_node, message_vect) =
;  2839     2955  1 !++
;  2840     2956  1 ! Functional Description
;  2841     2957  1 !
;  2842     2958  1 !       This routine performs the action associated with the UNDEFINE command.
;  2843     2959  1 !
;  2844     2960  1 ! Routine Inputs
;  2845     2961  1 !
;  2846     2962  1 !       verb_node -     The head of a command execution tree. This is built
;  2847     2963  1 !                       by the routine DBG$NPARSE_UNDEFINE, and its structure
;  2848     2964  1 !                       is described in the header of that routine.
;  2849     2965  1 !       message_vect -  An error message vector.
;  2850     2966  1 !
;  2851     2967  1 ! Routine Outputs
;  2852     2968  1 !
;  2853     2969  1 !       Entries may be remove from the DEFINE symbol table. (See DBGLIB.REQ
;  2854     2970  1 !       for documentation of the structure of the DEFINE symbol table.
;  2855     2971  1 !
;  2856     2972  1 !       The routine value is one of:
;  2857     2973  1 !       sts$k_success - Success code.
;  2858     2974  1 !       sts$k_severe -  Error. An error message vector is contructed.
;  2859     2975  1 !--
;  2860     2976  2     BEGIN
;  2861     2977  2
;  2862     2978  2     MAP
;  2863     2979  2         verb_node : REF dbg$verb_node;
;  2864     2980  2
;  2865     2981  2     LOCAL
;  2866     2982  2         first_time,                             ! TRUE first time in loop
;  2867     2983  2         global_flag,                            ! TRUE for global symbols
;  2868     2984  2         noun_node       : REF dbg$noun_node,    ! Points to a noun node
;  2869     2985  2         adverb_node     : REF dbg$adverb_node,  ! Points to an adverb node
;  2870     2986  2         removed_flag;                           ! TRUE after the definition is removed
;  2871     2987  2
;  2872     2988  2     ! Check for UNDEFINE/KEY.
;  2873     2989  2     !
;  2874     2990  2     IF .verb_node [dbg$b_verb_composite] EQL undefine_key
;  2875     2991  2     THEN
;  2876     2992  2
;  2877     2993  3         BEGIN
;  2878     2994  3
;  2879     2995  3         LOCAL
;  2880     2996  3             all_flag,
;  2881     2997  3             context             : INITIAL(0),
;  2882     2998  3             output_log,
;  2883     2999  3             del_status,
;  2884     3000  3             temp_state_address  : REF dbg$state_name_node,
;  2885     3001  3             state_desc_address  : REF dbg$state_name_node,
;  2886     3002  3             desc_ptr            : REF dbg$stg_desc,
;  2887     3003  3             key_name_desc       : dbg$stg_desc,
;  2888     3004  3             state_name_desc     : dbg$stg_desc;
;  2889     3005  3
;  2890     3006  3             !+
;  2891     3007  3             !       We will set up the noun and verb pointers and proceed to walk
;  2892     3008  3             ! down the adverb list with the knowledge that the qualifier information
;  2893     3009  3             ! is in order.  After checking the qualifiers, a call is made to the
;  2894     3010  3             ! routine SMG$DELETE_KEY_DEF to execute the command; if the /ALL
```

```
;  2895     3011  3        ! qualifier exits then calls are made to SMG$LIST_KEY_DEFS to get all
;  2896     3012  3        ! the key definitions in the table.  A call is also made for each
;  2897     3013  3        ! state specified by the State qualifier.
;  2898     3014  3        !     Then exit successfully, unless some error was found on the way.
;  2899     3015  3        !-
;  2900     3016  3
;  2901     3017  3        ! Initialize descriptors
;  2902     3018  3        !
;  2903     3019  3
;  2904     3020  3        key_name_desc[dsc$w_length] = 0;
;  2905     3021  3        key_name_desc[dsc$b_dtype]  = dsc$k_dtype_t;
;  2906     3022  3        key_name_desc[dsc$b_class]  = dsc$k_class_d;
;  2907     3023  3        key_name_desc[dsc$a_pointer]= 0;
;  2908     3024  3
;  2909     3025  3        state_name_desc[dsc$w_length] = 0;
;  2910     3026  3        state_name_desc[dsc$b_dtype]  = dsc$k_dtype_t;
;  2911     3027  3        state_name_desc[dsc$b_class]  = dsc$k_class_d;
;  2912     3028  3        state_name_desc[dsc$a_pointer]= 0;
;  2913     3029  3
;  2914     3030  3        noun_node   = .verb_node [dbg$l_verb_object_ptr];
;  2915     3031  3        adverb_node = .verb_node [dbg$l_verb_adverb_ptr];
;  2916     3032  3
;  2917     3033  3        ! ALL qualifier
;  2918     3034  3
;  2919     3035  3        all_flag = .adverb_node [dbg$l_adverb_value];
;  2920     3036  3        adverb_node = .adverb_node [dbg$l_adverb_link];
;  2921     3037  3
;  2922     3038  3        ! LOG qualifier
;  2923     3039  3
;  2924     3040  3        output_log = NOT .adverb_node [dbg$l_adverb_value];
;  2925     3041  3        adverb_node = .adverb_node [dbg$l_adverb_link];
;  2926     3042  3
;  2927     3043  3        ! STATE qualifier
;  2928     3044  3
;  2929     3045  3        state_desc_address = .adverb_node [dbg$l_adverb_value];
;  2930     3046  3
;  2931     3047  3        ! If the /ALL qualifier exists
;  2932     3048  3        !
;  2933     3049  3
;  2934     3050  3        WHILE .all_flag DO
;  2935     3051  4            BEGIN
;  2936     3052  4            temp_state_address = .state_desc_address;
;  2937     3053  4
;  2938     3054  4            del_status = smg$list_key_defs(dbg$gl_key_table_id,
;  2939     3055  4                                           context,
;  2940     3056  4                                           key_name_desc,
;  2941     3057  4                                           state_name_desc);
;  2942     3058  4            IF NOT .del_status
;  2943     3059  4            THEN
;  2944     3060  4                IF .del_status EQL smg$_nomorekeys
;  2945     3061  4                THEN
;  2946     3062  4                    EXITLOOP
;  2947     3063  4                ELSE
;  2948     3064  4                    SIGNAL(dbg$_delkeyerr);
;  2949     3065  4
;  2950     3066  4            WHILE .temp_state_address NEQ 0 DO
;  2951     3067  5                BEGIN
```

```
;  2952     3068  5                desc_ptr = .temp_state_address [dbg$l_state_name_ptr];
;  2953     3069  5
;  2954     3070  5                ! Check to see if the state names match, if so, delete the key
;  2955     3071  5                ! Remember, str$compare_eql returns 0 for a match.
;  2956     3072  5                !
;  2957     3073  5
;  2958     3074  5                IF NOT str$compare_eql(state_name_desc, .desc_ptr)
;  2959     3075  5                THEN
;  2960     3076  6                    BEGIN
;  2961     3077  6                    del_status = smg$delete_key_def (dbg$gl_key_table_id,
;  2962     3078  6                                                     key_name_desc,
;  2963     3079  6                                                     state_name_desc);
;  2964     3080  6                    IF NOT .del_status
;  2965     3081  6                    THEN
;  2966     3082  6                        IF .del_status EQL smg$_keynotdef
;  2967     3083  6                        THEN
;  2968     3084  7                            BEGIN
;  2969     3085  7                            IF .output_log THEN
;  2970     3086  7                                SIGNAL(dbg$_undkey, 2, state_name_desc, key_name_desc);
;  2971     3087  7                            EXITLOOP;
;  2972     3088  7                            END
;  2973     3089  6                        ELSE
;  2974     3090  6                            SIGNAL(dbg$_delkeyerr);
;  2975     3091  6
;  2976     3092  7                    IF (.output_log) AND (NOT .del_status EQL smg$_keynotdef)
;  2977     3093  6                    THEN
;  2978     3094  7                        BEGIN
;  2979     3095  7                        SIGNAL(dbg$_delkey, 2, state_name_desc, key_name_desc);
;  2980     3096  7                        EXITLOOP;
;  2981     3097  6                        END;
;  2982     3098  6                    END
;  2983     3099  5                ELSE
;  2984     3100  5                    ! If the state names are not the same, look for the next one
;  2985     3101  5                    !
;  2986     3102  5
;  2987     3103  5                    temp_state_address =.temp_state_address [dbg$l_state_name_link];
;  2988     3104  4                END;
;  2989     3105  3            END;
;  2990     3106  3
;  2991     3107  3            ! If not /ALL
;  2992     3108  3            !
;  2993     3109  3
;  2994     3110  3            WHILE NOT .all_flag DO
;  2995     3111  4                BEGIN
;  2996     3112  4                ch$move(8, .noun_node [dbg$l_noun_value], key_name_desc);
;  2997     3113  4                ch$move(8, .state_desc_address [dbg$l_state_name_ptr], state_name_desc);
;  2998     3114  4
;  2999     3115  4                del_status = smg$delete_key_def (dbg$gl_key_table_id,
;  3000     3116  4                                                 key_name_desc,
;  3001     3117  4                                                 state_name_desc);
;  3002     3118  4                IF NOT .del_status
;  3003     3119  4                THEN
;  3004     3120  4                    IF .del_status EQL smg$_keynotdef
;  3005     3121  4                    THEN
;  3006     3122  5                        BEGIN
;  3007     3123  5                        IF .output_log THEN
;  3008     3124  5                            SIGNAL(dbg$_undkey, 2, state_name_desc, key_name_desc);
```

```
;  3009          3125  5                 END
;  3010          3126  4             ELSE
;  3011          3127  4                 SIGNAL(dbg$_delkeyerr);
;  3012          3128  4
;  3013          3129  5             IF (.output_log) AND (NOT .del_status EQL smg$_keynotdef)
;  3014          3130  4             THEN
;  3015          3131  4                 SIGNAL(dbg$_delkey, 2, state_name_desc, key_name_desc);
;  3016          3132  4
;  3017          3133  4             state_desc_address = .state_desc_address [dbg$l_state_name_link];
;  3018          3134  4             IF .state_desc_address EQL 0
;  3019          3135  4             THEN
;  3020          3136  4                 EXITLOOP;
;  3021          3137  3             END;
;  3022          3138  3
;  3023          3139  3         RETURN sts$k_success;
;  3024          3140  2         END;
;  3025          3141  2
;  3026          3142  2
;  3027          3143  2     ! Check for UNDEFINE/ALL
;  3028          3144  2     !
;  3029          3145  2     IF .verb_node [dbg$b_verb_composite] EQL undefine_all
;  3030          3146  2     THEN
;  3031          3147  2         RETURN dbg$def_sym_remove_all (FALSE, .message_vect);
;  3032          3148  2
;  3033          3149  2     ! Check for UNDEFINE/ALL/GLOBAL
;  3034          3150  2     !
;  3035          3151  2     IF .verb_node [dbg$b_verb_composite] EQL undefine_all_global
;  3036          3152  2     THEN
;  3037          3153  2         RETURN dbg$def_sym_remove_all (TRUE, .message_vect);
;  3038          3154  2
;  3039          3155  2     ! Loop through the DEFINE list.
;  3040          3156  2     !
;  3041          3157  2     first_time = TRUE;
;  3042          3158  2     WHILE TRUE DO
;  3043          3159  3         BEGIN
;  3044          3160  3
;  3045          3161  3         ! For the first time around the loop, recover the noun node
;  3046          3162  3         ! by following the pointer in the verb node.
;  3047          3163  3         !
;  3048          3164  3         IF .first_time
;  3049          3165  3         THEN
;  3050          3166  4             BEGIN
;  3051          3167  4             noun_node = .verb_node [dbg$l_verb_object_ptr];
;  3052          3168  4
;  3053          3169  4             ! Set first_time to false for future times around the loop.
;  3054          3170  4             !
;  3055          3171  4             first_time = FALSE;
;  3056          3172  4             END
;  3057          3173  4
;  3058          3174  4         ! For subsequent times around the loop, get the next noun node
;  3059          3175  4         ! from the link in the current noun node.
;  3060          3176  4         ! Exit the loop when that link is zero.
;  3061          3177  4         !
;  3062          3178  3         ELSE
;  3063          3179  4             BEGIN
;  3064          3180  4             noun_node = .noun_node [dbg$l_noun_link];
;  3065          3181  4             IF .noun_node EQL 0
```

```
;  3066          3182  4              THEN
;  3067          3183  4                  EXITLOOP;
;  3068          3184                   END;
;  3069          3185
;  3070          3186                ! Now determine whether the define was local or global.
;  3071          3187                !
;  3072          3188                IF .noun_node [dbg$l_adjective_ptr] EQL define_global
;  3073          3189                THEN
;  3074          3190                    global_flag = TRUE
;  3075          3191                ELSE
;  3076          3192                    global_flag = FALSE;
;  3077          3193
;  3078          3194                ! We just call the routine DBG$DEF_SYM_REMOVE to perform the action.
;  3079          3195                !
;  3080          3196                IF NOT dbg$def_sym_remove (.noun_node [dbg$l_noun_value],
;  3081          3197                                           .global_flag,
;  3082          3198                                           removed_flag,
;  3083          3199                                           .message_vect)
;  3084          3200                THEN
;  3085          3201                    RETURN sts$k_severe;
;  3086          3202
;  3087          3203                ! Signal an informational if the symbol was not defined.
;  3088          3204                !
;  3089          3205                IF NOT .removed_flag
;  3090          3206                THEN
;  3091          3207                    SIGNAL (dbg$_notdefine, 1, .noun_node [dbg$l_noun_value]);
;  3092          3208
;  3093          3209                END; ! While loop
;  3094          3210
;  3095          3211  2         RETURN sts$k_success;
;  3096          3212  1         END;
```

