```
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
```

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN   SSSSSSSS  HH      HH    OOOOOO    WW      WW
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN   SSSSSSSS  HH      HH    OOOOOO    WW      WW
DD      DD  BB      BB    GG        NN      NN  SS         HH      HH   OO      OO  WW      WW
DD      DD  BB      BB    GG        NN      NN  SS         HH      HH   OO      OO  WW      WW
DD      DD  BB      BB    GG        NNNN    NN  SS         HH      HH   OO      OO  WW      WW
DD      DD  BB      BB    GG        NNNN    NN  SS         HH      HH   OO      OO  WW      WW
DD      DD  BBBBBBBB      GG        NN  NN  NN   SSSSSS    HHHHHHHHHH   OO      OO  WW      WW
DD      DD  BBBBBBBB      GG        NN  NN  NN   SSSSSS    HHHHHHHHHH   OO      OO  WW      WW
DD      DD  BB      BB    GG  GGGGGG NN   NNNN         SS  HH      HH   OO      OO  WW  WW  WW
DD      DD  BB      BB    GG  GGGGGG NN   NNNN         SS  HH      HH   OO      OO  WW  WW  WW
DD      DD  BB      BB    GG      GG NN    NN          SS  HH      HH   OO      OO  WWWW  WWWW     ....
DD      DD  BB      BB    GG      GG NN    NN          SS  HH      HH   OO      OO  WWWW  WWWW     ....
DDDDDDDD    BBBBBBBB      GGGGGG   NN    NN   SSSSSSSS   HH      HH    OOOOOO    WW      WW      ....
DDDDDDDD    BBBBBBBB      GGGGGG   NN    NN   SSSSSSSS   HH      HH    OOOOOO    WW      WW      ....

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 DBGNSHOW (IDENT = 'V04-000') =
;     2        0002  0
;     3        0003  1 BEGIN
;     4        0004  1
;     5        0005  1 !
;     6        0006  1 !*****************************************************************************
;     7        0007  1 !*                                                                           *
;     8        0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9        0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10        0010  1 !*  ALL RIGHTS RESERVED.                                                      *
;    11        0011  1 !*                                                                           *
;    12        0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13        0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14        0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15        0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16        0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17        0017  1 !*  TRANSFERRED.                                                              *
;    18        0018  1 !*                                                                           *
;    19        0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20        0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21        0021  1 !*  CORPORATION.                                                              *
;    22        0022  1 !*                                                                           *
;    23        0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24        0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    25        0025  1 !*                                                                           *
;    26        0026  1 !*                                                                           *
;    27        0027  1 !*****************************************************************************
;    28        0028  1 !
;    29        0029  1 !
;    30        0030  1 ! MODULE FUNCTION
;    31        0031  1 !       This module contains the ATN parse network and the command execution network
;    32        0032  1 !       to support the SHOW ... command. The parse network constructs a command
;    33        0033  1 !       execution tree consisting of a verb node as the head, and 0 or more noun
;    34        0034  1 !       nodes and adverb nodes. The execution network uses the command execution
;    35        0035  1 !       tree as input and performs the corresponding semantic actions.
;    36        0036  1 !
;    37        0037  1 ! AUTHOR:       David Plummer, CREATION DATE:   3/31/80
;    38        0038  1 !
;    39        0039  1 ! MODIFIED BY:
;    40        0040  1 !
;    41        0041  1 !       Richard Title   16-Sep-81
;    42        0042  1 !       Sid Maxwell      3-Dec-81
;    43        0043  1 !       Ping Sager      19-Feb-82
;    44        0044  1 !       V. Holt         14-May-82
;    45        0045  1 !       Brad Becker     13-Sep-83
;    46        0046  1 !
;    47        0047  1 ! REVISION HISTORY:
;    48        0048  1 !
;    49        0049  1 ! 3.01  16-SEP-81       RT      Implemented SHOW SOURCE
;    50        0050  1 ! 3.02   9-OCT-81       RT      Implemented SHOW MARGINS and SHOW MAX_SOURCE_FILES
;    51        0051  1 ! 3.03  21-Oct-81       RT      Implemented SHOW SEARCH
;    52        0052  1 ! 3.04   3-Dec-81       SRM     Changed SHOW CALLS to check AT_FAULT instead
;    53        0053  1 !                               of AT_BREAK and AT_STEP_END
;    54        0054  1 ! 3B.0  19-Feb-82       PS      Implemented SHOW SYMBOL
;    55        0055  1 !       06-May-82       RT      Implemented SHOW DEFINE
;    56        0056  1 !       07-May-82       RT      Implemented SHOW SYMBOLS/DEFINED
;    57        0057  1 !       07-May-82       RT      Implemented SHOW DEVELOPER
```

```
;    58          0058  1 !        14-May-82       VJH     Added call to DBG$FLUSHBUF, eliminating need to
;    59          0059  1 !                                 initialize local buffer pointers.
;    60          0060  1 !         7-Jun-82       VJH     Removed all references to DBG$FAO_PUT and
;    61          0061  1 !                                 DBG$OUT_PUT, as these are now obsolete.
;    62          0062  1 !        04-Apr-83       RT      Removed all references to VJH, as she
;    63          0063  1 !                                 is now obsolete. (Just kidding, Vicki....)
;    64          0064  1 !        04-Apr-83       RT      Made SHO SYM also show defined symbols
;    65          0065  1 ! 4.0    13-Sep-83       BAB     Implemented SHOW KEY
;    66          0066  1 !
;    67          0067  1
;    68          0068  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    69          0202  1
;    70          0203  1 LIBRARY 'LIB$:DBGGEN.L32';
;    71          0204  1
;    72          0205  1 FORWARD ROUTINE
;    73          0206  1         DBG$NPARSE_SHOW,                        ! Parse network for SHOW command
;    74          0207  1         DBG$NPARSE_SHOW_KEY,                    ! Parse network for SHOW KEY command
;    75          0208  1         DBG$NEXECUTE_SHOW,                      ! Execution network for SHOW command
;    76          0209  1         DBG$NEXECUTE_SHOW_KEY,                  ! Execution network for SHOW KEY command
;    77          0210  1         DBG$NSHOW_MARGINS : NOVALUE,            ! Displays margin settings
;    78          0211  1         DBG$NSHOW_MAX_SOURCE_FILES : NOVALUE,   ! Displays max_source_file setting
;    79          0212  1         DBG$NSHOW_OUTPUT : NOVALUE,             ! Displays output configuration of debugger
;    80          0213  1         DBG$SHOW_RADIX: NOVALUE;                ! Display radix settings
```

```
;    82          0214  1 EXTERNAL ROUTINE
;    83          0215  1     DBG$EVENT_SHOW_CANCEL_SYNTAX,      ! Syntax for SHOW!CANCEL BREAK!TRACE!WATCH
;    84          0216  1     DBG$EVENT_SHOW_CANCEL_SEMANTICS,   ! Semantics for SHOW!CANCEL BREAK!TRACE!WATCH
;    85          0217  1     DBG$DUMP_DEFINE,                   ! Dump define symbol table
;    86          0218  1     DBG$FAO_OUT: NOVALUE,              !
;    87          0219  1     DBG$NGET_TRANS_RADIX,
;    88          0220  1     DBG$SCR_EXECUTE_SHODISP_CMD:NOVALUE,! Execute the SHOW DISPLAY command
;    89          0221  1     DBG$SCR_EXECUTE_SHOSEL_CMD: NOVALUE,! Execute the SHOW SELECT command
;    90          0222  1     DBG$SCR_EXECUTE_SHOWIND_CMD:NOVALUE,! Execute the SHOW WINDOW command
;    91          0223  1     DBG$SCR_PARSE_SHODISP_CMD: NOVALUE, ! Parse the SHOW DISPLAY command
;    92          0224  1     DBG$SCR_PARSE_SHOWIND_CMD: NOVALUE, ! Parse the SHOW WINDOW command
;    93          0225  1     DBG$SHOW_TYPE,                     ! Displays default and override types
;    94          0226  1     DBG$SHOW_MODE,                     ! Displays mode
;    95          0227  1     DBG$SHOW_MODULE,                   ! Outputs the module chain
;    96          0228  1     DBG$SHOW_SEARCH: NOVALUE,          ! Displays search settings
;    97          0229  1     DBG$SHOW_DEFINE: NOVALUE,          ! Displays define setting
;    98          0230  1     DBG$SHOW_STEP,                     ! Outputs user defined step settings
;    99          0231  1     DBG$NPARSE_SHOW_TASK: NOVALUE,     ! Parse the SHOW TASK command
;   100          0232  1     DBG$NEXECUTE_SHOW_TASK: NOVALUE,   ! Execute the SHOW TASK command
;   101          0233  1     DBG$RST_SHOWSCOPE,                 ! Outputs user set scopes
;   102          0234  1     DBG$TRACEBACK,                     ! Shows current runframe nesting
;   103          0235  1     DBG$NNEXT_WORD,                    ! Isolates next word of input for syntax errors
;   104          0236  1     DBG$NSYNTAX_ERROR,                 ! Outputs a syntax error
;   105          0237  1     DBG$NMAKE_ARG_VECT,                ! Constructs a message argument vector
;   106          0238  1     DBG$NSAVE_DECIMAL_INTEGER,         ! Converts input ASCII to integer
;   107          0239  1     DBG$NSAVE_STRING,                  ! Stores a string from input
;   108          0240  1     DBG$GET_TEMPMEM,                   ! Allocates listed dynamic storage
;   109          0241  1     DBG$PRINT: NOVALUE,                ! Formatted ASCII output
;   110          0242  1     DBG$NEWLINE: NOVALUE,              ! Flush the output buffer
;   111          0243  1     DBG$FLUSHBUF: NOVALUE,             ! Initialize new print line
;   112          0244  1     DBG$LANGUAGE,                      ! Returns language setting
;   113          0245  1     DBG$SRC_SHOW_SOURCE: NOVALUE,      ! Implements the SHOW SOURCE command
;   114          0246  1     DBG$NPARSE_SCOPE_LIST,             ! Parses scope list
;   115          0247  1     DBG$STA_SHOWSYMBOL,                ! Execute the SHOW SYMBOL command
;   116          0248  1     DBG$NMATCH,                        ! Counted string matching routine for parsing
;   117          0249  1     DBG$READ_KEY_INFO,                 ! Reads the key-name/state_name for SHOW KEY
;   118          0250  1     STR$COMPARE_EQL,                   ! Returns false if descriptors are equal
;   119          0251  1     SMG$LIST_KEY_DEFS,                 ! Returns all key definitions
;   120          0252  1     SMG$SET_DEFAULT_STATE;             ! Returns the default key state
;   121          0253  1
;   122          0254  1 EXTERNAL
;   123          0255  1     DBG$RUNFRAME: BLOCK [,BYTE],       ! User runframe
;   124          0256  1     DBG$GL_DEVELOPER: BITVECTOR,       ! Set to different developer modes
;   125          0257  1     DBG$GB_KEYPAD_INPUT: BYTE,         ! TRUE if keypad input is enabled
;   126          0258  1     DBG$GB_LANGUAGE: BYTE,             ! Language index
;   127          0259  1     DBG$GB_RADIX: VECTOR[3, BYTE],     ! Radix settings
;   128          0260  1     DBG$GL_LOGFAB: BLOCK [,BYTE],      ! FAB for LOG file
;   129          0261  1     DBG$GL_KEY_TABLE_ID,
;   130          0262  1     DBG$GL_LOGNAM: REF $NAM_DECL,      ! NAM block for LOG file
;   131          0263  1     DBG$GL_CONTEXT: BITVECTOR,         ! Version 2 context vector
;   132          0264  1     DBG$GB_DEF_OUT: VECTOR [,BYTE],    ! Vector for output configuration
;   133          0265  1     DBG$SRC_LEFT_MARGIN,               ! Margin
;   134          0266  1     DBG$SRC_RIGHT_MARGIN,              ! settings.
;   135          0267  1     DBG$SRC_MAX_FILES,                 ! Maximum number of open source
;   136          0268  1                                        !      files (DBGSOURCE)
;   137          0269  1     DBG$SRC_TERM_WIDTH,                ! The current terminal width
;   138          0270  1     DBG$GL_ORIG_COMMAND_PTR,           ! Pointer to original command string
```