```
                                    0FFC 00000          .ENTRY  DBG$NEXECUTE_UNDEFINE, Save R2,R3,R4,R5,R6,-; 2954
                                                                R7,R8,R9,R10,R11                              ;
                          5E          20  C2 00002          SUBL2   #32, SP                                 ;
                          58    04    AC  D0 00005          MOVL    VERB_NODE, R8                           ; 2990
                          03    01    A8  91 00009          CMPB    1(R8), #3                               ;
                                      03  13 0000D          BEQL    1$                                      ;
                                    0176  31 0000F          BRW     16$                                     ;
                                      6E  D4 00012 1$:      CLRL    CONTEXT                                 ; 2993
                    14    AE 020E0000     8F  D0 00014          MOVL    #34471936, KEY_NAME_DESC                ; 3020
                                    18    AE  D4 0001C          CLRL    KEY_NAME_DESC+4                         ; 3023
                    08    AE 020E0000     8F  D0 0001F          MOVL    #34471936, STATE_NAME_DESC              ; 3025
                                    0C    AE  D4 00027          CLRL    STATE_NAME_DESC+4                       ; 3028
                          57    08    A8  D0 0002A          MOVL    8(R8), NOUN_NODE                        ; 3030
                          50    04    A8  D0 0002E          MOVL    4(R8), ADVERB_NODE                      ; 3031
                          52    04    A0  D0 00032          MOVL    4(ADVERB_NODE), ALL_FLAG                ; 3035
                          50    08    A0  D0 00036          MOVL    8(ADVERB_NODE), ADVERB_NODE             ; 3036
                          5A    04    A0  D2 0003A          MCOML   4(ADVERB_NODE), OUTPUT_LOG              ; 3040
                          50    08    A0  D0 0003E          MOVL    8(ADVERB_NODE), ADVERB_NODE             ; 3041
                          56    04    A0  D0 00042          MOVL    4(ADVERB_NODE), STATE_DESC_ADDRESS      ; 3045
                          03          52  E8 00046 2$:      BLBS    ALL_FLAG, 4$                            ; 3052
                                    00B8  31 00049 3$:      BRW     10$                                     ;
```

```
                53              56  D0 0004C 4$:    MOVL    STATE_DESC_ADDRESS, TEMP_STATE_ADDRESS
                          08    AE  9F 0004F        PUSHAB  STATE_NAME_DESC                         ; 3054
                          18    AE  9F 00052        PUSHAB  KEY_NAME_DESC
                          08    AE  9F 00055        PUSHAB  CONTEXT
                00000000G       00  9F 00058        PUSHAB  DBG$GL_KEY_TABLE_ID
00000000G       00              04  FB 0005E        CALLS   #4, SMG$LIST_KEY_DEFS
                59              50  D0 00065        MOVL    R0, DEL_STATUS
                16              59  E8 00068        BLBS    DEL_STATUS, 5$                          ; 3058
00000000G       8F              59  D1 0006B        CMPL    DEL_STATUS, #SMG$_NOMOREKEYS            ; 3060
                                D5  13 00072        BEQL    3$
                00028118        8F  DD 00074        PUSHL   #164120                                 ; 3064
00000000G       00              01  FB 0007A        CALLS   #1, LIB$SIGNAL
                                53  D5 00081 5$:    TSTL    TEMP_STATE_ADDRESS                      ; 3066
                                C1  13 00083        BEQL    2$
                54              63  D0 00085        MOVL    (TEMP_STATE_ADDRESS), DESC_PTR          ; 3068
                                54  DD 00088        PUSHL   DESC_PTR                                ; 3074
                          0C    AE  9F 0008A        PUSHAB  STATE_NAME_DESC
00000000G       00              02  FB 0008D        CALLS   #2, STR$COMPARE_EQL
                66              50  E8 00094        BLBS    R0, 9$
                          08    AE  9F 00097        PUSHAB  STATE_NAME_DESC                         ; 3077
                          18    AE  9F 0009A        PUSHAB  KEY_NAME_DESC
                00000000G       00  9F 0009D        PUSHAB  DBG$GL_KEY_TABLE_ID
00000000G       00              03  FB 000A3        CALLS   #3, SMG$DELETE_KEY_DEF
                59              50  D0 000AA        MOVL    R0, DEL_STATUS
                29              59  E8 000AD        BLBS    DEL_STATUS, 7$                          ; 3080
00000000G       8F              59  D1 000B0        CMPL    DEL_STATUS, #SMG$_KEYNOTDEF             ; 3082
                                13  12 000B7        BNEQ    6$
                8A              5A  E9 000B9        BLBC    OUTPUT_LOG, 2$                          ; 3085
                          14    AE  9F 000BC        PUSHAB  KEY_NAME_DESC                           ; 3086
                          0C    AE  9F 000BF        PUSHAB  STATE_NAME_DESC
                                02  DD 000C2        PUSHL   #2
                000280CB        8F  DD 000C4        PUSHL   #164043
                                27  11 000CA        BRB     8$
                00028118        8F  DD 000CC 6$:    PUSHL   #164120                                 ; 3090
        00000000G       00          01  FB 000D2        CALLS   #1, LIB$SIGNAL
                        A5          5A  E9 000D9 7$:    BLBC    OUTPUT_LOG, 5$                      ; 3092
        00000000G       8F          59  D1 000DC        CMPL    DEL_STATUS, #SMG$_KEYNOTDEF
                                    9C  13 000E3        BEQL    5$
                                14  AE  9F 000E5        PUSHAB  KEY_NAME_DESC                       ; 3095
                                0C  AE  9F 000E8        PUSHAB  STATE_NAME_DESC
                                    02  DD 000EB        PUSHL   #2
                                000280BB    8F  DD 000ED        PUSHL   #164027
        00000000G       00              04  FB 000F3 8$:    CALLS   #4, LIB$SIGNAL
                                FF49        31 000FA        BRW     2$                              ; 3094
                        53          04  A3  D0 000FD 9$:    MOVL    4(TEMP_STATE_ADDRESS), TEMP_STATE_ADDRESS   ; 3103
                                FF7D        31 00101        BRW     5$                              ; 3066
                        5B                  52  D2 00104 10$:   MCOML   ALL_FLAG, R11              ; 3110
                        7C                  5B  E9 00107 11$:   BLBC    R11, 15$                    ; 3129
14  AE          00      B7                  08  28 0010A       MOVC3   #8, @0(NOUN_NODE), KEY_NAME_DESC    ; 3112
08  AE          00      B6                  08  28 00110       MOVC3   #8, @0(STATE_DESC_ADDRESS), STATE_NAME_DESC  ; 3113
                                    08  AE  9F 00116        PUSHAB  STATE_NAME_DESC                 ; 3115
                                    18  AE  9F 00119        PUSHAB  KEY_NAME_DESC
                                000000000G      00  9F 0011C        PUSHAB  DBG$GL_KEY_TABLE_ID
        00000000G       00                  03  FB 00122        CALLS   #3, SMG$DELETE_KEY_DEF
                        59                  50  D0 00129        MOVL    R0, DEL_STATUS
                        30                  59  E8 0012C        BLBS    DEL_STATUS, 13$             ; 3118
        00000000G       8F                  59  D1 0012F        CMPL    DEL_STATUS, #SMG$_KEYNOTDEF ; 3120
```

```
                        1A  12 00136          BNEQ    12$                                                     ;
                    45      5A  E9 00138          BLBC    OUTPUT_LOG, 14$                                         ; 3123
                        14  AE  9F 0013B          PUSHAB  KEY_NAME_DESC                                           ; 3124
                        0C  AE  9F 0013E          PUSHAB  STATE_NAME_DESC                                         ;
                            02  DD 00141          PUSHL   #2                                                      ;
                00000280CB  8F  DD 00143          PUSHL   #164043                                                 ;
00000000G       00          04  FB 00149          CALLS   #4, LIB$SIGNAL                                          ;
                            0D  11 00150          BRB     13$                                                     ; 3120
                00028118    8F  DD 00152 12$:     PUSHL   #164120                                                 ; 3127
00000000G       00          01  FB 00158          CALLS   #1, LIB$SIGNAL                                          ;
                    1E      5A  E9 0015F 13$:     BLBC    OUTPUT_LOG, 14$                                         ; 3129
00000000G       8F          59  D1 00162          CMPL    DEL_STATUS, #SMG$_KEYNOTDEF                             ;
                            15  13 00169          BEQL    14$                                                     ;
                        14  AE  9F 0016B          PUSHAB  KEY_NAME_DESC                                           ; 3131
                        0C  AE  9F 0016E          PUSHAB  STATE_NAME_DESC                                         ;
                            02  DD 00171          PUSHL   #2                                                      ;
                000280BB    8F  DD 00173          PUSHL   #164027                                                 ;
00000000G       00          04  FB 00179          CALLS   #4, LIB$SIGNAL                                          ;
                    56  04  A6  D0 00180 14$:     MOVL    4(STATE_DESC_ADDRESS), STATE_DESC_ADDRESS               ; 3133
                            81  12 00184          BNEQ    11$                                                     ; 3134
                            6C  11 00186 15$:     BRB     26$                                                     ; 3139
                    01      01  A8  91 00188 16$:     CMPB    1(R8), #1                                           ; 3145
                            07  12 0018C          BNEQ    17$                                                     ;
                        08  AC  DD 0018E          PUSHL   MESSAGE_VECT                                            ; 3147
                            7E  D4 00191          CLRL    -(SP)                                                   ;
                            0B  11 00193          BRB     18$                                                     ;
                    02      01  A8  91 00195 17$:     CMPB    1(R8), #2                                           ; 3151
                            0B  12 00199          BNEQ    19$                                                     ;
                        08  AC  DD 0019B          PUSHL   MESSAGE_VECT                                            ; 3153
                            01  DD 0019E          PUSHL   #1                                                      ;
            EF0C    CF          02  FB 001A0 18$:     CALLS   #2, DBG$DEF_SYM_REMOVE_ALL                          ;
                            04  001A5          RET                                                             ;
                    53          01  D0 001A6 19$:     MOVL    #1, FIRST_TIME                                      ; 3157
                    08          53  E9 001A9 20$:     BLBC    FIRST_TIME, 21$                                     ; 3164
                    57      08  A8  D0 001AC          MOVL    8(R8), NOUN_NODE                                    ; 3167
                            53  D4 001B0          CLRL    FIRST_TIME                                              ; 3171
                            06  11 001B2          BRB     22$                                                     ; 3164
                    57      08  A7  D0 001B4 21$:     MOVL    8(NOUN_NODE), NOUN_NODE                             ; 3180
                            3A  13 001B8          BEQL    26$                                                     ; 3181
                    01      04  A7  D1 001BA 22$:     CMPL    4(NOUN_NODE), #1                                    ; 3188
                            05  12 001BE          BNEQ    23$                                                     ;
                    52          01  D0 001C0          MOVL    #1, GLOBAL_FLAG                                     ; 3190
                            02  11 001C3          BRB     24$                                                     ;
                    52          52  D4 001C5 23$:     CLRL    GLOBAL_FLAG                                         ; 3192
                        08  AC  DD 001C7 24$:     PUSHL   MESSAGE_VECT                                            ; 3199
                        08  AE  9F 001CA          PUSHAB  REMOVED_FLAG                                            ; 3196
                            52  DD 001CD          PUSHL   GLOBAL_FLAG                                             ; 3197
                            67  DD 001CF          PUSHL   (NOUN_NODE)                                             ; 3196
            EE5F    CF          04  FB 001D1          CALLS   #4, DBG$DEF_SYM_REMOVE                              ;
                    04          50  E8 001D6          BLBS    R0, 25$                                             ;
                    50          04  D0 001D9          MOVL    #4, R0                                              ; 3201
                            04  001DC          RET                                                             ;
                    C8      04  AE  E8 001DD 25$:     BLBS    REMOVED_FLAG, 20$                                   ; 3205
                            67  DD 001E1          PUSHL   (NOUN_NODE)                                             ; 3207
                            01  DD 001E3          PUSHL   #1                                                      ;
                000286B3    8F  DD 001E5          PUSHL   #165555                                                 ;
00000000G       00          03  FB 001EB          CALLS   #3, LIB$SIGNAL                                          ;
```

```
                          B5  11 001F2              BRB     20$                                     ; 3158
                50        01  D0 001F4 26$:        MOVL    #1, R0                                  ; 3211
                              04 001F7              RET                                             ; 3212