```
;  139     0271  1     DBG$GL_UPCASE_COMMAND_PTR: VECTOR[2];
;  140     0272  1                                             ! Pointers to start and end
;  141     0273  1                                             !    of current command string
;  142     0274  1
;  143     0275  1
;  144     0276  1 EXTERNAL LITERAL
;  145     0277  1     SMG$_NOMOREKEYS,
;  146     0278  1     SMG$_KEYNOTDEF;
;  147     0279  1
;  148     0280  1 LITERAL
;  149     0281  1
;  150     0282  1     ! Legal adverb literals for SHOW SYMBOL qualifiers
;  151     0283  1     !
;  152     0284  1     SYMBOL_TYPE         = 1,
;  153     0285  1     SYMBOL_ADDRESS      = 2,
;  154     0286  1     SYMBOL_DIRECT       = 3,
;  155     0287  1     SYMBOL_RST          = 4,
;  156     0288  1     SYMBOL_DST          = 5,
;  157     0289  1     SYMBOL_DEFINED      = 6,
;  158     0290  1
;  159     0291  1
;  160     0292  1     ! Composite verb literals
;  161     0293  1     !
;  162     0294  1     ! Note - you may cause yourself problems if you try to renumber these,
;  163     0295  1     ! because some of these numbers must be the same as the corresponding
;  164     0296  1     ! EVENT$K_SHOW_XXX in DBGLIB.REQ.
;  165     0297  1     !
;  166     0298  1     MIN_SHOW                    = 1,
;  167     0299  1     SHOW_BREAK                  = 1,    ! Also EVENT$K_SHOW_BREAK
;  168     0300  1     SHOW_CALLS                  = 2,
;  169     0301  1     SHOW_CALLS_DIGIT            = 3,
;  170     0302  1     SHOW_LANGUAGE               = 4,
;  171     0303  1     SHOW_LOG                    = 5,
;  172     0304  1     SHOW_MODE                   = 6,
;  173     0305  1     SHOW_MODULE                 = 7,
;  174     0306  1     SHOW_OUTPUT                 = 8,
;  175     0307  1     SHOW_RADIX                  = 28,
;  176     0308  1     SHOW_RADIX_OVERRIDE         = 29,
;  177     0309  1     SHOW_SCOPE                  = 9,
;  178     0310  1     SHOW_STEP                   = 10,
;  179     0311  1     SHOW_TRACE                  = 11,   ! Also EVENT$K_SHOW_TRACE
;  180     0312  1     SHOW_TYPE                   = 12,
;  181     0313  1     SHOW_TYPE_OVERRIDE          = 13,
;  182     0314  1     SHOW_WATCH                  = 14,   ! Also EVENT$K_SHOW_WATCH
;  183     0315  1     SHOW_SOURCE                 = 15,
;  184     0316  1     SHOW_MARGINS                = 16,
;  185     0317  1     SHOW_MAX_SOURCE_FILES       = 17,
;  186     0318  1     SHOW_SEARCH                 = 18,
;  187     0319  1     SHOW_SYMBOL                 = 19,
;  188     0320  1     SHOW_DEFINE                 = 20,
;  189     0321  1     SHOW_SYMBOL_DEFINED         = 21,
;  190     0322  1     SHOW_DEVELOPER              = 22,
;  191     0323  1     SHOW_DISPLAY                = 23,
;  192     0324  1     SHOW_SELECT                 = 24,
;  193     0325  1     SHOW_TERMINAL               = 25,
;  194     0326  1     SHOW_WINDOW                 = 26,
;  195     0327  1     SHOW_KEY                    = 27,
```

```
;  196          0328  1      SHOW_TASK                       = 30,
;  197          0329  1      MAX_SHOW                        = 30;
;  198          0330  1
;  199          0331  1  !
;  200          0332  1  ! MACROS
;  201          0333  1  !
;  202          0334  1  ! The following macro is just an abbreviation for some error-reporting
;  203          0335  1  ! code that occurs repeatedly
;  204          0336  1  !
;  205        M 0337  1  MACRO report_error =
;  206        M 0338  1      BEGIN
;  207        M 0339  1      .message_vect = (
;  208        M 0340  1          IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  209        M 0341  1          THEN
;  210        M 0342  1              dbg$nmake_arg_vect (dbg$_needmore)
;  211        M 0343  1          ELSE
;  212        M 0344  1              dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;  213        M 0345  1      RETURN sts$k_severe;
;  214          0346  1      END %;
;  215          0347  1
;  216          0348  1  !+
;  217          0349  1  ! Definition for the list of state names in the IF_STATE qualifier of the
;  218          0350  1  ! Define/key command.
;  219          0351  1  !-
;  220          0352  1
;  221          0353  1  FIELD
;  222          0354  1      DBG$STATE_NAME_FIELDS =
;  223          0355  1          SET
;  224          0356  1
;  225          0357  1          DBG$L_STATE_NAME_PTR            = [0, 0, 32, 0],        ! Pointer to name descriptor
;  226          0358  1          DBG$L_STATE_NAME_LINK           = [1, 0, 32, 0]         ! Pointer to next state name
;  227          0359  1
;  228          0360  1          TES;
;  229          0361  1
;  230          0362  1  LITERAL
;  231          0363  1          DBG$K_STATE_NAME_SIZE           = 2;                    ! length in long words
;  232          0364  1
;  233          0365  1  MACRO
;  234          0366  1          DBG$STATE_NAME_NODE = BLOCK [DBG$K_STATE_NAME_SIZE] FIELD (DBG$STATE_NAME_FIELDS) %;
```

```
;  236     0367  1 GLOBAL ROUTINE DBG$NPARSE_SHOW (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;  237     0368  1 !
;  238     0369  1 ! FUNCTIONAL DESCRIPTION:
;  239     0370  1 !
;  240     0371  1 !       This routine comprises the ATN parse network for the SHOW command. The
;  241     0372  1 !       network constructs a command execution tree consisting of a linked list
;  242     0373  1 !       of verb, noun, and possibly adverb nodes which the exectuion network accepts
;  243     0374  1 !       as input.
;  244     0375  1 !
;  245     0376  1 ! FORMAL PARAMETERS:
;  246     0377  1 !
;  247     0378  1 !       INPUT_DESC -            Descriptor which points to the command input buffer
;  248     0379  1 !
;  249     0380  1 !       VERB_NODE -             The head node in the command execution tree
;  250     0381  1 !
;  251     0382  1 !       MESSAGE_VECT -          The address of a longword to contain the address
;  252     0383  1 !                               of a message argument vector
;  253     0384  1 !
;  254     0385  1 ! IMPLICIT INPUTS:
;  255     0386  1 !
;  256     0387  1 !       NONE
;  257     0388  1 !
;  258     0389  1 ! IMPLICIT OUTPUTS:
;  259     0390  1 !
;  260     0391  1 !       The command execution (parse) tree is constructed and linked to the verb
;  261     0392  1 !       node.
;  262     0393  1 !
;  263     0394  1 ! ROUTINE VALUE:
;  264     0395  1 !
;  265     0396  1 !       An unsigned integer longword completion code
;  266     0397  1 !
;  267     0398  1 ! COMPLETION CODES:
;  268     0399  1 !
;  269     0400  1 !       STS$K_SEVERE (4) -      Parsing error encountered
;  270     0401  1 !
;  271     0402  1 !       STS$K_SUCCESS (1) -     Successful parse and construction of the command
;  272     0403  1 !                               exectuion tree.
;  273     0404  1 !
;  274     0405  1
;  275     0406  2     BEGIN
;  276     0407  2
;  277     0408  2     MAP
;  278     0409  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to the Verb Node
;  279     0410  2
;  280     0411  2     BIND
;  281     0412  2         DBG$CS_ADDRESS  = UPLIT BYTE (%ASCIC 'ADDRESS'),
;  282     0413  2         DBG$CS_ALL      = UPLIT BYTE (%ASCIC 'ALL'),
;  283     0414  2         DBG$CS_BREAK    = UPLIT BYTE (%ASCIC 'BREAK'),
;  284     0415  2         DBG$CS_CALLS    = UPLIT BYTE (%ASCIC 'CALLS'),
;  285     0416  2         DBG$CS_DEFINE   = UPLIT BYTE (%ASCIC 'DEFINE'),
;  286     0417  2         DBG$CS_DEFINED  = UPLIT BYTE (%ASCIC 'DEFINED'),
;  287     0418  2         DBG$CS_DEVELOPER= UPLIT BYTE (%ASCIC 'DEVELOPER'),
;  288     0419  2         DBG$CS_DIRECT   = UPLIT BYTE (%ASCIC 'DIRECT'),
;  289     0420  2         DBG$CS_DISPLAY  = UPLIT BYTE (%ASCIC 'DISPLAY'),
;  290     0421  2         DBG$CS_DST      = UPLIT BYTE (%ASCIC 'DST'),
;  291     0422  2         DBG$CS_GLOBAL   = UPLIT BYTE (%ASCIC 'GLOBAL'),
;  292     0423  2         DBG$CS_IN       = UPLIT BYTE (%ASCIC 'IN'),
```

```
;  293     0424  2          DBG$CS_INPUT     = UPLIT BYTE (%ASCIC 'INPUT'),
;  294     0425  2          DBG$CS_KEY       = UPLIT BYTE (%ASCIC 'KEY'),
;  295     0426  2          DBG$CS_LANGUAGE  = UPLIT BYTE (%ASCIC 'LANGUAGE'),
;  296     0427  2          DBG$CS_LOCAL     = UPLIT BYTE (%ASCIC 'LOCAL'),
;  297     0428  2          DBG$CS_LOG       = UPLIT BYTE (%ASCIC 'LOG'),
;  298     0429  2          DBG$CS_MARGINS   = UPLIT BYTE (%ASCIC 'MARGINS'),
;  299     0430  2          DBG$CS_MAX_SOURCE_FILES =
;  300     0431  2                             UPLIT BYTE (%ASCIC 'MAX_SOURCE_FILES'),
;  301     0432  2          DBG$CS_MODE      = UPLIT BYTE (%ASCIC 'MODE'),
;  302     0433  2          DBG$CS_MODULE    = UPLIT BYTE (%ASCIC 'MODULE'),
;  303     0434  2          DBG$CS_OUTPUT    = UPLIT BYTE (%ASCIC 'OUTPUT'),
;  304     0435  2          DBG$CS_OVERRIDE  = UPLIT BYTE (%ASCIC 'OVERRIDE'),
;  305     0436  2          DBG$CS_RADIX     = UPLIT BYTE (%ASCIC 'RADIX'),
;  306     0437  2          DBG$CS_RST       = UPLIT BYTE (%ASCIC 'RST'),
;  307     0438  2          DBG$CS_SCOPE     = UPLIT BYTE (%ASCIC 'SCOPE'),
;  308     0439  2          DBG$CS_SEARCH    = UPLIT BYTE (%ASCIC 'SEARCH'),
;  309     0440  2          DBG$CS_SELECT    = UPLIT BYTE (%ASCIC 'SELECT'),
;  310     0441  2          DBG$CS_SOURCE    = UPLIT BYTE (%ASCIC 'SOURCE'),
;  311     0442  2          DBG$CS_STEP      = UPLIT BYTE (%ASCIC 'STEP'),
;  312     0443  2          DBG$CS_SYMBOL    = UPLIT BYTE (%ASCIC 'SYMBOL'),
;  313     0444  2          DBG$CS_TASK      = UPLIT BYTE (%ASCIC 'TASK'),
;  314     0445  2          DBG$CS_TERMINAL  = UPLIT BYTE (%ASCIC 'TERMINAL'),
;  315     0446  2          DBG$CS_TRACE     = UPLIT BYTE (%ASCIC 'TRACE'),
;  316     0447  2          DBG$CS_TYPE      = UPLIT BYTE (%ASCIC 'TYPE'),
;  317     0448  2          DBG$CS_WATCH     = UPLIT BYTE (%ASCIC 'WATCH'),
;  318     0449  2          DBG$CS_WINDOW    = UPLIT BYTE (%ASCIC 'WINDOW'),
;  319     0450  2          DBG$CS_CR        = UPLIT BYTE (1, DBG$K_CAR_RETURN),
;  320     0451  2          DBG$CS_COMMA     = UPLIT BYTE (%ASCIC ','),
;  321     0452  2          DBG$CS_SLASH     = UPLIT BYTE (%ASCIC '/');
;  322     0453  2
;  323     0454  2      LOCAL
;  324     0455  2          ADVERB_NODE: REF DBG$ADVERB_NODE,!
;  325     0456  2          LINK,                            ! Link field to be filled in
;  326     0457  2                                           !      with Adverb Node address
;  327     0458  2          NOUN_NODE: REF DBG$NOUN_NODE,
;  328     0459  2          TMP_BUF1: REF VECTOR[,BYTE],
;  329     0460  2          TMP_BUF2: REF VECTOR[,BYTE];
;  330     0461  2
;  331     0462  2
;  332     0463  2
;  333     0464  2      ! Recognize keyword
;  334     0465  2      !
;  335     0466  2      SELECTONE TRUE OF
;  336     0467  2              SET
;  337     0468  2
;  338     0469  2              [dbg$nmatch (.input_desc, dbg$cs_break, 1)] :
;  339     0470  3                  BEGIN
;  340     0471  3                  VERB_NODE [DBG$B_VERB_COMPOSITE] = EVENT$K_SHOW_BREAK;
;  341     0472  3                  RETURN DBG$EVENT_SHOW_CANCEL_SYNTAX (.INPUT_DESC,
;  342     0473  3                                                       .VERB_NODE,
;  343     0474  3                                                       .MESSAGE_VECT
;  344     0475  3                                                      );
;  345     0476  3                  END;
;  346     0477  2
;  347     0478  2              [dbg$nmatch (.input_desc, dbg$cs_calls, 1)] :
;  348     0479  3                  BEGIN
;  349     0480  3                  verb_node [dbg$b_verb_composite] = show_calls;
```