; Routine Size:  504 bytes,    Routine Base:  DBG$CODE + 11A8
```

```
;  3098          3213  1 GLOBAL ROUTINE dbg$nread_name (input_desc, result_addr, message_vect) =
;  3099          3214  1 !++
;  3100          3215  1 ! Functional Description
;  3101          3216  1 !
;  3102          3217  1 !       Reads the name which is the left-hand-side of the DEFINE command.
;  3103          3218  1 !       For now, the names allowed are any string beginning with an
;  3104          3219  1 !       alphabetic and followed by any number of alphabetics or numerics
;  3105          3220  1 !       and also characters from the set { _, $, % }.
;  3106          3221  1 !       The input string descriptor is updated beyond the string that
;  3107          3222  1 !       is read. Space is allocated for a counted string to hold the
;  3108          3223  1 !       result.
;  3109          3224  1 !
;  3110          3225  1 ! Inputs
;  3111          3226  1 !
;  3112          3227  1 !       input_desc -    A pointer to a string descriptor with the input.
;  3113          3228  1 !       result_addr -   The address in which to place the result.
;  3114          3229  1 !       message_vect -  A pointer to an error message vector.
;  3115          3230  1 !
;  3116          3231  1 ! Outputs
;  3117          3232  1 !
;  3118          3233  1 !       Space is allocated for a counted string to hold the result.
;  3119          3234  1 !       A pointer to this counted string is returned in result_addr.
;  3120          3235  1 !       One of the following values is returned:
;  3121          3236  1 !       sts$k_success -         Success code.
;  3122          3237  1 !       sts$k_severe -          Failure. The input did not contain
;  3123          3238  1 !                               a legal name. An error message vector
;  3124          3239  1 !                               is constructed and returned in message_vect.
;  3125          3240  1 !--
;  3126          3241  2     BEGIN
;  3127          3242  2
;  3128          3243  2     MAP
;  3129          3244  2         input_desc: REF BLOCK [,BYTE];
;  3130          3245  2
;  3131          3246  2     LOCAL
;  3132          3247  2         char,                               ! Holds a character in the input stream
;  3133          3248  2         count,                              ! Count of characters in the name
;  3134          3249  2         first_char,                         ! Flag saying we are
;  3135          3250  2                                             !     reading the first character
;  3136          3251  2         pointer,                            ! Pointer into the input stream
;  3137          3252  2         result: REF VECTOR[,BYTE];          ! Holds the result
;  3138          3253  2
;  3139          3254  2     ! Check for exhausted input.
;  3140          3255  2     !
;  3141          3256  2     IF .input_desc [dsc$w_length] EQL 0
;  3142          3257  2     THEN
;  3143          3258  3         BEGIN
;  3144          3259  3         .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  3145          3260  3         RETURN sts$k_severe;
;  3146          3261  2         END;
;  3147          3262  2
;  3148          3263  2     ! Read past leading blanks.
;  3149          3264  2     !
;  3150          3265  2     WHILE TRUE DO
;  3151          3266  3         BEGIN
;  3152          3267  3         char = ch$rchar (.input_desc [dsc$a_pointer]);
;  3153          3268  3         IF  .char NEQ dbg$k_blank
;  3154          3269  3         AND .char NEQ dbg$k_tab
```

```
;  3155     3270  3          THEN
;  3156     3271  3              EXITLOOP;
;  3157     3272  3          input_desc [dsc$a_pointer] = ch$plus (.input_desc [dsc$a_pointer], 1);
;  3158     3273  3          input_desc [dsc$w_length] = .input_desc [dsc$w_length] - 1;
;  3159     3274  2          END;
;  3160     3275  2
;  3161     3276  2      ! Check for exhausted input again.
;  3162     3277  2      !
;  3163     3278  2      IF .input_desc [dsc$w_length] EQL 0
;  3164     3279  2      THEN
;  3165     3280  3          BEGIN
;  3166     3281  3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  3167     3282  3          RETURN sts$k_severe;
;  3168     3283  3          END;
;  3169     3284  2
;  3170     3285  2      ! Initialize the count to zero and the pointer to point to the
;  3171     3286  2      ! next character in the input stream.
;  3172     3287  2      !
;  3173     3288  2      count = 0;
;  3174     3289  2      pointer = .input_desc [dsc$a_pointer];
;  3175     3290  2
;  3176     3291  2      ! Read until we hit a character that cannot be part of the string.
;  3177     3292  2      !
;  3178     3293  2      first_char = TRUE;
;  3179     3294  2      WHILE TRUE DO
;  3180     3295  3          BEGIN
;  3181     3296  3          char = ch$rchar (.pointer);
;  3182     3297  3
;  3183     3298  3          ! For first character we accept only alphabetics.
;  3184     3299  3          ! Allow % also
;  3185     3300  3          !
;  3186     3301  4          IF .first_char AND (.char LSS 'A' OR .char GTR 'Z')
;  3187     3302  4                         AND (.char LSS 'a' OR .char GTR 'z')
;  3188     3303  3                         AND .char NEQ '%'
;  3189     3304  3          THEN
;  3190     3305  3              EXITLOOP;
;  3191     3306  3
;  3192     3307  3          ! Set first_char to FALSE for future times around loop.
;  3193     3308  3          !
;  3194     3309  3          first_char = FALSE;
;  3195     3310  3
;  3196     3311  3          ! Accept alphabetics, numerics, $, _, %
;  3197     3312  3          !
;  3198     3313  3          IF (.char LSS 'A' OR .char GTR 'Z') AND .char NEQ '_'
;  3199     3314  3                         AND .char NEQ '$' AND .char NEQ '%'
;  3200     3315  4                         AND (.char LSS '0' OR .char GTR '9')
;  3201     3316  4                         AND (.char LSS 'a' OR .char GTR 'z')
;  3202     3317  3          THEN
;  3203     3318  3              EXITLOOP;
;  3204     3319  3
;  3205     3320  3          count = .count + 1;
;  3206     3321  3          pointer = ch$plus (.pointer, 1);
;  3207     3322  2          END;
;  3208     3323  2
;  3209     3324  2      ! Check for running off the end of the input stream.
;  3210     3325  2      !
;  3211     3326  2      IF .count GTR .input_desc [dsc$w_length]
```

```
;  3212          3327  2     THEN
;  3213          3328  3         BEGIN
;  3214          3329  3         .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  3215          3330  3         RETURN sts$k_severe;
;  3216          3331  2         END;
;  3217          3332  2
;  3218          3333  2     ! Check for no characters read.
;  3219          3334  2     !
;  3220          3335  2     IF .count EQL 0
;  3221          3336  2     THEN
;  3222          3337  3         BEGIN
;  3223          3338  3         .message_vect = dbg$nmake_arg_vect (dbg$_illdefnam, 1,
;  3224          3339  3                 dbg$nnext_word(.input_desc));
;  3225          3340  3         RETURN sts$k_severe;
;  3226          3341  2         END;
;  3227          3342  2
;  3228          3343  2     ! Allocate space for the result.
;  3229          3344  2     !
;  3230          3345  2     result = dbg$get_memory (1+(1+.count)/4);
;  3231          3346  2
;  3232          3347  2     ! Fill in the result, translating lower case to upper case.
;  3233          3348  2     !
;  3234          3349  2     result[0] = .count;
;  3235          3350  2     pointer = result[1];
;  3236          3351  2     INCR i FROM 1 TO .count DO
;  3237          3352  3         BEGIN
;  3238          3353  3         char = ch$rchar_a (input_desc[dsc$a_pointer]);
;  3239          3354  3         IF .char GEQ 'a' AND .char LEQ 'z'
;  3240          3355  3         THEN
;  3241          3356  3             char = .char - ('a' - 'A');
;  3242          3357  3         ch$wchar_a (.char, pointer);
;  3243          3358  2         END;
;  3244          3359  2     .result_addr = .result;
;  3245          3360  2
;  3246          3361  2     ! Update the input descriptor to point past the string that was read.
;  3247          3362  2     ! The pointer has already been advanced.
;  3248          3363  2     !
;  3249          3364  2     input_desc [dsc$w_length] = .input_desc [dsc$w_length] - .count;
;  3250          3365  2
;  3251          3366  2     RETURN sts$k_success;
;  3252          3367  2
;  3253          3368  1     END; ! dbg$nread_name


                                00FC 00000          .ENTRY  DBG$NREAD_NAME, Save R2,R3,R4,R5,R6,R7      ; 3213
                    57 00000000G   00   9E 00002          MOVAB   DBG$NMAKE_ARG_VECT, R7                      ;
                    54         04  AC   D0 00009          MOVL    INPUT_DESC, R4                              ; 3256
                                   64   B5 0000D          TSTW    (R4)                                        ;
                                   1A   13 0000F          BEQL    4$                                          ;
                    56         04  A4   9E 00011          MOVAB   4(R4), R6                                   ; 3267
                    53         00  B6   9A 00015 1$:      MOVZBL  @0(R6), CHAR                                ;
                    20             53   D1 00019          CMPL    CHAR, #32                                   ; 3268
                                   05   13 0001C          BEQL    2$                                          ;
                    09             53   D1 0001E          CMPL    CHAR, #9                                    ; 3269
```

```
                                        06  12 00021          BNEQ    3$                                                  ;
                                        66  D6 00023 2$:      INCL    (R6)                                                ; 3272
                                        64  B7 00025          DECW    (R4)                                                ; 3273
                                        EC  11 00027          BRB     1$                                                  ; 3265
                                        64  B5 00029 3$:      TSTW    (R4)                                                ; 3278
                                        03  12 0002B 4$:      BNEQ    5$                                                  ;
                                      0087  31 0002D          BRW     14$                                                 ;
                                        52  D4 00030 5$:      CLRL    COUNT                                               ; 3288
                                    55  66  D0 00032          MOVL    (R6), POINTER                                       ; 3289
                                    50  01  D0 00035          MOVL    #1, FIRST_CHAR                                      ; 3293
                                    53  65  9A 00038 6$:      MOVZBL  (POINTER), CHAR                                     ; 3296
                                    29  50  E9 0003B          BLBC    FIRST_CHAR, 9$                                      ; 3301
                          00000041  8F  53  D1 0003E          CMPL    CHAR, #65                                           ;
                                        09  19 00045          BLSS    7$                                                  ;
                          0000005A  8F  53  D1 00047          CMPL    CHAR, #90                                           ;
                                        17  15 0004E          BLEQ    9$                                                  ;
                          00000061  8F  53  D1 00050 7$:      CMPL    CHAR, #97                                           ; 3302
                                        09  19 00057          BLSS    8$                                                  ;
                          0000007A  8F  53  D1 00059          CMPL    CHAR, #122                                          ;
                                        05  15 00060          BLEQ    9$                                                  ;
                                25      53  D1 00062 8$:      CMPL    CHAR, #37                                           ; 3303
                                        49  12 00065          BNEQ    13$                                                 ;
                                        50  D4 00067 9$:      CLRL    FIRST_CHAR                                          ; 3309
                          00000041  8F  53  D1 00069          CMPL    CHAR, #65                                           ; 3313
                                        09  19 00070          BLSS    10$                                                 ;
                          0000005A  8F  53  D1 00072          CMPL    CHAR, #90                                           ;
                                        2F  15 00079          BLEQ    12$                                                 ;
                          0000005F  8F  53  D1 0007B 10$:     CMPL    CHAR, #95                                           ;
                                        26  13 00082          BEQL    12$                                                 ;
                                24      53  D1 00084          CMPL    CHAR, #36                                           ; 3314
                                        21  13 00087          BEQL    12$                                                 ;
                                25      53  D1 00089          CMPL    CHAR, #37                                           ;
                                        1C  13 0008C          BEQL    12$                                                 ;
                                30      53  D1 0008E          CMPL    CHAR, #48                                           ; 3315
                                        05  19 00091          BLSS    11$                                                 ;
                                39      53  D1 00093          CMPL    CHAR, #57                                           ;
                                        12  15 00096          BLEQ    12$                                                 ;
                          00000061  8F  53  D1 00098 11$:     CMPL    CHAR, #97                                           ; 3316
                                        0F  19 0009F          BLSS    13$                                                 ;
                          0000007A  8F  53  D1 000A1          CMPL    CHAR, #122                                          ;
                                        06  14 000A8          BGTR    13$                                                 ;
                                        52  D6 000AA 12$:     INCL    COUNT                                               ; 3320
                                        55  D6 000AC          INCL    POINTER                                             ; 3321
                                        88  11 000AE          BRB     6$                                                  ; 3294
52                        64        10      00  ED 000B0 13$:     CMPZV   #0, #16, (R4), COUNT                                ; 3326
                                        0B  18 000B5          BGEQ    15$                                                 ;
                                    000280D0    8F  DD 000B7 14$:     PUSHL   #164048                                     ; 3329
                                67      01  FB 000BD          CALLS   #1, DBG$NMAKE_ARG_VECT                              ;
                                        1A  11 000C0          BRB     16$                                                 ;
                                        52  D5 000C2 15$:     TSTL    COUNT                                               ; 3335
                                        1E  12 000C4          BNEQ    17$                                                 ;
                                        54  DD 000C6          PUSHL   R4                                                  ; 3339
                          00000000G 00      01  FB 000C8          CALLS   #1, DBG$NNEXT_WORD                                  ;
                                        50  DD 000CF          PUSHL   R0                                                  ;
                                        01  DD 000D1          PUSHL   #1                                                  ; 3338
                                    00028E50    8F  DD 000D3          PUSHL   #167504                                     ;
                                67      03  FB 000D9          CALLS   #3, DBG$NMAKE_ARG_VECT                              ;
```

```
                      0C   BC           50   D0 000DC 16$:   MOVL    R0, @MESSAGE_VECT                       ;
                           50           04   D0 000E0        MOVL    #4, R0                                  ; 3340
                                        04   000E3           RET                                             ;
                           50      01   A2   9E 000E4 17$:   MOVAB   1(R2), R0                               ; 3345
                           50           04   C6 000E8        DIVL2   #4, R0                                  ;
                                   01   A0   9F 000EB        PUSHAB  1(R0)                                   ;
                00000000G  00           01   FB 000EE        CALLS   #1, DBG$GET_MEMORY                      ;
                           60           52   90 000F5        MOVB    COUNT, (RESULT)                         ; 3349
                           55      01   A0   9E 000F8        MOVAB   1(R0), POINTER                          ; 3350
                                        51   D4 000FC        CLRL    I                                       ; 3351
                                        1E   11 000FE        BRB     20$                                     ;
                           53      00   B6   9A 00100 18$:   MOVZBL  @0(R6), CHAR                           ; 3353
                                        66   D6 00104        INCL    (R6)                                    ;
                00000061   8F           53   D1 00106        CMPL    CHAR, #97                               ; 3354
                                        0C   19 00108        BLSS    19$                                     ;
                0000007A   8F           53   D1 0010F        CMPL    CHAR, #122                              ;
                                        03   14 00116        BGTR    19$                                     ;
                           53           20   C2 00118        SUBL2   #32, CHAR                               ; 3356
                           85           53   90 0011B 19$:   MOVB    CHAR, (POINTER)+                        ; 3357
                     DE    51           52   F3 0011E 20$:   AOBLEQ  COUNT, I, 18$                          ; 3351
                      08   BC           50   D0 00122        MOVL    RESULT, @RESULT_ADDR                    ; 3359
                           64           52   A2 00126        SUBW2   COUNT, (R4)                             ; 3364
                           50           01   D0 00129        MOVL    #1, R0                                  ; 3366
                                        04   0012C           RET                                             ; 3368