```
;  350          0481  3                ! May have to accept an integer. In any case, we need a noun node.
;  351          0482  3
;  352          0483  3                noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  353          0484  3                verb_node [dbg$l_verb_object_ptr] = .noun_node;
;  354          0485  3
;  355          0486  3
;  356          0487  3                ! Start out with -1 for the value of the integer. If the input
;  357          0488  3                ! line is not null, then we will try to obtain an integer.
;  358          0489  3                !
;  359          0490  3                IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  360          0491  3                THEN
;  361          0492  3                    BEGIN
;  362          0493  4                    noun_node [dbg$l_noun_value] = -1;
;  363          0494  4                    END
;  364          0495  4                ELSE
;  365          0496  3                    BEGIN
;  366          0497  4                    IF NOT dbg$nsave_decimal_integer (.input_desc,
;  367          0498  4                                                      noun_node [dbg$l_noun_value],
;  368          0499  4                                                      .message_vect)
;  369          0500  4                    THEN
;  370          0501  4                        RETURN sts$k_severe;
;  371          0502  4                    END;
;  372          0503  3                END;
;  373          0504  2
;  374          0505  2
;  375          0506  2            ! Handle the SHOW DEFINE command.
;  376          0507  2            !
;  377          0508  2            [DBG$NMATCH (.INPUT_DESC, DBG$CS_DEFINE, 1)]:
;  378          0509  2                BEGIN
;  379          0510  3                VERB_NODE [DBG$B_VERB_COMPOSITE] = SHOW_DEFINE;
;  380          0511  3                END;
;  381          0512  2
;  382          0513  2
;  383          0514  2            ! Handle the SHOW DEVELOPER command.
;  384          0515  2            !
;  385          0516  2            [DBG$NMATCH (.INPUT_DESC, DBG$CS_DEVELOPER, 9)]:
;  386          0517  2                BEGIN
;  387          0518  3                VERB_NODE [DBG$B_VERB_COMPOSITE] = SHOW_DEVELOPER;
;  388          0519  3                END;
;  389          0520  2
;  390          0521  2
;  391          0522  2            ! Handle the SHOW DISPLAY command.
;  392          0523  2            !
;  393          0524  2            [DBG$NMATCH (.INPUT_DESC, DBG$CS_DISPLAY, 3)]:
;  394          0525  2                BEGIN
;  395          0526  3                VERB_NODE[DBG$B_VERB_COMPOSITE] = SHOW_DISPLAY;
;  396          0527  3                DBG$SCR_PARSE_SHODISP_CMD(.INPUT_DESC, .VERB_NODE);
;  397          0528  3                END;
;  398          0529  2
;  399          0530  2
;  400          0531  2            ! Handle the SHOW KEY command.
;  401          0532  2            !
;  402          0533  2            [dbg$nmatch (.input_desc, dbg$cs_key, 1)]:
;  403          0534  2                BEGIN
;  404          0535  3                RETURN dbg$nparse_show_key(.input_desc, .verb_node, .message_vect);
;  405          0536  3                END;
;  406          0537  2
```

```
;  407     0538  2
;  408     0539  2              ! Handle the SHOW LANGUAGE command.
;  409     0540  2              !
;  410     0541  2              [dbg$nmatch (.input_desc, dbg$cs_language, 2)] :
;  411     0542  3                  BEGIN
;  412     0543  3                  verb_node [dbg$b_verb_composite] = show_language;
;  413     0544  2                  END;
;  414     0545  2
;  415     0546  2              [dbg$nmatch (.input_desc, dbg$cs_log, 2)] :
;  416     0547  3                  BEGIN
;  417     0548  3                  verb_node [dbg$b_verb_composite] = show_log;
;  418     0549  2                  END;
;  419     0550  2
;  420     0551  2              [dbg$nmatch (.input_desc, dbg$cs_margins, 3)] :
;  421     0552  3                  BEGIN
;  422     0553  3                  verb_node [dbg$b_verb_composite] = show_margins;
;  423     0554  2                  END;
;  424     0555  2
;  425     0556  2              [dbg$nmatch (.input_desc, dbg$cs_max_source_files, 3)] :
;  426     0557  3                  BEGIN
;  427     0558  3                  verb_node [dbg$b_verb_composite] = show_max_source_files;
;  428     0559  2                  END;
;  429     0560  2
;  430     0561  2              [dbg$nmatch (.input_desc, dbg$cs_mode, 1)] :
;  431     0562  3                  BEGIN
;  432     0563  3                  verb_node [dbg$b_verb_composite] = show_mode;
;  433     0564  2                  END;
;  434     0565  2
;  435     0566  2              [dbg$nmatch (.input_desc, dbg$cs_module, 4)] :
;  436     0567  3                  BEGIN
;  437     0568  3                  verb_node [dbg$b_verb_composite] = show_module;
;  438     0569  2                  END;
;  439     0570  2
;  440     0571  2              [dbg$nmatch (.input_desc, dbg$cs_output, 1)] :
;  441     0572  3                  BEGIN
;  442     0573  3                  verb_node [dbg$b_verb_composite] = show_output;
;  443     0574  2                  END;
;  444     0575  2
;  445     0576  2              [dbg$nmatch (.input_desc, dbg$cs_radix, 1)]:
;  446     0577  3                  BEGIN
;  447     0578  3                  verb_node[dbg$b_verb_composite] = show_radix;
;  448     0579  3                  WHILE dbg$nmatch (.input_desc, dbg$cs_slash, 1) DO
;  449     0580  3                      SELECTONE TRUE OF
;  450     0581  3                          SET
;  451     0582  3
;  452     0583  3                          ! SHOW RADIX/OVERRIDE. Change the verb composite to
;  453     0584  3                          ! indicate this.
;  454     0585  3                          !
;  455     0586  3                          [dbg$nmatch (.input_desc, dbg$cs_override, 1)]:
;  456     0587  3                              verb_node[dbg$b_verb_composite] = show_radix_override;
;  457     0588  3
;  458     0589  3                          ! Ignore /INPUT and /OUTPUT - we will show both on
;  459     0590  3                          ! a SHOW RADIX command anyway.
;  460     0591  3                          !
;  461     0592  3                          [dbg$nmatch (.input_desc, dbg$cs_input, 1)]:
;  462     0593  3                              0;
;  463     0594  3                          [dbg$nmatch (.input_desc, dbg$cs_output, 2)]:
```

```
;  464     0595  3                0;
;  465     0596  3
;  466     0597  3            ! Any other condition is an error.
;  467     0598  3            !
;  468     0599  3            [dbg$nmatch (.input_desc, dbg$cs_cr, 1)]:
;  469     0600  3                SIGNAL (dbg$_needmore);
;  470     0601  3
;  471     0602  3            [OTHERWISE]:
;  472     0603  4                BEGIN
;  473     0604  4                LOCAL
;  474     0605  4                    cs: REF VECTOR[,BYTE],
;  475     0606  4                    stg_desc: dbg$stg_desc;
;  476     0607  4                cs = dbg$nnext_word(.input_desc);
;  477     0608  4                stg_desc[dsc$b_class] = dsc$k_class_s;
;  478     0609  4                stg_desc[dsc$b_dtype] = dsc$k_dtype_t;
;  479     0610  4                stg_desc[dsc$w_length] = .cs[0];
;  480     0611  4                stg_desc[dsc$a_pointer] = cs[1];
;  481     0612  4                SIGNAL (dbg$_syntax, 1, stg_desc);
;  482     0613  3                END;
;  483     0614  3            TES;
;  484     0615  2        END;
;  485     0616  2
;  486     0617  2    [dbg$nmatch (.input_desc, dbg$cs_scope, 2)] :
;  487     0618  3        BEGIN
;  488     0619  3        verb_node [dbg$b_verb_composite] = show_scope;
;  489     0620  2        END;
;  490     0621  2
;  491     0622  2
;  492     0623  2    ! Handle the SHOW SEARCH command.
;  493     0624  2    !
;  494     0625  2    [DBG$NMATCH (.INPUT_DESC, DBG$CS_SEARCH, 2)] :
;  495     0626  2        VERB_NODE [DBG$B_VERB_COMPOSITE] = SHOW_SEARCH;
;  496     0627  2
;  497     0628  2
;  498     0629  2    ! Handle the SHOW SELECT command.
;  499     0630  2    !
;  500     0631  2    [DBG$NMATCH (.INPUT_DESC, DBG$CS_SELECT, 3)]:
;  501     0632  2        VERB_NODE[DBG$B_VERB_COMPOSITE] = SHOW_SELECT;
;  502     0633  2
;  503     0634  2
;  504     0635  2    ! Handle the SHOW SOURCE command.
;  505     0636  2    !
;  506     0637  2    [DBG$NMATCH (.INPUT_DESC, DBG$CS_SOURCE, 2)] :
;  507     0638  2        VERB_NODE[DBG$B_VERB_COMPOSITE] = SHOW_SOURCE;
;  508     0639  2
;  509     0640  2
;  510     0641  2    [dbg$nmatch (.input_desc, dbg$cs_step, 1)] :
;  511     0642  3        BEGIN
;  512     0643  3        verb_node [dbg$b_verb_composite] = show_step;
;  513     0644  2        END;
;  514     0645  2
;  515     0646  2
;  516     0647  2    ! SHOW SYMBOL[/qulifier...] namespec[,namespec...] [IN scope[,scope...]]
;  517     0648  2    !
;  518     0649  2    [dbg$nmatch (.input_desc, dbg$cs_symbol, 2)]:
;  519     0650  3        BEGIN
;  520     0651  3        LOCAL
```

```
;  521          0652  3          addr_flag,                      ! /ADDRESS
;  522          0653  3          data_list: ref vector[,long],   ! A link list of data
;  523          0654  3                                          ! symbols (A,B,C)
;  524          0655  3          global_flag,                    ! (Only valid for defined
;  525          0656  3                                          ! symbols) - TRUE to show
;  526          0657  3                                          ! globally defined symbols
;  527          0658  3                                          ! (the default); false
;  528          0659  3                                          ! for /LOCAL symbols
;  529          0660  3          type_flag;                      ! /TYPE
;  530          0661  3
;  531          0662  3      ! Initialize flags.
;  532          0663  3      !
;  533          0664  3      global_flag = TRUE;
;  534          0665  3      addr_flag = FALSE;
;  535          0666  3      type_flag = FALSE;
;  536          0667  3
;  537          0668  3
;  538          0669  3      ! Indicate that the command was SHOW SYMBOL.
;  539          0670  3      !
;  540          0671  3      verb_node [dbg$b_verb_composite] = show_symbol;
;  541          0672  3
;  542          0673  3
;  543          0674  3      ! Check to see if there is/are qualifier(s) for this command.
;  544          0675  3      ! If there is/are, then constructs adverb node(s) for it.
;  545          0676  3      ! /DST and /RST are valid, only if the flag is set to DEVELOPER.
;  546          0677  3      !
;  547          0678  3      link = verb_node [dbg$l_verb_adverb_ptr];
;  548          0679  3      WHILE dbg$nmatch (.input_desc, dbg$cs_slash, 1) DO
;  549          0680  4          BEGIN
;  550          0681  4
;  551          0682  4
;  552          0683  4          ! Case on the qualifier.
;  553          0684  4          !
;  554          0685  4          SELECTONE TRUE OF
;  555          0686  4              SET
;  556          0687  4
;  557          0688  4
;  558          0689  4              ! SHOW SYM/ADDRESS.  Construct an Adverb Node and link
;  559          0690  4              ! it in.
;  560          0691  4              !
;  561          0692  4              [dbg$nmatch (.input_desc, dbg$cs_address, 1)]:
;  562          0693  5                  BEGIN
;  563          0694  5                  addr_flag = TRUE;
;  564          0695  5                  adverb_node = dbg$get_tempmem (dbg$k_adverb_node_size);
;  565          0696  5                  .link = .adverb_node;
;  566          0697  5                  link = adverb_node [dbg$l_adverb_link];
;  567          0698  5                  adverb_node [dbg$b_adverb_litera[] = symbol_address;
;  568          0699  4                  END;
;  569          0700  4
;  570          0701  4
;  571          0702  4              ! SHOW SYM/DEFINED.  This qualifier restricts the
;  572          0703  4              ! search to the defined symbols.
;  573          0704  4              !
;  574          0705  4              [dbg$nmatch (.input_desc, dbg$cs_defined, 2)]:
;  575          0706  5                  BEGIN
;  576          0707  5                  verb_node [dbg$b_verb_composite] = show_symbol_defined;
;  577          0708  4                  END;
```