; Routine Size:  301 bytes,    Routine Base:  DBG$CODE + 13A0
```

```
;  3255          3369  1 GLOBAL ROUTINE dbg$save_loc (desc1, desc2): NOVALUE =
;  3256          3370  1 !
;  3257          3371  1 ! ROUTINE FUNCTION
;  3258          3372  1 !
;  3259          3373  1 !       Save away the given descriptor(s) as the current value of dot.
;  3260          3374  1 !
;  3261          3375  1 ! INPUTS
;  3262          3376  1 !
;  3263          3377  1 !       DESC1           - points to a descriptor to be saved as .
;  3264          3378  1 !       DESC2           - points to a VMS descriptor which may also be
;  3265          3379  1 !                         saved. This is stored in an own variable in
;  3266          3380  1 !                         this module.
;  3267          3381  1 !
;  3268          3382  1 ! IMPLICIT OUTPUT
;  3269          3383  1 !
;  3270          3384  1 !       A copy of the given descriptor is constructed out of permanent memory.
;  3271          3385  1 !       The DEFINE table is modified to include a new entry for %CURLOC.
;  3272          3386  1 !       The secondary descriptor, if present, is copied into an area in this
;  3273          3387  1 !       module.
;  3274          3388  1 !
;  3275          3389  2     BEGIN
;  3276          3390  2
;  3277          3391  2     BUILTIN
;  3278          3392  2         actualcount;            ! Count of actual parameters
;  3279          3393  2
;  3280          3394  2     LOCAL
;  3281          3395  2         desc1_copy,             ! Points to a copy of DESC1.
;  3282          3396  2         dummy,                  ! Third parameter for message vectors
;  3283          3397  2                                 !    (not used here)
;  3284          3398  2         name,                   ! Point to name %CURLOC.
;  3285          3399  2         symid_list,             ! Points to a symid list.
;  3286          3400  2         vms_desc;               ! Points to the vms descriptor to be saved
;  3287          3401  2
;  3288          3402  2     IF actualcount() LSS 2
;  3289          3403  2     THEN
;  3290          3404  2         vms_desc = 0
;  3291          3405  2     ELSE
;  3292          3406  2         vms_desc = .desc2;
;  3293          3407  2
;  3294          3408  2     ! Copy the descriptor.
;  3295          3409  2     !
;  3296          3410  2     dbg$nget_symid (.desc1, symid_list, dummy);
;  3297          3411  2     dbg$ncopy_desc (.desc1, desc1_copy, dummy);
;  3298          3412  2     dbg$sta_lock_symid (.symid_list);
;  3299          3413  2
;  3300          3414  2     ! Save away the copy.
;  3301          3415  2     ! The name must be allocated out of permanent memory.
;  3302          3416  2     !
;  3303          3417  2     name = dbg$get_memory (2);
;  3304          3418  2     ch$move (8, UPLIT BYTE (%ASCIC '%CURLOC'), .name);
;  3305          3419  2     dbg$def_sym_add(.name, define_address, .desc1_copy, TRUE, dummy, dummy);
;  3306          3420  2
;  3307          3421  2     ! See if a secondary descriptor was supplied.
;  3308          3422  2     !
;  3309          3423  2     IF .vms_desc EQL 0
;  3310          3424  2     THEN
;  3311          3425  2         dbg$gl_curloc_vmsdesc = 0
```

```
;  3312          3426  2     ELSE
;  3313          3427  3         BEGIN
;  3314          3428  3
;  3315          3429  3         ! Copy it into the area in this module.
;  3316          3430  3         ! and set up the pointer to this area.
;  3317          3431  3         !
;  3318          3432  3         ch$move (12, .vms_desc, curloc_vmsdesc);
;  3319          3433  3         dbg$gl_curloc_vmsdesc = curloc_vmsdesc;
;  3320          3434  2         END;
;  3321          3435  2
;  3322          3436  2     RETURN;
;  3323          3437  1     END;


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

                      43 4F 4C 52 55 43 25 07  00118 P.ACD:  .ASCII  <7>\%CURLOC\                              ;


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

                                 03FC 00000             .ENTRY  DBG$SAVE_LOC, Save R2,R3,R4,R5,R6,R7,R8,R9  ; 3369
                   59 00000000'  EF  9E 00002           MOVAB   CURLOC_VMSDESC, R9                          ;
                   58 00000000'  EF  9E 00009           MOVAB   DBG$GL_CURLOC_VMSDESC, R8                   ;
                   5E            0C  C2 00010           SUBL2   #12, SP                                     ;
                   02            6C  91 00013           CMPB    (AP), #2                                    ; 3402
                                 04  1E 00016           BGEQU   1$                                          ;
                                 57  D4 00018           CLRL    VMS_DESC                                    ; 3404
                                 04  11 0001A           BRB     2$                                          ;
                         57      08  AC  D0 0001C 1$:    MOVL    DESC2, VMS_DESC                             ; 3406
                                 08  AE  9F 00020 2$:    PUSHAB  DUMMY                                       ; 3410
                                 04  AE  9F 00023        PUSHAB  SYMID_LIST                                  ;
                                 04  AC  DD 00026        PUSHL   DESC1                                       ;
                00000000G  00        03  FB 00029        CALLS   #3, DBG$NGET_SYMID                          ;
                                 08  AE  9F 00030        PUSHAB  DUMMY                                       ; 3411
                                 08  AE  9F 00033        PUSHAB  DESC1_COPY                                  ;
                                 04  AC  DD 00036        PUSHL   DESC1                                       ;
                00000000G  00        03  FB 00039        CALLS   #3, DBG$NCOPY_DESC                          ;
                                     6E  DD 00040        PUSHL   SYMID_LIST                                  ; 3412
                00000000G  00        01  FB 00042        CALLS   #1, DBG$STA_LOCK_SYMID                      ;
                                     02  DD 00049        PUSHL   #2                                          ; 3417
                00000000G  00        01  FB 0004B        CALLS   #1, DBG$GET_MEMORY                          ;
                           56        50  D0 00052        MOVL    R0, NAME                                    ;
             66 00000000'  EF        08  28 00055        MOVC3   #8, P.ACD, (NAME)                           ; 3418
                                 08  AE  9F 0005D        PUSHAB  DUMMY                                       ; 3419
                                 0C  AE  9F 00060        PUSHAB  DUMMY                                       ;
                                     01  DD 00063        PUSHL   #1                                          ;
                                 10  AE  DD 00065        PUSHL   DESC1_COPY                                  ;
                                     01  DD 00068        PUSHL   #1                                          ;
                                     56  DD 0006A        PUSHL   NAME                                        ;
                      EAF2       CF      06  FB 0006C    CALLS   #6, DBG$DEF_SYM_ADD                         ;
                                     57  D5 00071        TSTL    VMS_DESC                                    ; 3423
                                     03  12 00073        BNEQ    3$                                          ;
                                     68  D4 00075        CLRL    DBG$GL_CURLOC_VMSDESC                       ; 3425
                                         04 00077        RET                                                 ;
```

```
                     69          67          0C  28 00078 3$:     MOVC3   #12, (VMS_DESC), CURLOC_VMSDESC          ; 3432
                                 68          69  9E 0007C         MOVAB   CURLOC_VMSDESC, DBG$GL_CURLOC_VMSDESC   ; 3433
                                             04  0007F         RET                                             ; 3437

; Routine Size:  128 bytes,    Routine Base:  DBG$CODE + 14CD
```

```
;  3325          3438  1 GLOBAL ROUTINE dbg$save_val (desc): NOVALUE =
;  3326          3439  1 !
;  3327          3440  1 ! ROUTINE FUNCTION
;  3328          3441  1 !
;  3329          3442  1 !       Save away the given descriptor as the current value of backslash.
;  3330          3443  1 !
;  3331          3444  1 ! INPUTS
;  3332          3445  1 !
;  3333          3446  1 !       DESC            - points to a descriptor to be saved as \
;  3334          3447  1 !
;  3335          3448  1 ! IMPLICIT OUTPUT
;  3336          3449  1 !
;  3337          3450  1 !       A copy of the given descriptor is constructed out of permanent memory.
;  3338          3451  1 !       The DEFINE table is modified to include a new entry for %CURVAL.
;  3339          3452  1 !
;  3340          3453  2     BEGIN
;  3341          3454  2
;  3342          3455  2     LOCAL
;  3343          3456  2         desc_copy,              ! Points to a copy of DESC.
;  3344          3457  2         dummy,                  ! Third parameter for message vectors
;  3345          3458  2                                 !    (not used here)
;  3346          3459  2         name,                   ! Point to name %CURVAL
;  3347          3460  2         symid_list;             ! Points to a symid list.
;  3348          3461  2
;  3349          3462  2     ! Copy the descriptor.
;  3350          3463  2     !
;  3351          3464  2     dbg$nget_symid (.desc, symid_list, dummy);
;  3352          3465  2     dbg$ncopy_desc (.desc, desc_copy, dummy);
;  3353          3466  2     dbg$sta_lock_symid (.symid_list);
;  3354          3467  2
;  3355          3468  2     ! Save away the copy.
;  3356          3469  2     ! The name must be allocated out of permanent memory.
;  3357          3470  2     !
;  3358          3471  2     name = dbg$get_memory (2);
;  3359          3472  2     ch$move (8, UPLIT BYTE (%ASCIC '%CURVAL'), .name);
;  3360          3473  2     dbg$def_sym_add(.name, define_value, .desc_copy, TRUE, dummy, dummy);
;  3361          3474  2
;  3362          3475  2     RETURN;
;  3363          3476  1     END;


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

                4C  41  56  52  55  43  25  07  00120 P.ACE:  .ASCII  <7>\%CURVAL\                                ;


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

                                    007C 00000          .ENTRY  DBG$SAVE_VAL, Save R2,R3,R4,R5,R6                ; 3438
                        5E          0C   C2 00002       SUBL2   #12, SP                                          ;
                                08  AE   9F 00005       PUSHAB  DUMMY                                            ; 3464
                                04  AE   9F 00008       PUSHAB  SYMID_LIST                                       ;
                                04  AC   DD 0000B       PUSHL   DESC                                             ;
                00000000G       00  03   FB 0000E       CALLS   #3, DBG$NGET_SYMID                               ;
                                08  AE   9F 00015       PUSHAB  DUMMY                                            ; 3465
```

```
                                08 AE 9F 00018   PUSHAB  DESC_COPY                                ;
                                04 AC DD 0001B   PUSHL   DESC                                    ;
                  00000000G   00    03 FB 0001E   CALLS   #3, DBG$NCOPY_DESC                     ;
                                    6E DD 00025   PUSHL   SYMID_LIST                             ; 3466
                  00000000G   00    01 FB 00027   CALLS   #1, DBG$STA_LOCK_SYMID                 ;
                                    02 DD 0002E   PUSHL   #2                                     ; 3471
                  00000000G   00    01 FB 00030   CALLS   #1, DBG$GET_MEMORY                     ;
                                56  50 DO 00037   MOVL    R0, NAME                               ;
               66 00000000'   EF    08 28 0003A   MOVC3   #8, P.ACE, (NAME)                      ; 3472
                                08 AE 9F 00042   PUSHAB  DUMMY                                   ; 3473
                                0C AE 9F 00045   PUSHAB  DUMMY                                   ;
                                    01 DD 00048   PUSHL   #1                                     ;
                                10 AE DD 0004A   PUSHL   DESC_COPY                               ;
                                    05 DD 0004D   PUSHL   #5                                     ;
                                    56 DD 0004F   PUSHL   NAME                                   ;
                     EA8D       CF    06 FB 00051   CALLS   #6, DBG$DEF_SYM_ADD                  ;
                                      04 00056   RET                                            ; 3476

; Routine Size:  87 bytes,    Routine Base:  DBG$CODE + 154D
```

```
;  3365          3477  1 GLOBAL ROUTINE DBG$WILDCARD_NAME_MATCH (NAME1, NAME2) =
;  3366          3478  1 !++
;  3367          3479  1 ! Functional Description
;  3368          3480  1 !       Determines whether the wildcarded name in NAME1 matches the name
;  3369          3481  1 !       in NAME2. Asterisk is the wildcard character, and it may match any
;  3370          3482  1 !       string of zero or more characters.
;  3371          3483  1 !
;  3372          3484  1 ! Inputs
;  3373          3485  1 !       NAME1           - points to a counted string containing a name. The
;  3374          3486  1 !                         name may include one or more * (the wildcard char)
;  3375          3487  1 !       NAME2           - points to a counted string containing a name.
;  3376          3488  1 !
;  3377          3489  1 ! Outputs
;  3378          3490  1 !       The return value is one of:
;  3379          3491  1 !       TRUE    - the names do match
;  3380          3492  1 !       FALSE   - the names do not match
;  3381          3493  1 !--
;  3382          3494  2     BEGIN
;  3383          3495  2
;  3384          3496  2     MAP
;  3385          3497  2         NAME1   : REF VECTOR [,BYTE],
;  3386          3498  2         NAME2   : REF VECTOR [,BYTE];
;  3387          3499  2
;  3388          3500  2     LOCAL
;  3389          3501  2         ASTER_PTR,                  ! Points to * in NAME1
;  3390          3502  2         LENGTH,                     ! Length to asterisk
;  3391          3503  2         LENGTH1,                    ! Remaining length of first string
;  3392          3504  2         LENGTH2,                    ! Remaining length of second string
;  3393          3505  2         NAME1_PTR,                  ! Pointer into NAME1 string
;  3394          3506  2         NAME2_PTR,                  ! Pointer into NAME2 string
;  3395          3507  2         NEW_NAME1: REF VECTOR[,BYTE],   ! Padded copy of NAME1
;  3396          3508  2         NEW_NAME2: REF VECTOR[,BYTE],   ! Padded copy of NAME2
;  3397          3509  2         POOLID,                     ! Memory Pool id
;  3398          3510  2         RET_VALUE,                  ! Return value
;  3399          3511  2         SUBSTR_PTR;                 ! Points to substring that we find
;  3400          3512  2
;  3401          3513  2     ! Copy names into new area padded on left and right.
;  3402          3514  2     !
;  3403          3515  2     POOLID = DBG$PUSH_TEMPMEM();
;  3404          3516  2
;  3405          3517  2     NEW_NAME1 = DBG$GET_TEMPMEM ((5+.NAME1[0])/4);
;  3406          3518  2     CH$MOVE (.NAME1[0], NAME1[1], NEW_NAME1[1]);
;  3407          3519  2     NEW_NAME1[0] = 0;
;  3408          3520  2     NEW_NAME1[1+.NAME1[0]] = 0;
;  3409          3521  2
;  3410          3522  2     NEW_NAME2 = DBG$GET_TEMPMEM ((5+.NAME2[0])/4);
;  3411          3523  2     CH$MOVE (.NAME2[0], NAME2[1], NEW_NAME2[1]);
;  3412          3524  2     NEW_NAME2[0] = 0;
;  3413          3525  2     NEW_NAME2[1+.NAME2[0]] = 0;
;  3414          3526  2
;  3415          3527  2     ! Initialize lengths, pointers, and flags.
;  3416          3528  2     !
;  3417          3529  2     LENGTH1 = 2+.NAME1[0];
;  3418          3530  2     NAME1_PTR = .NEW_NAME1;
;  3419          3531  2     LENGTH2 = 2+.NAME2[0];
;  3420          3532  2     NAME2_PTR = .NEW_NAME2;
;  3421          3533  2     RET_VALUE = TRUE;
```

```
;  3422          3534  2
;  3423          3535  2      ! Loop through the portions of the string between asterisks.
;  3424          3536  2      !
;  3425          3537  2      WHILE .LENGTH1 GTR 0 DO
;  3426          3538  3          BEGIN
;  3427          3539  3
;  3428          3540  3          ! If we have exhausted the second string, return false.
;  3429          3541  3          !
;  3430          3542  3          IF .LENGTH2 LEQ 0
;  3431          3543  3          THEN
;  3432          3544  4              BEGIN
;  3433          3545  4              RET_VALUE = FALSE;
;  3434          3546  4              EXITLOOP;
;  3435          3547  3              END;
;  3436          3548  3
;  3437          3549  3          ! Obtain pointer to first asterisk.
;  3438          3550  3          !
;  3439          3551  3          ASTER_PTR = CH$FIND_CH (.LENGTH1, .NAME1_PTR, '*');
;  3440          3552  3          IF .ASTER_PTR EQL 0
;  3441          3553  3          THEN
;  3442          3554  3              ASTER_PTR = .NAME1_PTR + .LENGTH1;
;  3443          3555  3          LENGTH = .ASTER_PTR - .NAME1_PTR;
;  3444          3556  3
;  3445          3557  3          ! Look for the next match.
;  3446          3558  3          !
;  3447          3559  3          SUBSTR_PTR = CH$FIND_SUB ( .LENGTH2,     ! Context length
;  3448          3560  3                                     .NAME2_PTR,  ! Context pointer
;  3449          3561  3                                     .LENGTH,           ! Pattern length
;  3450          3562  3                                     .NAME1_PTR); ! Pattern pointer
;  3451          3563  3          IF .SUBSTR_PTR EQL 0
;  3452          3564  3          THEN
;  3453          3565  4              BEGIN
;  3454          3566  4              RET_VALUE = FALSE;
;  3455          3567  4              EXITLOOP;
;  3456          3568  3              END;
;  3457          3569  3
;  3458          3570  3          LENGTH1 = .LENGTH1 - (1 + .LENGTH);
;  3459          3571  3          NAME1_PTR = .ASTER_PTR + 1;
;  3460          3572  3          LENGTH2 = .LENGTH2 - (.LENGTH + .SUBSTR_PTR - .NAME2_PTR);
;  3461          3573  3          NAME2_PTR = .SUBSTR_PTR + .LENGTH;
;  3462          3574  2          END;
;  3463          3575  2
;  3464          3576  2      ! If we have exactly matched the second string, return true.
;  3465          3577  2      !
;  3466          3578  2      IF .LENGTH2 NEQ 0
;  3467          3579  2      THEN
;  3468          3580  2          RET_VALUE = FALSE;
;  3469          3581  2
;  3470          3582  2      DBG$POP_TEMPMEM (.POOLID);
;  3471          3583  2      RETURN .RET_VALUE;
;  3472          3584  1      END; ! dbg$wildcard_name_match


                                                OFFC 00000          .ENTRY  DBG$WILDCARD_NAME_MATCH, Save R2,R3,R4,R5,- ; 3477
```

```
                                                              R6,R7,R8,R9,R10,R11
                 00000000G  00        00  FB 00002        CALLS   #0, DBG$PUSH_TEMPMEM                          ; 3515
                            5B        50  D0 00009        MOVL    R0, POOLID                                    ;
                            59    04  AC  D0 0000C        MOVL    NAME1, R9                                     ; 3517
                            50        69  9A 00010        MOVZBL  (R9), R0                                      ;
                            50        05  C0 00013        ADDL2   #5, R0                                        ;
              7E            50        04  C7 00016        DIVL3   #4, R0, -(SP)                                 ;
                 00000000G  00        01  FB 0001A        CALLS   #1, DBG$GET_TEMPMEM                           ;
                            57        50  D0 00021        MOVL    R0, NEW_NAME1                                 ;
                            50        69  9A 00024        MOVZBL  (R9), R0                                      ; 3518
       01     A7        01  A9        50  28 00027        MOVC3   R0, 1(R9), 1(NEW_NAME1)                       ;
                                      67  94 0002D        CLRB    (NEW_NAME1)                                   ; 3519
                            50        69  9A 0002F        MOVZBL  (R9), R0                                      ; 3520
                                  01 A047 94 00032        CLRB    1(R0)[NEW_NAME1]                              ;
                            58    08  AC  D0 00036        MOVL    NAME2, R8                                     ; 3522
                            50        68  9A 0003A        MOVZBL  (R8), R0                                      ;
                            50        05  C0 0003D        ADDL2   #5, R0                                        ;
              7E            50        04  C7 00040        DIVL3   #4, R0, -(SP)                                 ;
                 00000000G  00        01  FB 00044        CALLS   #1, DBG$GET_TEMPMEM                           ;
                            56        50  D0 0004B        MOVL    R0, NEW_NAME2                                 ;
                            50        68  9A 0004E        MOVZBL  (R8), R0                                      ; 3523
       01     A6        01  A8        50  28 00051        MOVC3   R0, 1(R8), 1(NEW_NAME2)                       ;
                                      66  94 00057        CLRB    (NEW_NAME2)                                   ; 3524
                            50        68  9A 00059        MOVZBL  (R8), R0                                      ; 3525
                                  01 A046 94 0005C        CLRB    1(R0)[NEW_NAME2]                              ;
                            54        69  9A 00060        MOVZBL  (R9), LENGTH1                                 ; 3529
                            54        02  C0 00063        ADDL2   #2, LENGTH1                                   ;
                            59        68  9A 00066        MOVZBL  (R8), LENGTH2                                 ; 3531
                            59        02  C0 00069        ADDL2   #2, LENGTH2                                   ;
                            55        56  D0 0006C        MOVL    NEW_NAME2, NAME2_PTR                          ; 3532
                            58        01  D0 0006F        MOVL    #1, RET_VALUE                                 ; 3533
                                      54  D5 00072 1$:    TSTL    LENGTH1                                       ; 3537
                                      49  15 00074        BLEQ    7$                                            ;
                                      59  D5 00076        TSTL    LENGTH2                                       ; 3542
                                      24  15 00078        BLEQ    5$                                            ;
              67            54        2A  3A 0007A        LOCC    #42, LENGTH1, (NAME1_PTR)                     ; 3551
                                      02  12 0007E        BNEQ    2$                                            ;
                                      51  D4 00080        CLRL    R1                                            ;
                            5A        51  D0 00082 2$:    MOVL    R1, ASTER_PTR                                 ;
                                      04  12 00085        BNEQ    3$                                            ; 3552
              5A            57        54  C1 00087        ADDL3   LENGTH1, NAME1_PTR, ASTER_PTR                 ; 3554
              56            5A        57  C3 0008B 3$:    SUBL3   NAME1_PTR, ASTER_PTR, LENGTH                  ; 3555
  65          59            67        56  39 0008F        MATCHC  LENGTH, (NAME1_PTR), LENGTH2, (NAME2_PTR)     ; 3562
                                      03  13 00094        BEQL    4$                                            ;
                            53        56  D0 00096        MOVL    LENGTH, R3                                    ;
                            53        56  C2 00099 4$:    SUBL2   LENGTH, R3                                    ;
                                      04  12 0009C        BNEQ    6$                                            ; 3563
                                      58  D4 0009E 5$:    CLRL    RET_VALUE                                     ; 3566
                                      1D  11 000A0        BRB     7$                                            ; 3565
              50            54        56  C3 000A2 6$:    SUBL3   LENGTH, LENGTH1, R0                           ; 3570
                            54    FF  A0  9E 000A6        MOVAB   -1(R0), LENGTH1                               ;
                            57    01  AA  9E 000AA        MOVAB   1(R10), NAME1_PTR                             ; 3571
              50            56        53  C1 000AE        ADDL3   SUBSTR_PTR, LENGTH, R0                        ; 3572
              50            55        50  C3 000B2        SUBL3   R0, NAME2_PTR, R0                             ;
                            59        50  C0 000B6        ADDL2   R0, LENGTH2                                   ;
              55            53        56  C1 000B9        ADDL3   LENGTH, SUBSTR_PTR, NAME2_PTR                 ; 3573
                                      B3  11 000BD        BRB     1$                                            ; 3537
```

```
                                 59  D5 000BF 7$:     TSTL    LENGTH2                                 ; 3578
                                 02  13 000C1        BEQL    8$                                      ;
                                 58  D4 000C3        CLRL    RET_VALUE                               ; 3580
                                 5B  DD 000C5 8$:     PUSHL   POOLID                                  ; 3582
                     00000000G   00  01  FB 000C7    CALLS   #1, DBG$POP_TEMPMEM                     ;
                                 50  58  D0 000CE    MOVL    RET_VALUE, R0                           ; 3583
                                     04 000D1        RET                                             ; 3584

; Routine Size:  210 bytes,    Routine Base:  DBG$CODE + 15A4
```

```
;  3474     3585  1 ROUTINE dump_entry (sym_ptr, addr_flag, type_flag, message_vect) =
;  3475     3586  1 !++
;  3476     3587  1 ! Routine Description
;  3477     3588  1 !
;  3478     3589  1 !       This routine dumps an entry in a define list.
;  3479     3590  1 !
;  3480     3591  1 ! Inputs
;  3481     3592  1 !
;  3482     3593  1 !       sym_ptr         - A pointer to the entry.
;  3483     3594  1 !       addr_flag       - says to display the thing that the symbol is bound to
;  3484     3595  1 !       type_flag       - says to display the kind of defined symbol
;  3485     3596  1 !       message_vect    - An error message vector
;  3486     3597  1 !
;  3487     3598  1 ! Outputs
;  3488     3599  1 !
;  3489     3600  1 !       The entry is displayed at the terminal.
;  3490     3601  1 !       A status code is returned.
;  3491     3602  1 !--
;  3492     3603  2     BEGIN
;  3493     3604  2
;  3494     3605  2     MAP
;  3495     3606  2         sym_ptr: REF define$entry;
;  3496     3607  2
;  3497     3608  2     LOCAL
;  3498     3609  2         sym_kind: BYTE; ! Holds code for kind of symbol
;  3499     3610  2
;  3500     3611  2     ! Print the symbol name.
;  3501     3612  2     !
;  3502     3613  2     dbg$print (UPLIT BYTE (%ASCIC 'defined '));
;  3503     3614  2     dbg$print (.sym_ptr [def$a_name]);
;  3504     3615  2     dbg$newline();
;  3505     3616  2
;  3506     3617  2     ! Print the value.
;  3507     3618  2     !
;  3508     3619  2     IF .addr_flag
;  3509     3620  2     THEN
;  3510     3621  3         BEGIN
;  3511     3622  3         dbg$print (UPLIT BYTE (%ASCIC '    bound to: '));
;  3512     3623  3         CASE .sym_ptr [def$b_kind] FROM define_lowest TO define_highest OF
;  3513     3624  3             SET
;  3514     3625  3             [define_address] :
;  3515     3626  4                 BEGIN
;  3516     3627  4                 LOCAL
;  3517     3628  4                     addr_exp_desc : REF dbg$valdesc,
;  3518     3629  4                     string_desc : dbg$stg_desc;
;  3519     3630  4
;  3520     3631  4                 addr_exp_desc = .sym_ptr [def$a_value];
;  3521     3632  4
;  3522     3633  4                 ! Case on kind of descriptor.
;  3523     3634  4                 !
;  3524     3635  4                 CASE .addr_exp_desc [dbg$b_dhdr_type] FROM dbg$k_min_descr_type
;  3525     3636  4                                                       TO dbg$k_max_descr_type
;  3526     3637  4                                                       OF
;  3527     3638  4                     SET
;  3528     3639  4
;  3529     3640  4                     ! Implementation level 3 Primary Descriptors.
;  3530     3641  4                     !
```