```
;  578          0709  4
;  579          0710  4
;  580          0711  4                    ! SHOW SYM/DIRECT.  Construct an Adverb Node and link it in.
;  581          0712  4                    !
;  582          0713  4                    [dbg$nmatch (.input_desc, dbg$cs_direct, 2)]:
;  583          0714  5                        BEGIN
;  584          0715  5                        adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);
;  585          0716  5                        .link = .adverb_node;
;  586          0717  5                        link = adverb_node [dbg$l_adverb_link];
;  587          0718  5                        adverb_node [dbg$b_adverb_litera[] = symbol_direct;
;  588          0719  4                        END;
;  589          0720  4
;  590          0721  4
;  591          0722  4                    ! SHOW SYM/DEFINE/GLOBAL
;  592          0723  4                    !
;  593          0724  4                    [dbg$nmatch (.input_desc, dbg$cs_global, 1)]:
;  594          0725  5                        BEGIN
;  595          0726  5                        global_flag = TRUE;
;  596          0727  4                        END;
;  597          0728  4
;  598          0729  4
;  599          0730  4                    ! SHOW SYM/DEFINED/LOCAL
;  600          0731  4                    !
;  601          0732  4                    [dbg$nmatch (.input_desc, dbg$cs_local, 1)]:
;  602          0733  5                        BEGIN
;  603          0734  5                        global_flag = FALSE;
;  604          0735  4                        END;
;  605          0736  4
;  606          0737  4
;  607          0738  4                    ! SHOW SYM/TYPE.  Construct an Adverb Node and link
;  608          0739  4                    ! it in.
;  609          0740  4                    !
;  610          0741  4                    [dbg$nmatch (.input_desc, dbg$cs_type, 1)]:
;  611          0742  5                        BEGIN
;  612          0743  5                        type_flag = TRUE;
;  613          0744  5                        adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);
;  614          0745  5                        .link = .adverb_node;
;  615          0746  5                        link = adverb_node [dbg$l_adverb_link];
;  616          0747  5                        adverb_node [dbg$b_adverb_literal] = symbol_type;
;  617          0748  4                        END;
;  618          0749  4
;  619          0750  4
;  620          0751  4                    ! The remaining two are only allowed if developer
;  621          0752  4                    ! bit 0 is set.
;  622          0753  4                    !
;  623          0754  4                    [OTHERWISE]:
;  624          0755  5                        BEGIN
;  625          0756  5                        IF .DBG$GL_DEVELOPER[0]
;  626          0757  5                        THEN
;  627          0758  6                            BEGIN
;  628          0759  6                            SELECTONE TRUE OF
;  629          0760  6                                SET
;  630          0761  6
;  631          0762  6                                ! SHOW SYM/RST.  Construct and Adverb Node
;  632          0763  6                                ! and link it in.
;  633          0764  6                                !
;  634          0765  6                                [dbg$nmatch (.input_desc, dbg$cs_rst, 3)]:
```

```
;  635     0766  7                    BEGIN
;  636     0767  7                    adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);
;  637     0768  7                    .link = .adverb_node;
;  638     0769  7                    link = adverb_node [dbg$l_adverb_link];
;  639     0770  7                    adverb_node [dbg$b_adverb_literal] = symbol_rst;
;  640     0771  6                    END;
;  641     0772  6
;  642     0773  6                ! SHOW SYM/DST.  Construct and Adverb Node
;  643     0774  6                ! and link it in.
;  644     0775  6                !
;  645     0776  6                [dbg$nmatch (.input_desc, dbg$cs_dst, 3)]:
;  646     0777  7                    BEGIN
;  647     0778  7                    adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);
;  648     0779  7                    .link = .adverb_node;
;  649     0780  7                    link = adverb_node [dbg$l_adverb_link];
;  650     0781  7                    adverb_node [dbg$b_adverb_literal] = symbol_dst;
;  651     0782  6                    END;
;  652     0783  6
;  653     0784  6                ! Any other qualifier is an error.
;  654     0785  6                !
;  655     0786  6                [OTHERWISE]:
;  656     0787  7                    BEGIN
;  657     0788  7                    .message_vect =
;  658     0789  8                    (
;  659     0790  8                    IF dbg$nmatch(.input_desc, dbg$cs_cr, 1)
;  660     0791  8                    THEN
;  661     0792  8                        dbg$nmake_arg_vect(dbg$_needmore)
;  662     0793  8                    ELSE
;  663     0794  8                        dbg$nsyntax_error(dbg$nnext_word(.input_desc))
;  664     0795  7                    );
;  665     0796  7
;  666     0797  7                    RETURN sts$k_severe;
;  667     0798  6                    END;
;  668     0799  6
;  669     0800  6                TES;
;  670     0801  6
;  671     0802  6                END             ! Checking for /RST, /DST
;  672     0803  6
;  673     0804  5            ELSE
;  674     0805  6                BEGIN
;  675     0806  6                .message_vect =
;  676     0807  7                (
;  677     0808  7                 IF dbg$nmatch(.input_desc, dbg$cs_cr, 1)
;  678     0809  7                 THEN
;  679     0810  7                    dbg$nmake_arg_vect(dbg$_needmore)
;  680     0811  7                 ELSE
;  681     0812  7                    dbg$nsyntax_error(dbg$nnext_word(.input_desc))
;  682     0813  6                );
;  683     0814  6
;  684     0815  6                RETURN sts$k_severe;
;  685     0816  5                END;
;  686     0817  5
;  687     0818  4            END;
;  688     0819  4
;  689     0820  4        TES;            ! End of selecting qualifiers.
;  690     0821  4
;  691     0822  3    END;                ! End of While Slash Loop.
```

```
;  692          0823  3
;  693          0824
;  694          0825                    ! Put a 0 in the last link field in verb_node's adverb_node_ptr
;  695          0826                    ! field or adverb_node's link field.
;  696          0827                    !
;  697          0828                    .link = 0;
;  698          0829
;  699          0830
;  700          0831                    ! If the command was SHOW SYMBOL/DEFINED, then
;  701          0832                    ! there better not have been any qualifiers other than /GLOBAL
;  702          0833                    ! or /LOCAL.
;  703          0834                    !
;  704          0835                    IF .verb_node [dbg$l_verb_adverb_ptr] NEQ 0
;  705          0836                    THEN
;  706          0837                        IF .verb_node [dbg$b_verb_composite] EQL show_symbol_defined
;  707          0838                        THEN
;  708          0839  4                         BEGIN
;  709          0840  4                         .message_vect = dbg$nmake_arg_vect (dbg$_incomqual);
;  710          0841  4                         RETURN sts$k_severe;
;  711          0842  4                         END;
;  712          0843
;  713          0844                    ! Construct the noun node for pointers to symbol name and
;  714          0845                    ! scope list.
;  715          0846                    !
;  716          0847                    noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  717          0848                    verb_node [dbg$l_verb_object_ptr] = .noun_node;
;  718          0849                    link = noun_node [dbg$l_noun_value];
;  719          0850                    WHILE TRUE DO
;  720          0851  4                     BEGIN
;  721          0852  4                     data_list = dbg$get_tempmem(3);
;  722          0853  4                     .link = .data_list;
;  723          0854  4                     link = data_list[0];
;  724          0855  4
;  725          0856  4
;  726          0857  4                     ! For language C, we do some fancy footwork to
;  727          0858  4                     ! make sure we preserve the original casing of
;  728          0859  4                     ! the identifier (since casing is significant
;  729          0860  4                     ! in C).
;  730          0861  4                     !
;  731          0862  4                     IF .dbg$gb_language EQL dbg$k_c
;  732          0863  4                     THEN
;  733          0864  5                         BEGIN
;  734          0865  5                         MAP
;  735          0866  5                             input_desc: REF dbg$stg_desc;
;  736          0867  5                         LOCAL
;  737          0868  5                             length,
;  738          0869  5                             new_pointer: REF VECTOR [,BYTE],
;  739          0870  5                                          ! Pointer to orig. command input
;  740          0871  5                             pointer,     ! Pointer into input string
;  741          0872  5                             stg_desc: dbg$stg_desc, ! String descriptor
;  742          0873  5                             temp_ptr;
;  743          0874  5
;  744          0875  5
;  745          0876  5                             ! First check for no more input.
;  746          0877  5                             !
;  747          0878  5                             IF dbg$nmatch(.input_desc, dbg$cs_cr, 1)
;  748          0879  5                             THEN
```

```
;  749          0880  5                SIGNAL(dbg$_needmore);
;  750          0881  5
;  751          0882  5            pointer = .input_desc[dsc$a_pointer];
;  752          0883  5            IF (.pointer LSS .dbg$gl_upcase_command_ptr[0]) OR
;  753          0884  6               (.pointer GTR .dbg$gl_upcase_command_ptr[1])
;  754          0885  5            THEN
;  755          0886  5                $DBG_ERROR('DBGNSHOW\DBG$NPARSE_SHOW 10');
;  756          0887  5
;  757          0888  5            ! If we might be looking at %LABEL then don't
;  758          0889  5            ! go back to original case.
;  759          0890  5            !
;  760          0891  5            length = .input_desc[dsc$w_length];
;  761          0892  5            IF CH$EQL(6, .pointer, 6, UPLIT BYTE('%LABEL'))
;  762          0893  5            THEN
;  763          0894  5                new_pointer = .pointer
;  764          0895  5            ELSE
;  765          0896  6                BEGIN
;  766          0897  6
;  767          0898  6                ! We unfortunately have to allocate memory
;  768          0899  6                ! and copy strings in order to stuff a
;  769          0900  6                ! trailing carriage return at the end.
;  770          0901  6                !
;  771          0902  6                new_pointer = dbg$get_tempmem((.length+3)/4);
;  772          0903  6                temp_ptr = (.pointer - .dbg$gl_upcase_command_ptr[0]) +
;  773          0904  6                           .dbg$gl_orig_command_ptr;
;  774          0905  6                CH$MOVE (.length, .temp_ptr, .new_pointer);
;  775          0906  6                new_pointer[.length-1] = dbg$k_car_return;
;  776          0907  5                END;
;  777          0908  5
;  778          0909  5            ! Fill in the string descriptor.
;  779          0910  5            !
;  780          0911  5            stg_desc[dsc$b_class] = dsc$k_class_s;
;  781          0912  5            stg_desc[dsc$b_dtype] = dsc$k_dtype_t;
;  782          0913  5            stg_desc[dsc$w_length] = .length;
;  783          0914  5            stg_desc[dsc$a_pointer] = .new_pointer;
;  784          0915  5            stg_desc[dsc$l_pos] = 0;
;  785          0916  5
;  786          0917  5            ! Pick up the symbol name.
;  787          0918  5            !
;  788          0919  5            IF NOT dbg$nsave_string( stg_desc, data_list[1],
;  789          0920  5                                     .message_vect)
;  790          0921  5            THEN
;  791          0922  5                RETURN sts$k_severe;
;  792          0923  5
;  793          0924  5            ! Update the input descriptor.
;  794          0925  5            !
;  795          0926  5            input_desc[dsc$w_length] = .input_desc[dsc$w_length] -
;  796          0927  5                (.length - .stg_desc[dsc$w_length]);
;  797          0928  5            input_desc[dsc$a_pointer] = .input_desc[dsc$a_pointer] +
;  798          0929  5                (.length - .stg_desc[dsc$w_length]);
;  799          0930  5            END
;  800          0931  5
;  801          0932  5        ! All other languages besides C ...
;  802          0933  5        !
;  803          0934  4        ELSE
;  804          0935  4
;  805          0936  4            ! Pick up the symbol name.
```

```
;  806     0937  4                    !
;  807     0938  4                    IF NOT dbg$nsave_string(.input_desc, data_list[1],
;  808     0939  4                                            .message_vect)
;  809     0940  4                    THEN
;  810     0941  4                        RETURN sts$k_severe;
;  811     0942  4
;  812     0943  4
;  813     0944  4                    ! For SHOW SYM/DEFINED, fill in the adjective field in the
;  814     0945  4                    ! noun node with an encoding of the flags.
;  815     0946  4                    !
;  816     0947  4                    noun_node [dbg$l_adjective_ptr] = (.addr_flag * 4)+
;  817     0948  4                                                      (.global_flag * 2) +
;  818     0949  4                                                      (.type_flag);
;  819     0950  4
;  820     0951  4                    ! For ordinary SHOW SYMBOL, need to fix up %LABEL n and
;  821     0952  4                    ! also pick up the IN clause.
;  822     0953  4                    !
;  823     0954  4                    IF .verb_node [dbg$b_verb_composite] NEQ show_symbol_defined
;  824     0955  4                    THEN
;  825     0956  5                        BEGIN
;  826     0957  5
;  827     0958  5                        ! Check data symbol for special case %label n.
;  828     0959  5                        ! First pick up %LABEL, then pick up n.  Then concatenate
;  829     0960  5                        ! two strings with 1 space in between.
;  830     0961  5                        !
;  831     0962  5                        tmp_buf1 = .data_list[1];
;  832     0963  5                        IF CH$FIND_CH(.tmp_buf1[0], tmp_buf1[1],
;  833     0964  5                                      %C'\')
;  834     0965  5                        THEN
;  835     0966  5                            SIGNAL(dbg$_pathnotacp, 1, tmp_buf1[0]);
;  836     0967  5
;  837     0968  5                        IF CH$EQL(.tmp_buf1[0], tmp_buf1[1], 6, UPLIT BYTE('%LABEL'))
;  838     0969  5                        THEN
;  839     0970  6                            BEGIN
;  840     0971  6                            IF NOT dbg$nsave_string(.input_desc, data_list[1],
;  841     0972  6                                                    .message_vect)
;  842     0973  6                            THEN
;  843     0974  6                                RETURN sts$k_severe;
;  844     0975  6
;  845     0976  6                            tmp_buf2 = .data_list[1];
;  846     0977  6                            data_list[1] = dbg$get_tempmem
;  847     0978  6                                ((.tmp_buf1[0] + .tmp_buf2[0] + 1) / 4 + 1);
;  848     0979  6                            .data_list[1] = .tmp_buf1[0] + .tmp_buf2[0] + 1;
;  849     0980  6                            CH$MOVE(.tmp_buf1[0], tmp_buf1[1],
;  850     0981  6                                .data_list[1] + 1);
;  851     0982  6                            CH$MOVE(1, UPLIT BYTE(' '),
;  852     0983  6                                .data_list[1] + .tmp_buf1[0] + 1);
;  853     0984  6                            CH$MOVE(.tmp_buf2[0], tmp_buf2[1],
;  854     0985  6                                .data_list[1] + .tmp_buf1[0] + 2);
;  855     0986  6
;  856     0987  5                            END;
;  857     0988  5
;  858     0989  4                        END;
;  859     0990  4
;  860     0991  4                    IF NOT dbg$nmatch (.input_desc, dbg$cs_comma, 1)
;  861     0992  4                    THEN
;  862     0993  4                        EXITLOOP;
```