```
;  3531          3642  4                [dbg$k_primary_desc] :
;  3532          3643  4                    dbg$print_identifier (.addr_exp_desc);
;  3533          3644  4
;  3534          3645  4                ! Implementation level 3 volatile value descriptors
;  3535          3646  4                !
;  3536          3647  4                [dbg$k_v_value_desc] :
;  3537          3648  5                    BEGIN
;  3538          3649  5                    LOCAL
;  3539          3650  5                        val_desc: REF dbg$valdesc;
;  3540          3651  5
;  3541          3652  5                    ! Turn the volatile value descriptor into an ordinary
;  3542          3653  5                    ! value descriptor and then print it.
;  3543          3654  5                    !
;  3544          3655  5                    IF NOT dbg$ncopy_desc (.addr_exp_desc, val_desc,
;  3545          3656  5                                           .message_vect, FALSE)
;  3546          3657  5                    THEN
;  3547          3658  5                        RETURN sts$k_severe;
;  3548          3659  5                    val_desc[dbg$b_dhdr_type] = dbg$k_value_desc;
;  3549          3660  5                    val_desc[dbg$l_value_value0] =
;  3550          3661  5                        .val_desc[dbg$l_value_pointer];
;  3551          3662  5                    val_desc[dbg$l_value_pointer] =
;  3552          3663  5                         val_desc[dbg$l_value_value0];
;  3553          3664  5                    dbg$print_value (.val_desc,
;  3554          3665  5                                     .dbg$gb_radix[dbg$b_radix_output_over],
;  3555          3666  5                        .dbg$gl_sign_flag, false);
;  3556          3667  4                    END;
;  3557          3668  4
;  3558          3669  4                ! We do not expect any other kind of descriptor.
;  3559          3670  4                !
;  3560          3671  4                [INRANGE, OUTRANGE] :
;  3561          3672  4                    $dbg_error ('DBGDEFINE\DBG$DUMP_DEFINE');
;  3562          3673  4
;  3563          3674  4                TES;
;  3564          3675  3            END;
;  3565          3676  3
;  3566          3677  3            [define_command,
;  3567          3678  3             define_parameter,
;  3568          3679  3             define_procedure,
;  3569          3680  3             define_string] :
;  3570          3681  4                BEGIN
;  3571          3682  4                dbg$print (UPLIT BYTE (%ASCIC '"'));
;  3572          3683  4                dbg$print (.sym_ptr [def$a_value]);
;  3573          3684  4                dbg$print (UPLIT BYTE (%ASCIC '"'));
;  3574          3685  3                END;
;  3575          3686  3
;  3576          3687  3            [define_value] :
;  3577          3688  4                BEGIN
;  3578          3689  4                dbg$print_value (.sym_ptr[def$a_value],
;  3579          3690  4                                 .dbg$gb_radix[dbg$b_radix_output_over],
;  3580          3691  4                                 .dbg$gl_sign_flag, false);
;  3581          3692  3                END;
;  3582          3693  3
;  3583          3694  3            [INRANGE, OUTRANGE] :
;  3584          3695  3                $DBG_ERROR('DBGDEFINE\DUMP_ENTRY');
;  3585          3696  3
;  3586          3697  3            TES;
;  3587          3698  3        dbg$newline();
```

```
;  3588          3699  2          END;
;  3589          3700
;  3590          3701  2      ! Print the kind of symbol.
;  3591          3702  2      !
;  3592          3703  2      IF .type_flag
;  3593          3704  2      THEN
;  3594          3705  3          BEGIN
;  3595          3706  3          dbg$print (UPLIT BYTE (%ASCIC '    '));
;  3596          3707  3          CASE .sym_ptr [def$b_kind] FROM define_lowest TO define_highest OF
;  3597          3708  3              SET
;  3598          3709  3              [define_address] :
;  3599          3710  3                  dbg$print (UPLIT BYTE (%ASCIC 'was defined /address'));
;  3600          3711  3              [define_command] :
;  3601          3712  3                  dbg$print (UPLIT BYTE (%ASCIC 'was defined /command'));
;  3602          3713  3              [define_parameter] :
;  3603          3714  3                  dbg$print (UPLIT BYTE (%ASCIC 'parameter to DEBUG command procedure'));
;  3604          3715  3              [define_procedure] :
;  3605          3716  3                  dbg$print (UPLIT BYTE (%ASCIC 'was defined /procedure'));
;  3606          3717  3              [define_string] :
;  3607          3718  3                  dbg$print (UPLIT BYTE (%ASCIC 'was defined /string'));
;  3608          3719  3              [define_value] :
;  3609          3720  3                  dbg$print (UPLIT BYTE (%ASCIC 'was defined /value'));
;  3610          3721  3              [INRANGE,OUTRANGE]:
;  3611          3722  3                  $DBG_ERROR('DBGDEFINE\DUMP_ENTRY');
;  3612          3723  3              TES;
;  3613          3724  3          dbg$newline();
;  3614          3725  3          END;
;  3615          3726  2
;  3616          3727  2      RETURN sts$k_success;
;  3617          3728  1      END; ! of dump_entry
```

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

                                  20  64  65  6E  69  66  65  64  08  00128 P.ACF:  .ASCII  <8>\defined \                    ;
20  3A  6F  74  20  64  6E  75  6F  62  20  20  20  20  0E  00131 P.ACG:  .ASCII  <14>\    bound to: \                ;
24  47  42  44  5C  45  4E  49  46  45  44  47  42  44  19  00140 P.ACH:  .ASCII  <25>\DBGDEFINE\<92>\DBG$DUMP_DEFINE\    ;
                    45  4E  49  46  45  44  5F  50  4D  55  44  00014F
                                                    22  01  0015A P.ACI:  .ASCII  <1>\"\                                  ;
                                                    22  01  0015C P.ACJ:  .ASCII  <1>\"\                                  ;
50  4D  55  44  5C  45  4E  49  46  45  44  47  42  44  14  0015E P.ACK:  .ASCII  <20>\DBGDEFINE\<92>\DBG$DUMP_ENTRY\      ;
                                    59  52  54  4E  45  5F  00016D
                                        20  20  20  20  04  00173 P.ACL:  .ASCII  <4>\    \                               ;
61  2F  20  64  65  6E  69  66  65  64  20  73  61  77  14  00178 P.ACM:  .ASCII  <20>\was defined /address\               ;
                                    73  73  65  72  64  64  00187
63  2F  20  64  65  6E  69  66  65  64  20  73  61  77  14  0018D P.ACN:  .ASCII  <20>\was defined /command\               ;
                                    64  6E  61  6D  6D  6F  0019C
44  20  6F  74  20  72  65  74  65  6D  61  72  61  70  24  001A2 P.ACO:  .ASCII  \$parameter to DEBUG command procedure\  ;
72  70  20  64  6E  61  6D  6D  6F  63  20  47  55  42  45  001B1
                                    65  72  75  64  65  63  6F  001C0
70  2F  20  64  65  6E  69  66  65  64  20  73  61  77  16  001C7 P.ACP:  .ASCII  <22>\was defined /procedure\             ;
                                65  72  75  64  65  63  6F  72  001D6
73  2F  20  64  65  6E  69  66  65  64  20  73  61  77  13  001DE P.ACQ:  .ASCII  <19>\was defined /string\               ;
                                    67  6E  69  72  74  001ED
76  2F  20  64  65  6E  69  66  65  64  20  73  61  77  12  001F2 P.ACR:  .ASCII  <18>\was defined /value\                ;
                                        65  75  6C  61  00201
```

```
50  4D  55  44  5C  45  4E  49  46  45  44  47  42  44  14  00205 P.ACS:  .ASCII  <20>\DBGDEFINE\<92>\DUMP_ENTRY\          ;
                                    59  52  54  4E  45  5F  00214                                                           ;

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

                                             03FC 00000 DUMP_ENTRY:
                                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9            ; 3585
                        59 00000000G  00  9E 00002                      MOVAB   DBG$GB_RADIX+2, R9                      ;
                        58 00000000G  00  9E 00009                      MOVAB   DBG$GL_SIGN_FLAG, R8                    ;
                        57 00000000G  00  9E 00010                      MOVAB   LIB$SIGNAL, R7                          ;
                        56 00000000G  00  9E 00017                      MOVAB   DBG$NEWLINE, R6                         ;
                        55 00000000G  00  9E 0001E                      MOVAB   DBG$PRINT, R5                           ;
                        54 00000000'  EF  9E 00025                      MOVAB   P.ACF, R4                               ;
                        5E            10  C2 0002C                      SUBL2   #16, SP                                 ;
                                      54  DD 0002F                      PUSHL   R4                                      ; 3613
                        65            01  FB 00031                      CALLS   #1, DBG$PRINT                           ;
                        52       04   AC  D0 00034                      MOVL    SYM_PTR, R2                             ; 3614
                                 08   A2  DD 00038                      PUSHL   8(R2)                                   ;
                        65            01  FB 0003B                      CALLS   #1, DBG$PRINT                           ;
                        66            00  FB 0003E                      CALLS   #0, DBG$NEWLINE                         ; 3615
                        03       08   AC  E8 00041                      BLBS    ADDR_FLAG, 1$                           ; 3619
                                    00BA  31 00045                      BRW     15$                                     ;
                                 09   A4  9F 00048 1$:                  PUSHAB  P.ACG                                   ; 3622
                        65            01  FB 0004B                      CALLS   #1, DBG$PRINT                           ;
             06         01       10   A2  8F 0004E                      CASEB   16(R2), #1, #6                          ; 3623
    0087   0087       0087          0013     00053 2$:                  .WORD   4$-2$,-                                 ;
           000E       0087          009B     0005B                              11$-2$,-                                ;
                                                                                11$-2$,-                                ;
                                                                                11$-2$,-                                ;
                                                                                12$-2$,-                                ;
                                                                                11$-2$,-                                ;
                                                                                3$-2$                                   ;
                                 36   A4  9F 00061 3$:                  PUSHAB  P.ACK                                   ; 3695
                                      27  11 00064                      BRB     7$                                      ;
                        53       0C   A2  D0 00066 4$:                  MOVL    12(R2), ADDR_EXP_DESC                   ; 3631
             0C    78   8F       02   A3  8F 0006A                      CASEB   2(ADDR_EXP_DESC), #120, #12             ; 3635
    001A   001A       002A          001A     00070 5$:                  .WORD   6$-5$,-                                 ;
    001A   001A       001A          001A     00078                              8$-5$,-                                 ;
    0035   001A       001A          001A     00080                              6$-5$,-                                 ;
                                    001A     00088                              6$-5$,-                                 ;
                                                                                6$-5$,-                                 ;
                                                                                6$-5$,-                                 ;
                                                                                6$-5$,-                                 ;
                                                                                6$-5$,-                                 ;
                                                                                6$-5$,-                                 ;
                                                                                6$-5$,-                                 ;
                                                                                6$-5$,-                                 ;
                                                                                9$-5$,-                                 ;
                                                                                6$-5$                                   ;
                                 18   A4  9F 0008A 6$:                  PUSHAB  P.ACH                                   ; 3672
                                      01  DD 0008D 7$:                  PUSHL   #1                                      ;
                        00028362      8F  DD 0008F                      PUSHL   #164706                                 ;
                        67            03  FB 00095                      CALLS   #3, LIB$SIGNAL                          ;
                                      65  11 00098                      BRB     14$                                     ;
                                      53  DD 0009A 8$:                  PUSHL   ADDR_EXP_DESC                           ; 3643
```

```
                00000000G  00          01  FB 0009C        CALLS   #1, DBG$PRINT_IDENTIFIER        ;
                                       5A  11 000A3        BRB     14$                             ;
                                       7E  D4 000A5 9$:    CLRL    -(SP)                           ; 3655
                                  10   AC  DD 000A7        PUSHL   MESSAGE_VECT                    ; 3656
                                  08   AE  9F 000AA        PUSHAB  VAL_DESC                        ; 3655
                                       53  DD 000AD        PUSHL   ADDR_EXP_DESC                   ;
                00000000G  00          04  FB 000AF        CALLS   #4, DBG$NCOPY_DESC              ;
                           04          50  E8 000B6        BLBS    R0, 10$                         ;
                           50          04  D0 000B9        MOVL    #4, R0                          ; 3658
                                           04 000BC        RET                                     ;
                           50          6E  D0 000BD 10$:   MOVL    VAL_DESC, R0                    ; 3659
                     02    A0     7A   8F  90 000C0        MOVB    #122, 2(R0)                     ;
                     20    A0     18   A0  D0 000C5        MOVL    24(R0), 32(R0)                  ; 3661
                     18    A0     20   A0  9E 000CA        MOVAB   32(R0), 24(R0)                  ; 3663
                                       7E  D4 000CF        CLRL    -(SP)                           ; 3664
                                       68  DD 000D1        PUSHL   DBG$GL_SIGN_FLAG                ; 3666
                           7E          69  9A 000D3        MOVZBL  DBG$GB_RADIX+2, -(SP)           ; 3665
                                       50  DD 000D6        PUSHL   R0                              ; 3664
                                       1E  11 000D8        BRB     13$                             ;
                                  32   A4  9F 000DA 11$:   PUSHAB  P.ACI                           ; 3682
                           65          01  FB 000DD        CALLS   #1, DBG$PRINT                   ;
                                  0C   A2  DD 000E0        PUSHL   12(R2)                          ; 3683
                           65          01  FB 000E3        CALLS   #1, DBG$PRINT                   ;
                                  34   A4  9F 000E6        PUSHAB  P.ACJ                           ; 3684
                           65          01  FB 000E9        CALLS   #1, DBG$PRINT                   ;
                                       11  11 000EC        BRB     14$                             ; 3623
                                       7E  D4 000EE 12$:   CLRL    -(SP)                           ; 3689
                                       68  DD 000F0        PUSHL   DBG$GL_SIGN_FLAG                ; 3691
                           7E          69  9A 000F2        MOVZBL  DBG$GB_RADIX+2, -(SP)           ; 3690
                                  0C   A2  DD 000F5        PUSHL   12(R2)                          ; 3689
                00000000G  00          04  FB 000F8 13$:   CALLS   #4, DBG$PRINT_VALUE             ;
                           66          00  FB 000FF 14$:   CALLS   #0, DBG$NEWLINE                 ; 3698
                           4F     0C   AC  E9 00102 15$:   BLBC    TYPE_FLAG, 26$                  ; 3703
                                  4B   A4  9F 00106        PUSHAB  P.ACL                           ; 3706
                           65          01  FB 00109        CALLS   #1, DBG$PRINT                   ;
                     06    01     10   A2  8F 0010C        CASEB   16(R2), #1, #6                  ; 3707
        0034       002E  0024         001F    00111 16$:   .WORD   18$-16$,-                       ;
                   000E  0029         003A    00119                19$-16$,-                       ;
                                                                   21$-16$,-                       ;
                                                                   22$-16$,-                       ;
                                                                   23$-16$,-                       ;
                                                                   20$-16$,-                       ;
                                                                   17$-16$                         ;
                                00DD   C4  9F 0011F 17$:   PUSHAB  P.ACS                           ; 3722
                                       01  DD 00123        PUSHL   #1                              ;
                            00028362   8F  DD 00125        PUSHL   #164706                         ;
                           67          03  FB 0012B        CALLS   #3, LIB$SIGNAL                  ;
                                       22  11 0012E        BRB     25$                             ;
                                  50   A4  9F 00130 18$:   PUSHAB  P.ACM                           ; 3710
                                       1A  11 00133        BRB     24$                             ;
                                  65   A4  9F 00135 19$:   PUSHAB  P.ACN                           ; 3712
                                       15  11 00138        BRB     24$                             ;
                                  7A   A4  9F 0013A 20$:   PUSHAB  P.ACO                           ; 3714
                                       10  11 0013D        BRB     24$                             ;
                                009F   C4  9F 0013F 21$:   PUSHAB  P.ACP                           ; 3716
                                       0A  11 00143        BRB     24$                             ;
                                00B6   C4  9F 00145 22$:   PUSHAB  P.ACQ                           ; 3718
```

```
                         04  11 00149              BRB     24$                                     ;
                   00CA  C4  9F 0014B 23$:          PUSHAB  P.ACR                                   ; 3720
                65       01  FB 0014F 24$:          CALLS   #1, DBG$PRINT                           ;
                66       00  FB 00152 25$:          CALLS   #0, DBG$NEWLINE                         ; 3724
                50       01  D0 00155 26$:          MOVL    #1, R0                                  ; 3727
                             04 00158               RET                                             ; 3728

; Routine Size:  345 bytes,    Routine Base:  DBG$CODE + 1676
```

```
;  3619          3729  1 ROUTINE free_entry (sym_ptr, message_vect) =
;  3620          3730  1 !++
;  3621          3731  1 ! Routine Description
;  3622          3732  1 !
;  3623          3733  1 !       This routine frees up the space occupied by the define entry pointed
;  3624          3734  1 !       to by sym_ptr.
;  3625          3735  1 !
;  3626          3736  1 ! Inputs
;  3627          3737  1 !
;  3628          3738  1 !       sym_ptr -       Points to an entry in the define list
;  3629          3739  1 !       message_vect -  An error message vector
;  3630          3740  1 !
;  3631          3741  1 ! Outputs
;  3632          3742  1 !
;  3633          3743  1 !       A condition code which is one of:
;  3634          3744  1 !       STS$K_SUCCESS -         Success.
;  3635          3745  1 !       STS$K_SEVERE  -         Failure. An error message vector is constructed.
;  3636          3746  1 !--
;  3637          3747  2     BEGIN
;  3638          3748  2
;  3639          3749  2     MAP
;  3640          3750  2         sym_ptr : REF define$entry;
;  3641          3751  2
;  3642          3752  2     LOCAL
;  3643          3753  2         symid_list;                     ! Points to a symid list
;  3644          3754  2
;  3645          3755  2     ! Free up the space taken up by the name.
;  3646          3756  2     !
;  3647          3757  2     dbg$rel_memory (.sym_ptr[def$a_name]);
;  3648          3758  2
;  3649          3759  2     ! Free up the space taken up by the value.
;  3650          3760  2     !
;  3651          3761  2     CASE .sym_ptr[def$b_kind] FROM define_lowest TO define_highest OF
;  3652          3762  2         SET
;  3653          3763  2
;  3654          3764  2         ! Addresses are stored as an address expression descriptor.
;  3655          3765  2         ! We free up the space occupied by the descriptor, and depending
;  3656          3766  2         ! on the kind of descriptor, free up any space occupied by
;  3657          3767  2         ! auxiliary primary descriptors.
;  3658          3768  2         !
;  3659          3769  2         [define_address] :
;  3660          3770  3             BEGIN
;  3661          3771  3             IF dbg$nget_symid (.sym_ptr[def$a_value],
;  3662          3772  3                                symid_list, .message_vect)
;  3663          3773  3             THEN
;  3664          3774  3                 dbg$sta_unlock_symid (.symid_list);
;  3665          3775  3             IF NOT dbg$nfree_desc (.sym_ptr[def$a_value], .message_vect)
;  3666          3776  3             THEN
;  3667          3777  3                 RETURN sts$k_severe;
;  3668          3778  3             END;
;  3669          3779  2
;  3670          3780  2         ! For the four cases below, the value is stored as a counted
;  3671          3781  2         ! string, so we just free up the storage.
;  3672          3782  2         !
;  3673          3783  2         [define_command,
;  3674          3784  2         define_parameter,
;  3675          3785  2         define_procedure,
```

```
;  3676          3786  2          define_string] :
;  3677          3787  2              dbg$rel_memory (.sym_ptr[def$a_value]);
;  3678          3788  2
;  3679          3789  2          ! Values are stored in the form of language-specific
;  3680          3790  2          ! descriptors, so we call the free_desc routine that vectors on
;  3681          3791  2          ! the language and calls the appropriate routine to free up
;  3682          3792  2          ! the descriptor.
;  3683          3793  2          !
;  3684          3794  2          [define_value] :
;  3685          3795  3              BEGIN
;  3686          3796  3              IF dbg$nget_symid (.sym_ptr[def$a_value],
;  3687          3797  3                                 symid_list, .message_vect)
;  3688          3798  3              THEN
;  3689          3799  3                  dbg$sta_unlock_symid (.symid_list);
;  3690          3800  3              IF NOT dbg$rfree_desc (.sym_ptr[def$a_value], .message_vect)
;  3691          3801  3              THEN
;  3692          3802  3                  RETURN sts$k_severe;
;  3693          3803  3              END;
;  3694          3804  2
;  3695          3805  2          [INRANGE,OUTRANGE]:
;  3696          3806  2              $DBG_ERROR('DBGDEFINE\FREE_ENTRY');
;  3697          3807  2
;  3698          3808  2          TES;
;  3699          3809  2
;  3700          3810  2      RETURN sts$k_success;
;  3701          3811  2
;  3702          3812  1      END; ! free_entry


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

45 45 52 46 5C 45 4E 49 46 45 44 47 42 44 14  0021A P.ACT:    .ASCII  <20>\DBGDEFINE\<92>\FREE_ENTRY\                 ;
                        59 52 54 4E 45 5F     00229                                                               ;


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

                                              001C 00000 FREE_ENTRY:
                                                                .WORD   Save R2,R3,R4                               ; 3729
                                  54 00000000G  00   9E 00002       MOVAB   DBG$REL_MEMORY, R4                      ;
                                  53 00000000G  00   9E 00009       MOVAB   DBG$NGET_SYMID, R3                      ;
                                  5E            04   C2 00010       SUBL2   #4, SP                                  ;
                                  52      04    AC   D0 00013       MOVL    SYM_PTR, R2                             ; 3757
                                          08    A2   DD 00017       PUSHL   8(R2)                                   ;
                                  64            01   FB 0001A       CALLS   #1, DBG$REL_MEMORY                      ;
                            06            01    10   A2 8F 0001D    CASEB   16(R2), #1, #6                          ; 3761