```
;  863     0994  3                END;                    ! End of building the data list.
;  864     0995  3
;  865     0996  3            .link = 0;
;  866     0997  3
;  867     0998  3
;  868     0999  3            ! See if there is keyword IN followed the symbol name.
;  869     1000  3            ! If it is, pick up the scope list.
;  870     1001  3            !
;  871     1002  3            IF .verb_node [dbg$b_verb_composite] NEQ show_symbol_defined
;  872     1003  3            THEN
;  873     1004  4                BEGIN
;  874     1005  4                IF dbg$nmatch (.input_desc, dbg$cs_in, 2)
;  875     1006  4                THEN
;  876     1007  5                    BEGIN
;  877     1008  5                    IF NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)
;  878     1009  5                    THEN
;  879     1010  6                        BEGIN
;  880     1011  6                        IF NOT dbg$nparse_scope_list(.input_desc, noun_node[dbg$l_noun_value2],
;  881     1012  6                                                     .message_vect)
;  882     1013  6                        THEN
;  883     1014  6                            RETURN sts$k_severe;
;  884     1015  6                        END
;  885     1016  6
;  886     1017  5                    ELSE
;  887     1018  6                        BEGIN
;  888     1019  6                        .message_vect = dbg$nmake_arg_vect(dbg$_needmcre);
;  889     1020  6                        RETURN sts$k_severe;
;  890     1021  5                        END;
;  891     1022  4                    END;
;  892     1023  3                END;
;  893     1024  3
;  894     1025  3
;  895     1026  3                ! End of the command buffer. (we hope)
;  896     1027  3                !
;  897     1028  3                IF NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)
;  898     1029  3                THEN
;  899     1030  4                    BEGIN
;  900     1031  4                    .message_vect = dbg$nsyntax_error(dbg$nnext_word(.input_desc));
;  901     1032  4                    RETURN sts$k_severe;
;  902     1033  3                    END;
;  903     1034  3
;  904     1035  2                END;                        ! End of SHOW SYMBOL Parsing.
;  905     1036  2
;  906     1037  2
;  907     1038  2            ! Handle the SHOW TASK command.
;  908     1039  2            !
;  909     1040  2            [IF NOT .DBG$GL_DEVELOPER[0] THEN FALSE ELSE
;  910     1041  2             DBG$NMATCH (.INPUT_DESC, DBG$CS_TASK, 2)]:
;  911     1042  2                BEGIN
;  912     1043  3                VERB_NODE[DBG$B_VERB_COMPOSITE] = SHOW_TASK;
;  913     1044  3                DBG$NPARSE_SHOW_TASK(.INPUT_DESC, .VERB_NODE);
;  914     1045  3                END;
;  915     1046  2
;  916     1047  2
;  917     1048  2            ! Handle the SHOW TERMINAL command.
;  918     1049  2            !
;  919     1050  2            [DBG$NMATCH (.INPUT_DESC, DBG$CS_TERMINAL, 4)]:
```

```
;  920     1051  2                VERB_NODE[DBG$B_VERB_COMPOSITE] = SHOW_TERMINAL;
;  921     1052  2
;  922     1053  2
;  923     1054  2            ! Handle the SHOW TRACE command.
;  924     1055  2            !
;  925     1056  2            [dbg$nmatch (.input_desc, dbg$cs_trace, 1)] :
;  926     1057  3                BEGIN
;  927     1058  3                VERB_NODE [DBG$B_VERB_COMPOSITE] = EVENT$K_SHOW_TRACE;
;  928     1059  3                RETURN DBG$EVENT_SHOW_CANCEL_SYNTAX (.INPUT_DESC,
;  929     1060  3                                                     .VERB_NODE,
;  930     1061  3                                                     .MESSAGE_VECT
;  931     1062  3                                                    );
;  932     1063  3                END;
;  933     1064  2
;  934     1065  2            [dbg$nmatch (.input_desc, dbg$cs_type, 2)] :
;  935     1066  3                BEGIN
;  936     1067  3
;  937     1068  3
;  938     1069  3                ! We may have SHOW TYPE or SHOW TYPE/OVERRIDE.
;  939     1070  3                ! Check for slash
;  940     1071  3                !
;  941     1072  3                IF dbg$nmatch (.input_desc, dbg$cs_slash, 1)
;  942     1073  3                THEN
;  943     1074  4                    BEGIN
;  944     1075  4
;  945     1076  4                    IF NOT dbg$nmatch (.input_desc, dbg$cs_override, 1)
;  946     1077  4                    THEN
;  947     1078  5                        BEGIN
;  948     1079  5                        .message_vect =
;  949     1080  6                        (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  950     1081  6                        THEN
;  951     1082  6                            dbg$nmake_arg_vect (dbg$_needmore)
;  952     1083  6                        ELSE
;  953     1084  5                            dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;  954     1085  5                        RETURN sts$k_severe;
;  955     1086  4                        END;
;  956     1087  4                    verb_node [dbg$b_verb_composite] = show_type_override;
;  957     1088  4                    END
;  958     1089  3                ELSE
;  959     1090  4                    BEGIN
;  960     1091  4                    verb_node [dbg$b_verb_composite] = show_type;
;  961     1092  3                    END;
;  962     1093  2                END;
;  963     1094  2
;  964     1095  2            [dbg$nmatch (.input_desc, dbg$cs_watch, 1)] :
;  965     1096  3                BEGIN
;  966     1097  3                VERB_NODE [DBG$B_VERB_COMPOSITE] = EVENT$K_SHOW_WATCH;
;  967     1098  3                RETURN DBG$EVENT_SHOW_CANCEL_SYNTAX(
;  968     1099  3                        .INPUT_DESC, .VERB_NODE, .MESSAGE_VECT);
;  969     1100  2                END;
;  970     1101  2
;  971     1102  2
;  972     1103  2            ! Handle the SHOW WINDOW command.
;  973     1104  2            !
;  974     1105  2            [DBG$NMATCH (.INPUT_DESC, DBG$CS_WINDOW, 3)]:
;  975     1106  3                BEGIN
;  976     1107  3                VERB_NODE[DBG$B_VERB_COMPOSITE] = SHOW_WINDOW;
```

```
;   977          1108  3              DBG$SCR_PARSE_SHOWIND_CMD(.INPUT_DESC, .VERB_NODE);
;   978          1109  2              END;
;   979          1110  2
;   980          1111  2
;   981          1112  2          ! Any other kind of SHOW command constitutes a syntax error.
;   982          1113  2          !
;   983          1114  2          [OTHERWISE] : ! Parsing error
;   984          1115  3              BEGIN
;   985          1116  3              IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;   986          1117  3              THEN
;   987          1118  3                  .message_vect = dbg$nmake_arg_vect (dbg$_needmore)
;   988          1119  3              ELSE
;   989          1120  3                  .message_vect = dbg$nsyntax_error (dbg$nnext_word (.input_desc));
;   990          1121  3              RETURN sts$k_severe;
;   991          1122  2              END;
;   992          1123  2
;   993          1124  2          TES;
;   994          1125  2
;   995          1126  2      RETURN STS$K_SUCCESS;
;   996          1127  2
;   997          1128  1      END;


                                                        .TITLE  DBGNSHOW
                                                        .IDENT  \V04-000\

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

                                53  53  45  52  44  44  41  07  00000 P.AAA:  .ASCII  <7>\ADDRESS\                    ;
                                                4C  4C  41  03  00008 P.AAB:  .ASCII  <3>\ALL\                      ;
                                                4B  41  45  52  42  05  0000C P.AAC:  .ASCII  <5>\BREAK\             ;
                                                53  4C  4C  41  43  05  00012 P.AAD:  .ASCII  <5>\CALLS\             ;
                                            45  4E  49  46  45  44  06  00018 P.AAE:  .ASCII  <6>\DEFINE\            ;
                                        44  45  4E  49  46  45  44  07  0001F P.AAF:  .ASCII  <7>\DEFINED\           ;
                            52  45  50  4F  4C  45  56  45  44  09  00027 P.AAG:  .ASCII  <9>\DEVELOPER\           ;
                                        54  43  45  52  49  44  06  00031 P.AAH:  .ASCII  <6>\DIRECT\            ;
                                        59  41  4C  50  53  49  44  07  00038 P.AAI:  .ASCII  <7>\DISPLAY\           ;
                                                54  53  44  03  00040 P.AAJ:  .ASCII  <3>\DST\               ;
                                            4C  41  42  4F  4C  47  06  00044 P.AAK:  .ASCII  <6>\GLOBAL\            ;
                                                            4E  49  02  0004B P.AAL:  .ASCII  <2>\IN\                ;
                                                    54  55  50  4E  49  05  0004E P.AAM:  .ASCII  <5>\INPUT\           ;
                                                59  45  4B  03  00054 P.AAN:  .ASCII  <3>\KEY\               ;
                                        45  47  41  55  47  4E  41  4C  08  00058 P.AAO:  .ASCII  <8>\LANGUAGE\          ;
                                                    4C  41  43  4F  4C  05  00061 P.AAP:  .ASCII  <5>\LOCAL\           ;
                                                47  4F  4C  03  00067 P.AAQ:  .ASCII  <3>\LOG\               ;
                                        53  4E  49  47  52  41  4D  07  0006B P.AAR:  .ASCII  <7>\MARGINS\           ;
4C  49  46  5F  45  43  52  55  4F  53  5F  58  41  4D  10  00073 P.AAS:  .ASCII  <16>\MAX_SOURCE_FILES\
                                                        53  45  00082
                                                    45  44  4F  4D  04  00084 P.AAT:  .ASCII  <4>\MODE\              ;
                                            45  4C  55  44  4F  4D  06  00089 P.AAU:  .ASCII  <6>\MODULE\            ;
                                            54  55  50  54  55  4F  06  00090 P.AAV:  .ASCII  <6>\OUTPUT\            ;
                                45  44  49  52  52  45  56  4F  08  00097 P.AAW:  .ASCII  <8>\OVERRIDE\          ;
                                            58  49  44  41  52  05  000A0 P.AAX:  .ASCII  <5>\RADIX\             ;
                                                54  53  52  03  000A6 P.AAY:  .ASCII  <3>\RST\               ;
                                            45  50  4F  43  53  05  000AA P.AAZ:  .ASCII  <5>\SCOPE\             ;
                                        48  43  52  41  45  53  06  000B0 P.ABA:  .ASCII  <6>\SEARCH\            ;
                                        54  43  45  4C  45  53  06  000B7 P.ABB:  .ASCII  <6>\SELECT\            ;
```

```
                                                45 43 52 55 4F 53 06  000BE P.ABC:   .ASCII  <6>\SOURCE\
                                                   50 45 54 53 04  000C5 P.ABD:   .ASCII  <4>\STEP\
                                                4C 4F 42 4D 59 53 06  000CA P.ABE:   .ASCII  <6>\SYMBOL\
                                                      4B 53 41 54 04  000D1 P.ABF:   .ASCII  <4>\TASK\
                                             4C 41 4E 49 4D 52 45 54 08  000D6 P.ABG:   .ASCII  <8>\TERMINAL\
                                                   45 43 41 52 54 05  000DF P.ABH:   .ASCII  <5>\TRACE\
                                                   45 50 59 54 04  000E5 P.ABI:   .ASCII  <4>\TYPE\
                                                48 43 54 41 57 05  000EA P.ABJ:   .ASCII  <5>\WATCH\
                                                57 4F 44 4E 49 57 06  000F0 P.ABK:   .ASCII  <6>\WINDOW\
                                                                  0D 01  000F7 P.ABL:   .BYTE   1, 13
                                                                  2C 01  000F9 P.ABM:   .ASCII  <1>\,\
                                                                  2F 01  000FB P.ABN:   .ASCII  <1>\/\
4E 24 47 42 44 5C 57 4F 48 53 4E 47 42 44 1B  000FD P.ABO:   .ASCII  <27>\DBGNSHOW\<92>\DBG$NPARSE_SHOW 10\
      30 31 20 57 4F 48 53 5F 45 53 52 41 50 0010C
                                                   4C 45 42 41 4C 25  00119 P.ABP:   .ASCII  \%LABEL\
                                                   4C 45 42 41 4C 25  0011F P.ABQ:   .ASCII  \%LABEL\
                                                                  20  00125 P.ABR:   .ASCII  \ \
```