                0036        0036          0036       0025  00022 1$:  .WORD   3$-1$,-                                 ;
                            000E          0036       003E  0002A              4$-1$,-                                 ;
                                                                              4$-1$,-                                 ;
                                                                              4$-1$,-                                 ;
                                                                              5$-1$,-                                 ;
                                                                              4$-1$,-                                 ;
                                                                              2$-1$                                   ;
                                  00000000'     EF   9F 00030 2$:    PUSHAB  P.ACT                                   ; 3806
                                                01   DD 00036       PUSHL   #1                                      ;
```

```
                 00028362  8F  DD 00038          PUSHL   #164706                                  ;
00000000G  00              03  FB 0003E          CALLS   #3, LIB$SIGNAL                          ;
                           45  11 00045          BRB     8$                                      ;
                       08  AC  DD 00047 3$:      PUSHL   MESSAGE_VECT                            ; 3772
                       04  AE  9F 0004A          PUSHAB  SYMID_LIST                              ; 3771
                       0C  A2  DD 0004D          PUSHL   12(R2)                                  ;
                   63          03  FB 00050      CALLS   #3, DBG$NGET_SYMID                      ;
                   19          50  E8 00053      BLBS    R0, 6$                                  ;
                               20  11 00056      BRB     7$                                      ; 3775
                       0C  A2  DD 00058 4$:      PUSHL   12(R2)                                  ; 3787
                   64          01  FB 0005B      CALLS   #1, DBG$REL_MEMORY                      ;
                               2C  11 0005E      BRB     8$                                      ;
                       08  AC  DD 00060 5$:      PUSHL   MESSAGE_VECT                            ; 3797
                       04  AE  9F 00063          PUSHAB  SYMID_LIST                              ; 3796
                       0C  A2  DD 00066          PUSHL   12(R2)                                  ;
                   63          03  FB 00069      CALLS   #3, DBG$NGET_SYMID                      ;
                   09          50  E9 0006C      BLBC    R0, 7$                                  ;
                               6E  DD 0006F 6$:      PUSHL   SYMID_LIST                          ; 3799
00000000G  00              01  FB 00071          CALLS   #1, DBG$STA_UNLOCK_SYMID                ;
                       08  AC  DD 00078 7$:      PUSHL   MESSAGE_VECT                            ; 3800
                       0C  A2  DD 0007B          PUSHL   12(R2)                                  ;
00000000G  00              02  FB 0007E          CALLS   #2, DBG$NFREE_DESC                      ;
                   04          50  E8 00085      BLBS    R0, 8$                                  ;
                   50          04  D0 00088      MOVL    #4, R0                                  ; 3802
                               04 0008B          RET                                             ;
                   50          01  D0 0008C 8$:      MOVL    #1, R0                              ; 3810
                               04 0008F          RET                                             ; 3812

; Routine Size:  144 bytes,    Routine Base:  DBG$CODE + 17CF
```

```
; 3704        3813   1 ROUTINE name_match (name1, name2) =
; 3705        3814   1 !++
; 3706        3815   1 ! Functional Description
; 3707        3816   1 !
; 3708        3817   1 !       This routine is used by the SYM_ADD and SYM_FIND routines to
; 3709        3818   1 !       determine whether a pair of DEFINEd names are the same.
; 3710        3819   1 !       This is now a straightforward string match (which could be
; 3711        3820   1 !       done inline), but I made it a subroutine in case the rules
; 3712        3821   1 !       for name matching become more complex, in which case we
; 3713        3822   1 !       will only want to change the code in only one place.
; 3714        3823   1 !
; 3715        3824   1 ! Inputs
; 3716        3825   1 !
; 3717        3826   1 !       name1 - Points to a counted string for the first name.
; 3718        3827   1 !       name2 - Points to a counted string for the second name.
; 3719        3828   1 !
; 3720        3829   1 ! Outputs
; 3721        3830   1 !
; 3722        3831   1 !       Returns TRUE if the names match and FALSE if they don't.
; 3723        3832   1 !--
; 3724        3833   2     BEGIN
; 3725        3834   2
; 3726        3835   2     MAP
; 3727        3836   2         name1: REF VECTOR [,BYTE],
; 3728        3837   2         name2: REF VECTOR [,BYTE];
; 3729        3838   2
; 3730        3839   2     ! Compare lengths
; 3731        3840   2     !
; 3732        3841   2     IF .name1[0] NEQ .name2[0]
; 3733        3842   2     THEN
; 3734        3843   2         RETURN FALSE;
; 3735        3844   2
; 3736        3845   2     ! Compare the actual strings
; 3737        3846   2     !
; 3738        3847   2     IF NOT ch$eql (.name1[0], name1[1], .name2[0], name2[1])
; 3739        3848   2     THEN
; 3740        3849   2         RETURN FALSE;
; 3741        3850   2
; 3742        3851   2     RETURN TRUE;
; 3743        3852   1     END; ! name_match
```

```
                                        000C 00000 NAME_MATCH:
                                                   .WORD   Save R2,R3                       ; 3813
                 51        04   AC   D0 00002      MOVL    NAME1, R1                        ; 3841
                 50        08   AC   D0 00006      MOVL    NAME2, R0                        ;
                 60             61   91 0000A      CMPB    (R1), (R0)                       ;
                                14   12 0000D      BNEQ    1$                               ;
                 53             61   9A 0000F      MOVZBL  (R1), R3                         ; 3847
                 52             60   9A 00012      MOVZBL  (R0), R2                         ;
52      00       01             A1   53 2D 00015   CMPC5   R3, 1(R1), #0, R2, 1(R0)         ;
                           01   A0      0001B                                               ;
                                04   12 0001D      BNEQ    1$                               ;
                 50             01   D0 0001F      MOVL    #1, R0                           ; 3851
```

```
                        04 00022          RET                                                 ;
                    50  D4 00023 1$:      CLRL    R0                                          ; 3852
                        04 00025          RET                                                 ;

; Routine Size:  38 bytes,    Routine Base:  DBG$CODE + 185F
```

```
;  3745          3853  1
;  3746          3854  1 GLOBAL ROUTINE dbg$read_key_info (input_desc, result_desc, message_vect) =
;  3747          3855  1 !++
;  3748          3856  1 ! Functional Description
;  3749          3857  1 !
;  3750          3858  1 !       Reads a string from the DEFINE/KEY command.  The string will be
;  3751          3859  1 !       any length of alphanumeric characters, including $ and _.
;  3752          3860  1 !
;  3753          3861  1 !       The input string descriptor is updated beyond the string that
;  3754          3862  1 !       is read. Space is allocated for a counted string to hold the
;  3755          3863  1 !       result.
;  3756          3864  1 !
;  3757          3865  1 ! Inputs
;  3758          3866  1 !
;  3759          3867  1 !       input_desc -    A pointer to a string descriptor with the input.
;  3760          3868  1 !       result_desc -   A pointer to a string descriptor with the result.
;  3761          3869  1 !       message_vect -  A pointer to an error message vector.
;  3762          3870  1 !
;  3763          3871  1 ! Outputs
;  3764          3872  1 !
;  3765          3873  1 !       Space is allocated for a counted string to hold the result.
;  3766          3874  1 !       A pointer to this counted string is returned in result_addr.
;  3767          3875  1 !       One of the following values is returned:
;  3768          3876  1 !       sts$k_success -         Success code.
;  3769          3877  1 !       sts$k_severe -          If input descriptor has nothing in it to
;  3770          3878  1 !                               return.
;  3771          3879  1 !--
;  3772          3880  2     BEGIN
;  3773          3881  2
;  3774          3882  2     MAP
;  3775          3883  2         input_desc      : REF BLOCK [,BYTE],
;  3776          3884  2         result_desc     : REF BLOCK [,BYTE];
;  3777          3885  2
;  3778          3886  2     LOCAL
;  3779          3887  2         char,                                   ! Holds a character in the input stream
;  3780          3888  2         count,                                  ! Count of characters in the name
;  3781          3889  2         pointer,                                ! Pointer into the input stream
;  3782          3890  2         result  : REF VECTOR[,BYTE];            ! Holds the result
;  3783          3891  2
;  3784          3892  2     ! Check for exhausted input.
;  3785          3893  2     !
;  3786          3894  2     IF .input_desc [dsc$w_length] EQL 0
;  3787          3895  2     THEN
;  3788          3896  3         BEGIN
;  3789          3897  3         .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  3790          3898  3         RETURN sts$k_severe;
;  3791          3899  2         END;
;  3792          3900  2
;  3793          3901  2     ! Read past leading blanks.
;  3794          3902  2     !
;  3795          3903  2     WHILE TRUE DO
;  3796          3904  3         BEGIN
;  3797          3905  3         char = ch$rchar (.input_desc [dsc$a_pointer]);
;  3798          3906  3         IF  .char NEQ dbg$k_blank
;  3799          3907  3         AND .char NEQ dbg$k_tab
;  3800          3908  3         THEN
;  3801          3909  3             EXITLOOP;
```

```
;  3802     3910  3          input_desc [dsc$a_pointer] = ch$plus (.input_desc [dsc$a_pointer], 1);
;  3803     3911  3          input_desc [dsc$w_length] = .input_desc [dsc$w_length] - 1;
;  3804     3912  2          END;
;  3805     3913  2
;  3806     3914  2      ! Check for exhausted input again.
;  3807     3915  2      !
;  3808     3916  2      IF .input_desc [dsc$w_length] EQL 0
;  3809     3917  2      THEN
;  3810     3918  3          BEGIN
;  3811     3919  3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  3812     3920  3          RETURN sts$k_severe;
;  3813     3921  2          END;
;  3814     3922  2
;  3815     3923  2      ! Initialize the count to zero and the pointer to point to the
;  3816     3924  2      ! next character in the input stream.
;  3817     3925  2      !
;  3818     3926  2      count = 0;
;  3819     3927  2      pointer = .input_desc [dsc$a_pointer];
;  3820     3928  2
;  3821     3929  2      ! Read until we hit a character that cannot be part of the string.
;  3822     3930  2      !
;  3823     3931  2      WHILE TRUE DO
;  3824     3932  3          BEGIN
;  3825     3933  3          char = ch$rchar (.pointer);
;  3826     3934  3
;  3827     3935  3          ! Accept alphanumerics, $, and _.
;  3828     3936  3          !
;  3829     3937  3          IF (.char GEQ 'A' AND .char LEQ 'Z') OR
;  3830     3938  3             (.char GEQ 'a' AND .char LEQ 'z') OR
;  3831     3939  3             (.char GEQ '0' AND .char LEQ '9') OR
;  3832     3940  4             (.char EQL '$') OR (.char EQL '_')
;  3833     3941  3          THEN
;  3834     3942  4              BEGIN
;  3835     3943  4              count = .count + 1;
;  3836     3944  4              pointer = ch$plus (.pointer, 1);
;  3837     3945  4              END
;  3838     3946  3          ELSE
;  3839     3947  3              EXITLOOP;
;  3840     3948  2          END;
;  3841     3949  2
;  3842     3950  2      ! Check for no characters read.
;  3843     3951  2      !
;  3844     3952  2      IF .count EQL 0
;  3845     3953  2      THEN
;  3846     3954  3          BEGIN
;  3847     3955  3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
;  3848     3956  3          RETURN sts$k_severe;
;  3849     3957  2          END;
;  3850     3958  2
;  3851     3959  2      ! Allocate space for the result.
;  3852     3960  2      !
;  3853     3961  2      result = dbg$get_tempmem (1+(1+.count)/4);
;  3854     3962  2
;  3855     3963  2      ! Fill in the result, translating lower case to upper case.
;  3856     3964  2      !
;  3857     3965  2      result_desc [dsc$w_length] = .count;
;  3858     3966  2      pointer = result[0];
```

```
;  3859          3967  2     INCR i FROM 1 TO .count DO
;  3860          3968  3         BEGIN
;  3861          3969  3         char = ch$rchar_a (input_desc[dsc$a_pointer]);
;  3862          3970  3         IF .char GEQ 'a' AND .char LEQ 'z'
;  3863          3971  3         THEN
;  3864          3972  3             char = .char - ('a' - 'A');
;  3865          3973  3         ch$wchar_a (.char, pointer);
;  3866          3974  2         END;
;  3867          3975  2     result_desc [dsc$a_pointer] = .result;
;  3868          3976  2
;  3869          3977  2     ! Update the input descriptor to point past the string that was read.
;  3870          3978  2     ! The pointer has already been advanced.
;  3871          3979  2     !
;  3872          3980  2     input_desc [dsc$w_length] = .input_desc [dsc$w_length] - .count;
;  3873          3981  2
;  3874          3982  2     RETURN sts$k_success;
;  3875          3983  2
;  3876          3984  1     END;


                                00FC 00000          .ENTRY  DBG$READ_KEY_INFO, Save R2,R3,R4,R5,R6,R7    ; 3854
                          55    04 AC  D0 00002     MOVL    INPUT_DESC, R5                               ; 3894
                                   65  B5 00006     TSTW    (R5)                                         ;
                                   6A  13 00008     BEQL    10$                                          ;
                          54    04 A5  9E 0000A     MOVAB   4(R5), R4                                    ; 3905
                          53    00 B4  9A 0000E 1$: MOVZBL  @0(R4), CHAR                                 ;
                          20       53  D1 00012     CMPL    CHAR, #32                                    ; 3906
                                   05  13 00015     BEQL    2$                                           ;
                          09       53  D1 00017     CMPL    CHAR, #9                                     ; 3907
                                   06  12 0001A     BNEQ    3$                                           ;
                                   64  D6 0001C 2$:  INCL    (R4)                                         ; 3910
                                   65  B7 0001E     DECW    (R5)                                         ; 3911
                                   EC  11 00020     BRB     1$                                           ; 3903
                                   65  B5 00022 3$:  TSTW    (R5)                                         ; 3916
                                   4E  13 00024     BEQL    10$                                          ;
                                   52  D4 00026     CLRL    COUNT                                        ; 3926
                             57        64  D0 00028     MOVL    (R4), POINTER                            ; 3927
                             53        67  9A 0002B 4$:  MOVZBL  (POINTER), CHAR                         ; 3933
                00000041     8F        53  D1 0002E     CMPL    CHAR, #65                                ; 3937
                                       09  19 00035     BLSS    5$                                       ;
                0000005A     8F        53  D1 00037     CMPL    CHAR, #90                                ;
                                       2A  15 0003E     BLEQ    8$                                       ;
                00000061     8F        53  D1 00040 5$:  CMPL    CHAR, #97                                ; 3938
                                       09  19 00047     BLSS    6$                                       ;
                0000007A     8F        53  D1 00049     CMPL    CHAR, #122                               ;
                                       18  15 00050     BLEQ    8$                                       ;
                             30        53  D1 00052 6$:  CMPL    CHAR, #48                                ; 3939
                                       05  19 00055     BLSS    7$                                       ;
                             39        53  D1 00057     CMPL    CHAR, #57                                ;
                                       0E  15 0005A     BLEQ    8$                                       ;
                             24        53  D1 0005C 7$:  CMPL    CHAR, #36                                ; 3940
                                       09  13 0005F     BEQL    8$                                       ;
                0000005F     8F        53  D1 00061     CMPL    CHAR, #95                                ;
                                       06  12 00068     BNEQ    9$                                       ;
```

```
                                        52  D6 0006A 8$:    INCL    COUNT                           ; 3943
                                        57  D6 0006C        INCL    POINTER                         ; 3944
                                        BB  11 0006E        BRB     4$                              ; 3937
                                        52  D5 00070 9$:    TSTL    COUNT                           ; 3952
                                        15  12 00072        BNEQ    11$                             ;
                              000280D0  8F  DD 00074 10$:   PUSHL   #164048                        ; 3955
                00000000G   00          01  FB 0007A       CALLS   #1, DBG$NMAKE_ARG_VECT          ;
                        0C  BC          50  D0 00081       MOVL    R0, @MESSAGE_VECT               ;
                            50          04  D0 00085       MOVL    #4, R0                          ; 3956
                                        04  00088       RET                                     ;
                            50      01  A2  9E 00089 11$:  MOVAB   1(R2), R0                      ; 3961
                            50          04  C6 0008D       DIVL2   #4, R0                          ;
                                    01  A0  9F 00090       PUSHAB  1(R0)                           ;
                00000000G   00          01  FB 00093       CALLS   #1, DBG$GET_TEMPMEM             ;
                            51      08  AC  D0 0009A       MOVL    RESULT_DESC, R1                 ; 3965
                            61          52  B0 0009E       MOVW    COUNT, (R1)                     ;
                            57          50  D0 000A1       MOVL    RESULT, POINTER                 ; 3966
                                        56  D4 000A4       CLRL    I                               ; 3967
                                        1E  11 000A6       BRB     14$                             ;
                            53      00  B4  9A 000A8 12$:  MOVZBL  @0(R4), CHAR                    ; 3969
                                        64  D6 000AC       INCL    (R4)                            ;
                00000061    8F          53  D1 000AE       CMPL    CHAR, #97                       ; 3970
                                        0C  19 000B5       BLSS    13$                             ;
                0000007A    8F          53  D1 000B7       CMPL    CHAR, #122                      ;
                                        03  14 000BE       BGTR    13$                             ;
                            53          20  C2 000C0       SUBL2   #32, CHAR                       ; 3972
                            87          53  90 000C3 13$:  MOVB    CHAR, (POINTER)+                ; 3973
                        DE              56  F3 000C6 14$:  AOBLEQ  COUNT, I, 12$                   ; 3967
                            04      A1  50  D0 000CA       MOVL    RESULT, 4(R1)                   ; 3975
                                        65  52  A2 000CE       SUBW2   COUNT, (R5)                     ; 3980
                                        50  01  D0 000D1       MOVL    #1, R0                          ; 3982
                                            04 000D4       RET                                     ; 3984

; Routine Size:  213 bytes,    Routine Base:  DBG$CODE + 1885

; 3877          3985  1 END
; 3878          3986  1
; 3879          3987  0 ELUDOM


                                                            .EXTRN  LIB$SIGNAL

;
;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$OWN                        12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$GLOBAL                     12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$PLIT                      559  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                     6490  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
```

```
;                               Library Statistics
;
;                                        --------- Symbols ---------
;       File                             Total    Loaded   Percent      Pages      Processing
;                                                                      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       117         7          97      00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                           418         0         0          31      00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1      386        18         4          22      00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1      150         2         1          12      00:00.3


; Information:  3
; Warnings:     0
; Errors:       0


;                                       COMMAND QUALIFIERS

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

; Size:         6490 code + 583 data bytes
; Run Time:       01:55.7
; Elapsed Time:   06:41.0
; Lines/CPU Min:   2067
; Lexemes/CPU-Min: 12426
; Memory Used:  458 pages
; Compilation Complete
```

0079 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGCVTMAC
LIS

DBGDPC
LIS

DBGDEFINE
LIS