```
DBG$CS_ADDRESS=         P.AAA
DBG$CS_ALL=             P.AAB
DBG$CS_BREAK=           P.AAC
DBG$CS_CALLS=           P.AAD
DBG$CS_DEFINE=          P.AAE
DBG$CS_DEFINED=         P.AAF
DBG$CS_DEVELOPER=       P.AAG
DBG$CS_DIRECT=          P.AAH
DBG$CS_DISPLAY=         P.AAI
DBG$CS_DST=             P.AAJ
DBG$CS_GLOBAL=          P.AAK
DBG$CS_IN=              P.AAL
DBG$CS_INPUT=           P.AAM
DBG$CS_KEY=             P.AAN
DBG$CS_LANGUAGE=        P.AAO
DBG$CS_LOCAL=           P.AAP
DBG$CS_LOG=             P.AAQ
DBG$CS_MARGINS=         P.AAR
DBG$CS_MAX_SOURCE_FILES=
                        P.AAS
DBG$CS_MODE=            P.AAT
DBG$CS_MODULE=          P.AAU
DBG$CS_OUTPUT=          P.AAV
DBG$CS_OVERRIDE=        P.AAW
DBG$CS_RADIX=           P.AAX
DBG$CS_RST=             P.AAY
DBG$CS_SCOPE=           P.AAZ
DBG$CS_SEARCH=          P.ABA
DBG$CS_SELECT=          P.ABB
DBG$CS_SOURCE=          P.ABC
DBG$CS_STEP=            P.ABD
DBG$CS_SYMBOL=          P.ABE
DBG$CS_TASK=            P.ABF
DBG$CS_TERMINAL=        P.ABG
DBG$CS_TRACE=           P.ABH
DBG$CS_TYPE=            P.ABI
DBG$CS_WATCH=           P.ABJ
DBG$CS_WINDOW=          P.ABK
DBG$CS_CR=              P.ABL
```

```
                DBG$CS_COMMA=       P.ABM
                DBG$CS_SLASH=       P.ABN
                        .EXTRN  DBG$EVENT_SHOW_CANCEL_SYNTAX
                        .EXTRN  DBG$EVENT_SHOW_CANCEL_SEMANTICS
                        .EXTRN  DBG$DUMP_DEFINE
                        .EXTRN  DBG$FAO_OUT, DBG$NGET_TRANS_RADIX
                        .EXTRN  DBG$SCR_EXECUTE_SHODISP_CMD
                        .EXTRN  DBG$SCR_EXECUTE_SHOSEL_CMD
                        .EXTRN  DBG$SCR_EXECUTE_SHOWIND_CMD
                        .EXTRN  DBG$SCR_PARSE_SHODISP_CMD
                        .EXTRN  DBG$SCR_PARSE_SHOWIND_CMD
                        .EXTRN  DBG$SHOW_TYPE, DBG$SHOW_MODE
                        .EXTRN  DBG$SHOW_MODULE
                        .EXTRN  DBG$SHOW_SEARCH
                        .EXTRN  DBG$SHOW_DEFINE
                        .EXTRN  DBG$SHOW_STEP, DBG$NPARSE_SHOW_TASK
                        .EXTRN  DBG$NEXECUTE_SHOW_TASK
                        .EXTRN  DBG$RST_SHOWSCOPE
                        .EXTRN  DBG$TRACEBACK, DBG$NNEXT_WORD
                        .EXTRN  DBG$NSYNTAX_ERROR
                        .EXTRN  DBG$NMAKE_ARG_VECT
                        .EXTRN  DBG$NSAVE_DECIMAL_INTEGER
                        .EXTRN  DBG$NSAVE_STRING
                        .EXTRN  DBG$GET_TEMPMEM
                        .EXTRN  DBG$PRINT, DBG$NEWLINE
                        .EXTRN  DBG$FLUSHBUF, DBG$LANGUAGE
                        .EXTRN  DBG$SRC_SHOW_SOURCE
                        .EXTRN  DBG$NPARSE_SCOPE_LIST
                        .EXTRN  DBG$STA_SHOWSYMBOL
                        .EXTRN  DBG$NMATCH, DBG$READ_KEY_INFO
                        .EXTRN  STR$COMPARE_EQL
                        .EXTRN  SMG$LIST_KEY_DEFS
                        .EXTRN  SMG$SET_DEFAULT_STATE
                        .EXTRN  DBG$RUNFRAME, DBG$GL_DEVELOPER
                        .EXTRN  DBG$GB_KEYPAD_INPUT
                        .EXTRN  DBG$GB_LANGUAGE
                        .EXTRN  DBG$GB_RADIX, DBG$GL_LOGFAB
                        .EXTRN  DBG$GL_KEY_TABLE_ID
                        .EXTRN  DBG$GL_LOGNAM, DBG$GL_CONTEXT
                        .EXTRN  DBG$GB_DEF_OUT, DBG$SRC_LEFT_MARGIN
                        .EXTRN  DBG$SRC_RIGHT_MARGIN
                        .EXTRN  DBG$SRC_MAX_FILES
                        .EXTRN  DBG$SRC_TERM_WIDTH
                        .EXTRN  DBG$GL_ORIG_COMMAND_PTR
                        .EXTRN  DBG$GL_UPCASE_COMMAND_PTR
                        .EXTRN  SMG$_NOMOREKEYS
                        .EXTRN  SMG$_KEYNOTDEF

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

                    0FFC 00000          .ENTRY  DBG$NPARSE_SHOW, Save R2,R3,R4,R5,R6,R7,R8,-; 0367
                                                R9,R10,R11
5E                  20  C2 00002       SUBL2   #32, SP                                     ;
                    01  DD 00005       PUSHL   #1                                          ; 0469
   00000000'        EF  9F 00007       PUSHAB  DBG$CS_BREAK                                ;
58          04      AC  D0 0000D       MOVL    INPUT_DESC, R8                              ;
                    58  DD 00011       PUSHL   R8                                          ;
```

```
                        00000000G  00          03  FB 00013          CALLS   #3, DBG$NMATCH                                   ;
                                   01          50  D1 0001A          CMPL    R0, #1                                          ;
                                               09  12 0001D          BNEQ    1$                                              ;
      08  BC            08                 08  01  F0 0001F          INSV    #1, #8, #8, @VERB_NODE                          ; 0471
                                           082D  31 00025          BRW     84$                                             ; 0474
                                               01  DD 00028 1$:      PUSHL   #1                                              ; 0478
                                 00000000'     EF  9F 0002A          PUSHAB  DBG$CS_CALLS                                    ;
                                               58  DD 00030          PUSHL   R8                                              ;
                        00000000G  00          03  FB 00032          CALLS   #3, DBG$NMATCH                                  ;
                                   01          50  D1 00039          CMPL    R0, #1                                          ;
                                               49  12 0003C          BNEQ    3$                                              ;
                                   52      08  AC  D0 0003E          MOVL    VERB_NODE, R2                                   ; 0480
                                01 A2          02  90 00042          MOVB    #2, T(R2)                                       ;
                                               04  DD 00046          PUSHL   #4                                              ; 0484
                        00000000G  00          01  FB 00048          CALLS   #1, DBG$GET_TEMPMEM                             ;
                                08 AE          50  D0 0004F          MOVL    R0, NOUN_NODE                                   ;
                                08 A2      08  AE  D0 00053          MOVL    NOUN_NODE, 8(R2)                                ; 0485
                                               01  DD 00058          PUSHL   #1                                              ; 0491
                                 00000000'     EF  9F 0005A          PUSHAB  DBG$CS_CR                                       ;
                                               58  DD 00060          PUSHL   R8                                              ;
                        00000000G  00          03  FB 00062          CALLS   #3, DBG$NMATCH                                  ;
                                   06          50  E9 00069          BLBC    R0, 2$                                          ;
                                08 BE          01  CE 0006C          MNEGL   #1, @NOUN_NODE                                  ; 0494
                                               79  11 00070          BRB     6$                                              ; 0491
                                           0C  AC  DD 00072 2$:      PUSHL   MESSAGE_VECT                                    ; 0500
                                           0C  AE  DD 00075          PUSHL   NOUN_NODE                                       ; 0499
                                               58  DD 00078          PUSHL   R8                                              ;
                        00000000G  00          03  FB 0007A          CALLS   #3, DBG$NSAVE_DECIMAL_INTEGER                   ;
                                   67          50  E8 00081          BLBS    R0, 6$                                          ;
                                           081C  31 00084          BRW     88$                                             ; 0502
                                               01  DD 00087 3$:      PUSHL   #1                                              ; 0509
                                 00000000'     EF  9F 00089          PUSHAB  DBG$CS_DEFINE                                   ;
                                               58  DD 0008F          PUSHL   R8                                              ;
                        00000000G  00          03  FB 00091          CALLS   #3, DBG$NMATCH                                  ;
                                   01          50  D1 00098          CMPL    R0, #1                                          ;
                                               08  12 0009B          BNEQ    4$                                              ;
      08  BC            08                 08  14  F0 0009D          INSV    #20, #8, #8, @VERB_NODE                         ; 0511
                                               46  11 000A3          BRB     6$                                              ; 0466
                                               09  DD 000A5 4$:      PUSHL   #9                                              ; 0517
                                 00000000'     EF  9F 000A7          PUSHAB  DBG$CS_DEVELOPER                                ;
                                               58  DD 000AD          PUSHL   R8                                              ;
                        00000000G  00          03  FB 000AF          CALLS   #3, DBG$NMATCH                                  ;
                                   01          50  D1 000B6          CMPL    R0, #1                                          ;
                                               08  12 000B9          BNEQ    5$                                              ;
      03  BC            08                 08  16  F0 000BB          INSV    #22, #8, #8, @VERB_NODE                         ; 0519
                                               68  11 000C1          BRB     9$                                              ; 0466
                                               03  DD 000C3 5$:      PUSHL   #3                                              ; 0525
                                 00000000'     EF  9F 000C5          PUSHAB  DBG$CS_DISPLAY                                  ;
                                               58  DD 000CB          PUSHL   R8                                              ;
                        00000000G  00          03  FB 000CD          CALLS   #3, DBG$NMATCH                                  ;
                                   01          50  D1 000D4          CMPL    R0, #1                                          ;
                                               14  12 000D7          BNEQ    7$                                              ;
      08  BC            08                 08  17  F0 000D9          INSV    #23, #8, #8, @VERB_NODE                         ; 0527
                                           08  AC  DD 000DF          PUSHL   VERB_NODE                                       ; 0528
                                               58  DD 000E2          PUSHL   R8                                              ;
                        00000000G  00          02  FB 000E4          CALLS   #2, DBG$SCR_PARSE_SHODISP_CMD                   ;
                                               7A  11 000EB 6$:      BRB     12$                                             ; 0466
```

```
                                                01 DD 000ED 7$:     PUSHL   #1                                      ; 0534
                                00000000'       EF 9F 000EF         PUSHAB  DBG$CS_KEY                              ;
                                                58 DD 000F5         PUSHL   R8                                      ;
                          00000000G  00         03 FB 000F7         CALLS   #3, DBG$NMATCH                          ;
                                     01         50 D1 000FE         CMPL    R0, #1                                  ;
                                                0C 12 00101         BNEQ    8$                                      ;
                                7E        08    AC 7D 00103         MOVQ    VERB_NODE, -(SP)                        ; 0536
                                                58 DD 00107         PUSHL   R8                                      ;
                             0000V  CF          03 FB 00109         CALLS   #3, DBG$NPARSE_SHOW_KEY                 ;
                                                04 0010E            RET                                             ;
                                                02 DD 0010F 8$:     PUSHL   #2                                      ; 0541
                                00000000'       EF 9F 00111         PUSHAB  DBG$CS_LANGUAGE                         ;
                                                58 DD 00117         PUSHL   R8                                      ;
                          00000000G  00         03 FB 00119         CALLS   #3, DBG$NMATCH                          ;
                                     01         50 D1 00120         CMPL    R0, #1                                  ;
                                                08 12 00123         BNEQ    10$                                     ;
    08  BC            08            08         04 F0 00125         INSV    #4, #8, #8, @VERB_NODE                  ; 0543
                                                76 11 0012B 9$:     BRB     15$                                     ; 0466
                                                02 DD 0012D 10$:    PUSHL   #2                                      ; 0546
                                00000000'       EF 9F 0012F         PUSHAB  DBG$CS_LOG                              ;
                                                58 DD 00135         PUSHL   R8                                      ;
                          00000000G  00         03 FB 00137         CALLS   #3, DBG$NMATCH                          ;
                                     01         50 D1 0013E         CMPL    R0, #1                                  ;
                                                08 12 00141         BNEQ    11$                                     ;
    08  BC            08            08         05 F0 00143         INSV    #5, #8, #8, @VERB_NODE                  ; 0548
                                                76 11 00149         BRB     17$                                     ; 0466
                                                03 DD 0014B 11$:    PUSHL   #3                                      ; 0551
                                00000000'       EF 9F 0014D         PUSHAB  DBG$CS_MARGINS                          ;
                                                58 DD 00153         PUSHL   R8                                      ;
                          00000000G  00         03 FB 00155         CALLS   #3, DBG$NMATCH                          ;
                                     01         50 D1 0015C         CMPL    R0, #1                                  ;
                                                08 12 0015F         BNEQ    13$                                     ;
    08  BC            08            08         10 F0 00161         INSV    #16, #8, #8, @VERB_NODE                 ; 0553
                                                76 11 00167 12$:    BRB     19$                                     ; 0466
                                                03 DD 00169 13$:    PUSHL   #3                                      ; 0556
                                00000000'       EF 9F 0016B         PUSHAB  DBG$CS_MAX_SOURCE_FILES                 ;
                                                58 DD 00171         PUSHL   R8                                      ;
                          00000000G  00         03 FB 00173         CALLS   #3, DBG$NMATCH                          ;
                                     01         50 D1 0017A         CMPL    R0, #1                                  ;
                                                08 12 0017D         BNEQ    14$                                     ;
    08  BC            08            08         11 F0 0017F         INSV    #17, #8, #8, @VERB_NODE                 ; 0558
                                                58 11 00185         BRB     19$                                     ; 0466
                                                01 DD 00187 14$:    PUSHL   #1                                      ; 0561
                                00000000'       EF 9F 00189         PUSHAB  DBG$CS_MODE                             ;
                                                58 DD 0018F         PUSHL   R8                                      ;
                          00000000G  00         03 FB 00191         CALLS   #3, DBG$NMATCH                          ;
                                     01         50 D1 00198         CMPL    R0, #1                                  ;
                                                08 12 0019B         BNEQ    16$                                     ;
    08  BC            08            08         06 F0 0019D         INSV    #6, #8, #8, @VERB_NODE                  ; 0563
                                                3A 11 001A3 15$:    BRB     19$                                     ; 0466
                                                04 DD 001A5 16$:    PUSHL   #4                                      ; 0566
                                00000000'       EF 9F 001A7         PUSHAB  DBG$CS_MODULE                           ;
                                                58 DD 001AD         PUSHL   R8                                      ;
                          00000000G  00         03 FB 001AF         CALLS   #3, DBG$NMATCH                          ;
                                     01         50 D1 001B6         CMPL    R0, #1                                  ;
                                                08 12 001B9         BNEQ    18$                                     ;
    08  BC            08            08         07 F0 001BB         INSV    #7, #8, #8, @VERB_NODE                  ; 0568
```

```
                                                 1C  11 001C1 17$:     BRB     19$                                     ; 0466
                                                 01  DD 001C3 18$:     PUSHL   #1                                      ; 0571
                                      00000000'  EF  9F 001C5          PUSHAB  DBG$CS_OUTPUT                           ;
                                                 58  DD 001CB          PUSHL   R8                                      ;
                                   00000000G  00  03  FB 001CD          CALLS   #3, DBG$NMATCH                          ;
                                              01  50  D1 001D4          CMPL    R0, #1                                  ;
                                                  09  12 001D7          BNEQ    20$                                     ;
      08  BC            08            08          08  F0 001D9          INSV    #8, #8, #8, @VERB_NODE                  ; 0573
                                                06C5  31 001DF 19$:     BRW     89$                                     ; 0466
                                                 01  DD 001E2 20$:     PUSHL   #1                                      ; 0576
                                      00000000'  EF  9F 001E4          PUSHAB  DBG$CS_RADIX                            ;
                                                 58  DD 001EA          PUSHL   R8                                      ;
                                   00000000G  00  03  FB 001EC          CALLS   #3, DBG$NMATCH                          ;
                                              01  50  D1 001F3          CMPL    R0, #1                                  ;
                                                  03  13 001F6          BEQL    21$                                     ;
                                                00B5  31 001F8          BRW     26$                                     ;
      08  BC            08            08          1C  F0 001FB 21$:     INSV    #28, #8, #8, @VERB_NODE                 ; 0578
                                                 01  DD 00201 22$:     PUSHL   #1                                      ; 0579
                                      00000000'  EF  9F 00203          PUSHAB  DBG$CS_SLASH                            ;
                                                 58  DD 00209          PUSHL   R8                                      ;
                                   00000000G  00  03  FB 0020B          CALLS   #3, DBG$NMATCH                          ;
                                              CA  50  E9 00212          BLBC    R0, 19$                                 ;
                                                 01  DD 00215          PUSHL   #1                                      ; 0586
                                      00000000'  EF  9F 00217          PUSHAB  DBG$CS_OVERRIDE                         ;
                                                 58  DD 0021D          PUSHL   R8                                      ;
                                   00000000G  00  03  FB 0021F          CALLS   #3, DBG$NMATCH                          ;
                                              01  50  D1 00226          CMPL    R0, #1                                  ;
                                                  08  12 00229          BNEQ    24$                                     ;
      08  BC            08            08          1D  F0 0022B          INSV    #29, #8, #8, @VERB_NODE                 ; 0587
                                                 CE  11 00231 23$:     BRB     22$                                     ;
                                                 01  DD 00233 24$:     PUSHL   #1                                      ; 0592
                                      00000000'  EF  9F 00235          PUSHAB  DBG$CS_INPUT                            ;
                                                 58  DD 0023B          PUSHL   R8                                      ;
                                   00000000G  00  03  FB 0023D          CALLS   #3, DBG$NMATCH                          ;
                                              01  50  D1 00244          CMPL    R0, #1                                  ;
                                                  B8  13 00247          BEQL    22$                                     ;
                                                  02  DD 00249          PUSHL   #2                                      ; 0594
                                      00000000'  EF  9F 0024B          PUSHAB  DBG$CS_OUTPUT                           ;
                                                 58  DD 00251          PUSHL   R8                                      ;
                                   00000000G  00  03  FB 00253          CALLS   #3, DBG$NMATCH                          ;
                                              01  50  D1 0025A          CMPL    R0, #1                                  ;
                                                  A2  13 0025D          BEQL    22$                                     ;
                                                 01  DD 0025F          PUSHL   #1                                      ; 0599
                                      00000000'  EF  9F 00261          PUSHAB  DBG$CS_CR                               ;
                                                 58  DD 00267          PUSHL   R8                                      ;
                                   00000000G  00  03  FB 00269          CALLS   #3, DBG$NMATCH                          ;
                                              01  50  D1 00270          CMPL    R0, #1                                  ;
                                                  0F  12 00273          BNEQ    25$                                     ;
                                      000280D0   8F  DD 00275          PUSHL   #164048                                 ; 0600
                                   00000000G  00  01  FB 0027B          CALLS   #1, LIB$SIGNAL                          ;
                                                  AD  11 00282          BRB     23$                                     ;
                                                 58  DD 00284 25$:     PUSHL   R8                                      ; 0607
                                   00000000G  00  01  FB 00286          CALLS   #1, DBG$NNEXT_WORD                      ;
                                         16   AE      010E   8F  B0 0028D          MOVW    #270, STG_DESC+2                ; 0609
                                         14   AE             60  9B 00293          MOVZBW  (CS), STG_DESC                  ; 0610
                                         18   AE        01   A0  9E 00297          MOVAB   1(R0), STG_DESC+4               ; 0611
                                                     14   AE  9F 0029C          PUSHAB  STG_DESC                        ; 0612
```

```
                                   01  DD 0029F       PUSHL   #1                              :
                         00028238  8F  DD 002A1       PUSHL   #164408                         :
             00000000G  00         03  FB 002A7       CALLS   #3, LIB$SIGNAL                  :
                                   81  11 002AE       BRB     23$                             : 0580
                                   02  DD 002B0 26$:  PUSHL   #2                              : 0617
                         00000000'  EF  9F 002B2       PUSHAB  DBG$CS_SCOPE                    :
                                   58  DD 002B8       PUSHL   R8                              :
             00000000G  00         03  FB 002BA       CALLS   #3, DBG$NMATCH                  :
                        01         50  D1 002C1       CMPL    R0, #1                          :
                                   08  12 002C4       BNEQ    27$                             :
  08   BC        08     08         09  F0 002C6       INSV    #9, #8, #8, @VERB_NODE          : 0619
                                   76  11 002CC       BRB     31$                             : 0466
                                   02  DD 002CE 27$:  PUSHL   #2                              : 0625
                         00000000'  EF  9F 002D0       PUSHAB  DBG$CS_SEARCH                   :
                                   58  DD 002D6       PUSHL   R8                              :
             00000000G  00         03  FB 002D8       CALLS   #3, DBG$NMATCH                  :
                        01         50  D1 002DF       CMPL    R0, #1                          :
                                   08  12 002E2       BNEQ    28$                             :
  08   BC        08     08         12  F0 002E4       INSV    #18, #8, #8, @VERB_NODE         : 0626
                                   58  11 002EA       BRB     31$                             :
                                   03  DD 002EC 28$:  PUSHL   #3                              : 0631
                         00000000'  EF  9F 002EE       PUSHAB  DBG$CS_SELECT                   :
                                   58  DD 002F4       PUSHL   R8                              :
             00000000G  00         03  FB 002F6       CALLS   #3, DBG$NMATCH                  :
                        01         50  D1 002FD       CMPL    R0, #1                          :
                                   08  12 00300       BNEQ    29$                             :
  08   BC        08     08         18  F0 00302       INSV    #24, #8, #8, @VERB_NODE         : 0632
                                   3A  11 00308       BRB     31$                             :
                                   02  DD 0030A 29$:  PUSHL   #2                              : 0637
                         00000000'  EF  9F 0030C       PUSHAB  DBG$CS_SOURCE                   :
                                   58  DD 00312       PUSHL   R8                              :
             00000000G  00         03  FB 00314       CALLS   #3, DBG$NMATCH                  :
                        01         50  D1 0031B       CMPL    R0, #1                          :
                                   08  12 0031E       BNEQ    30$                             :
  08   BC        08     08         0F  F0 00320       INSV    #15, #8, #8, @VERB_NODE         : 0638
                                   1C  11 00326       BRB     31$                             :
                                   01  DD 00328 30$:  PUSHL   #1                              : 0641
                         00000000'  EF  9F 0032A       PUSHAB  DBG$CS_STEP                     :
                                   58  DD 00330       PUSHL   R8                              :
             00000000G  00         03  FB 00332       CALLS   #3, DBG$NMATCH                  :
                        01         50  D1 00339       CMPL    R0, #1                          :
                                   09  12 0033C       BNEQ    32$                             :
  08   BC        08     08         0A  F0 0033E       INSV    #10, #8, #8, @VERB_NODE         : 0643
                                 0560  31 00344 31$:  BRW     89$                             : 0466
                                   02  DD 00347 32$:  PUSHL   #2                              : 0649
                         00000000'  EF  9F 00349       PUSHAB  DBG$CS_SYMBOL                   :
                                   58  DD 0034F       PUSHL   R8                              :
             00000000G  00         03  FB 00351       CALLS   #3, DBG$NMATCH                  :
                        01         50  D1 00358       CMPL    R0, #1                          :
                                   03  13 0035B       BEQL    33$                             :
                                 03FB  31 0035D       BRW     72$                             :
                        52         01  D0 00360 33$:  MOVL    #1, GLOBAL_FLAG                 : 0664
                                   54  7C 00363       CLRQ    ADDR_FLAG                       : 0665
  08   BC        08     08         13  F0 00365       INSV    #19, #8, #8, @VERB_NODE         : 0671
                 50     08  AC     04  C1 0036B       ADDL3   #4, VERB_NODE, R0               : 0678
                        6E         60  9E 00370       MOVAB   (R0), LINK                      :
                                   01  DD 00373 34$:  PUSHL   #1                              : 0679
```

```
                                00000000'  EF  9F 00375          PUSHAB  DBG$CS_SLASH                    ;
                                           58  DD 0037B          PUSHL   R8                              ;
                    00000000G   00         03  FB 0037D          CALLS   #3, DBG$NMATCH                  ;
                                03         50  E8 00384          BLBS    R0, 35$                         ;
                                       014F 31  00387          BRW     49$                             ;
                                           01  DD 0038A 35$:     PUSHL   #1                              ; 0692
                                00000000'  EF  9F 0038C          PUSHAB  DBG$CS_ADDRESS                  ;
                                           58  DD 00392          PUSHL   R8                              ;
                    00000000G   00         03  FB 00394          CALLS   #3, DBG$NMATCH                  ;
                                01         50  D1 0039B          CMPL    R0, #1                          ;
                                           1C  12 0039E          BNEQ    37$                             ;
                                54         01  D0 003A0          MOVL    #1, ADDR_FLAG                   ; 0694
                                           03  DD 003A3          PUSHL   #3                              ; 0695
                    00000000G   00         01  FB 003A5          CALLS   #1, DBG$GET_TEMPMEM             ;
                                53         50  D0 003AC          MOVL    R0, ADVERB_NODE                 ;
                          00    BE         53  D0 003AF          MOVL    ADVERB_NODE, @LINK              ; 0696
                                6E   08    A3  9E 003B3          MOVAB   8(R3), LINK                     ; 0697
                                63         02  90 003B7          MOVB    #2, (ADVERB_NODE)               ; 0698
                                           B7  11 003BA 36$:     BRB     34$                             ; 0685
                                           02  DD 003BC 37$:     PUSHL   #2                              ; 0705
                                00000000'  EF  9F 003BE          PUSHAB  DBG$CS_DEFINED                  ;
                                           58  DD 003C4          PUSHL   R8                              ;
                    00000000G   00         03  FB 003C6          CALLS   #3, DBG$NMATCH                  ;
                                01         50  D1 003CD          CMPL    R0, #1                          ;
                                           08  12 003D0          BNEQ    38$                             ;
      08    BC          08     08         15  F0 003D2          INSV    #21, #8, #8, @VERB_NODE         ; 0707
                                           99  11 003D8          BRB     34$                             ; 0685
                                           02  DD 003DA 38$:     PUSHL   #2                              ; 0713
                                00000000'  EF  9F 003DC          PUSHAB  DBG$CS_DIRECT                   ;
                                           58  DD 003E2          PUSHL   R8                              ;
                    00000000G   00         03  FB 003E4          CALLS   #3, DBG$NMATCH                  ;
                                01         50  D1 003EB          CMPL    R0, #1                          ;
                                           19  12 003EE          BNEQ    39$                             ;
                                           03  DD 003F0          PUSHL   #3                              ; 0715
                    00000000G   00         01  FB 003F2          CALLS   #1, DBG$GET_TEMPMEM             ;
                                53         50  D0 003F9          MOVL    R0, ADVERB_NODE                 ;
                          00    BE         53  D0 003FC          MOVL    ADVERB_NODE, @LINK              ; 0716
                                6E   08    A3  9E 00400          MOVAB   8(R3), LINK                     ; 0717
                                63         03  90 00404          MOVB    #3, (ADVERB_NODE)               ; 0718
                                           65  11 00407          BRB     42$                             ; 0685
                                           01  DD 00409 39$:     PUSHL   #1                              ; 0724
                                00000000'  EF  9F 0040B          PUSHAB  DBG$CS_GLOBAL                   ;
                                           58  DD 00411          PUSHL   R8                              ;
                    00000000G   00         03  FB 00413          CALLS   #3, DBG$NMATCH                  ;
                                01         50  D1 0041A          CMPL    R0, #1                          ;
                                           05  12 0041D          BNEQ    40$                             ;
                                52         01  D0 0041F          MOVL    #1, GLOBAL_FLAG                 ; 0726
                                           96  11 00422          BRB     36$                             ; 0685
                                           01  DD 00424 40$:     PUSHL   #1                              ; 0732
                                00000000'  EF  9F 00426          PUSHAB  DBG$CS_LOCAL                    ;
                                           58  DD 0042C          PUSHL   R8                              ;
                    00000000G   00         03  FB 0042E          CALLS   #3, DBG$NMATCH                  ;
                                01         50  D1 00435          CMPL    R0, #1                          ;
                                           04  12 00438          BNEQ    41$                             ;
                                           52  D4 0043A          CLRL    GLOBAL_FLAG                     ; 0734
                                           69  11 0043C          BRB     46$                             ; 0685
                                           01  DD 0043E 41$:     PUSHL   #1                              ; 0741
```

```
                    00000000'  EF  9F 00440          PUSHAB  DBG$CS_TYPE                                    ;
                               58  DD 00446          PUSHL   R8                                             ;
00000000G  00                  03  FB 00448          CALLS   #3, DBG$NMATCH                                 ;
           01                  50  D1 0044F          CMPL    R0, #1                                         ;
                               1C  12 00452          BNEQ    43$                                            ;
           55                  01  D0 00454          MOVL    #1, TYPE_FLAG                                  ; 0743
                               03  DD 00457          PUSHL   #3                                             ; 0744
00000000G  00                  01  FB 00459          CALLS   #1, DBG$GET_TEMPMEM                            ;
           53                  50  D0 00460          MOVL    R0, ADVERB_NODE                                ;
        00 BE                  53  D0 00463          MOVL    ADVERB_NODE, @LINK                             ; 0745
           6E          08      A3  9E 00467          MOVAB   8(R3), LINK                                    ; 0746
           63                  01  90 0046B          MOVB    #1, (ADVERB_NODE)                              ; 0747
                               66  11 0046E  42$:     BRB     48$                                            ; 0685
           03 00000000G  00    E8  00470  43$:     BLBS    DBG$GL_DEVELOPER, 45$                          ; 0756
                         0392  31  00477  44$:     BRW     79$                                            ;
                               03  DD 0047A  45$:     PUSHL   #3                                             ; 0765
                    00000000'  EF  9F 0047C          PUSHAB  DBG$CS_RST                                     ;
                               58  DD 00482          PUSHL   R8                                             ;
00000000G  00                  03  FB 00484          CALLS   #3, DBG$NMATCH                                 ;
           01                  50  D1 0048B          CMPL    R0, #1                                         ;
                               19  12 0048E          BNEQ    47$                                            ;
                               03  DD 00490          PUSHL   #3                                             ; 0767
00000000G  00                  01  FB 00492          CALLS   #1, DBG$GET_TEMPMEM                            ;
           53                  50  D0 00499          MOVL    R0, ADVERB_NODE                                ;
        00 BE                  53  D0 0049C          MOVL    ADVERB_NODE, @LINK                             ; 0768
           6E          08      A3  9E 004A0          MOVAB   8(R3), LINK                                    ; 0769
           63                  04  90 004A4          MOVB    #4, (ADVERB_NODE)                              ; 0770
                               2D  11 004A7  46$:     BRB     48$                                            ; 0759
                               03  DD 004A9  47$:     PUSHL   #3                                             ; 0776
                    00000000'  EF  9F 004AB          PUSHAB  DBG$CS_DST                                     ;
                               58  DD 004B1          PUSHL   R8                                             ;
                  00000000G  00    03  FB 004B3          CALLS   #3, DBG$NMATCH                                 ;
                             01    50  D1 004BA          CMPL    R0, #1                                         ;
                                   B8  12 004BD          BNEQ    44$                                            ;
                                   03  DD 004BF          PUSHL   #3                                             ; 0778
                  00000000G  00    01  FB 004C1          CALLS   #1, DBG$GET_TEMPMEM                            ;
                             53    50  D0 004C8          MOVL    R0, ADVERB_NODE                                ;
                          00 BE    53  D0 004CB          MOVL    ADVERB_NODE, @LINK                             ; 0779
                             6E    08  A3  9E 004CF          MOVAB   8(R3), LINK                                    ; 0780
                             63        05  90 004D3          MOVB    #5, (ADVERB_NODE)                              ; 0781
                                   FE9A  31  004D6  48$:     BRW     34$                                            ; 0759
                                00 BE    D4  004D9  49$:     CLRL    @LINK                                          ; 0828
                    50          08  AC        04  C1 004DC          ADDL3   #4, VERB_NODE, R0                      ; 0835
                                   60  D5 004E1          TSTL    (R0)                                           ;
                                   18  13 004E3          BEQL    51$                                            ;
15          08  BC                 08  ED 004E5          CMPZV   #8, #8, @VERB_NODE, #21                        ; 0837
                                   10  12 004EB          BNEQ    51$                                            ;
                         00028F10  8F  DD 004ED          PUSHL   #167696                                        ; 0840
                  00000000G  00    01  FB 004F3  50$:     CALLS   #1, DBG$NMAKE_ARG_VECT                         ;
                                03A2  31  004FA          BRW     87$                                            ;
                                   04  DD 004FD  51$:     PUSHL   #4                                             ; 0847
                  00000000G  00    01  FB 004FF          CALLS   #1, DBG$GET_TEMPMEM                            ;
                          08 AE    50  D0 00506          MOVL    R0, NOUN_NODE                                  ;
                    50    08 AC    08  C1 0050A          ADDL3   #8, VERB_NODE, R0                              ; 0848
                             60        08  AE  D0 0050F          MOVL    NOUN_NODE, (R0)                                ;
                             6E        08  AE  D0 00513          MOVL    NOUN_NODE, LINK                                ; 0849
                             5B        0C  AC  D0 00517          MOVL    MESSAGE_VECT, R11                              ; 0920
```

```
                      52                02  C4 0051B         MULL2   #2, R2                                        : 0948
              10      AE          6244      DE 0051E         MOVAL   (R2)[ADDR_FLAG], 16(SP)                       : 0947
              10      AE                55  C0 00523         ADDL2   TYPE_FLAG, 16(SP)                             : 0949
                                        03  DD 00527 52$:    PUSHL   #3                                            : 0852
       00000000G      00                01  FB 00529         CALLS   #1, DBG$GET_TEMPMEM                           :
              0C      AE                50  D0 00530         MOVL    R0, DATA_LIST                                 :
              00      BE         0C     AE  D0 00534         MOVL    DATA_LIST, @LINK                              : 0853
                      6E         0C     AE  D0 00539         MOVL    DATA_LIST, LINK                               : 0854
        50    0C      AE                04  C1 0053D         ADDL3   #4, DATA_LIST, R0                             : 0919
                      5A                60  9E 00542         MOVAB   (R0), R10                                     :
                      07  00000000G     00  91 00545         CMPB    DBG$GB_LANGUAGE, #7                           : 0862
                                        03  13 0054C         BEQL    53$                                           :
                                      00B9  31 0054E         BRW     59$                                           :
                                        01  DD 00551 53$:    PUSHL   #1                                            : 0878
                          00000000'     EF  9F 00553         PUSHAB  DBG$CS_CR                                     :
                                        58  DD 00559         PUSHL   R8                                            :
       00000000G      00                03  FB 0055B         CALLS   #3, DBG$NMATCH                                :
                      0D                50  E9 00562         BLBC    R0, 54$                                       :
                          000280D0      8F  DD 00565         PUSHL   #164048                                       : 0880
       00000000G      00                01  FB 0056B         CALLS   #1, LIB$SIGNAL                                :
                      54         04     A8  D0 00572 54$:    MOVL    4(R8), POINTER                                : 0882
       00000000G      00                54  D1 00576         CMPL    POINTER, DBG$GL_UPCASE_COMMAND_PTR            : 0883
                      09                19  0057D            BLSS    55$                                           :
       00000000G      00                54  D1 0057F         CMPL    POINTER, DBG$GL_UPCASE_COMMAND_PTR+4          : 0884
                      15                15  00586            BLEQ    56$                                           :
                          00000000'     EF  9F 00588 55$:    PUSHAB  P.ABO                                         : 0886
                                        01  DD 0058E         PUSHL   #1                                            :
                          00028362      8F  DD 00590         PUSHL   #164706                                       :
       00000000G      00                03  FB 00596         CALLS   #3, LIB$SIGNAL                                :
                      57                68  3C 0059D 56$:    MOVZWL  (R8), LENGTH                                  : 0891
00000000'     EF      64                06  29 005A0         CMPC3   #6, (POINTER), P.ABP                          : 0892
                                        05  12 005A8         BNEQ    57$                                           :
                      59                54  D0 005AA         MOVL    POINTER, NEW_POINTER                          : 0894
                                        2A  11 005AD         BRB     58$                                           :
                      50         03     A7  9E 005AF 57$:    MOVAB   3(R7), R0                                     : 0902
        7E            50                04  C7 005B3         DIVL3   #4, R0, -(SP)                                 :
       00000000G      00                01  FB 005B7         CALLS   #1, DBG$GET_TEMPMEM                           :
                      59                50  D0 005BE         MOVL    R0, NEW_POINTER                               :
                      54  00000000G     00  C2 005C1         SUBL2   DBG$GL_UPCASE_COMMAND_PTR, R4                 : 0903
        50            54  00000000G     00  C1 005C8         ADDL3   DBG$GL_ORIG_COMMAND_PTR, R4, TEMP_PTR         : 0904
        69            60                57  28 005D0         MOVC3   LENGTH, (TEMP_PTR), (NEW_POINTER)             : 0905
              FF    A749                0D  90 005D4         MOVB    #13, -1(LENGTH)[NEW_POINTER]                  : 0906
              16      AE        010E    8F  B0 005D9 58$:    MOVW    #270, STG_DESC+2                              : 0912
              14      AE                57  B0 005DF         MOVW    LENGTH, STG_DESC                              : 0913
              18      AE                59  D0 005E3         MOVL    NEW_POINTER, STG_DESC+4                       : 0914
                                 1C     AE  D4 005E7         CLRL    STG_DESC+8                                    : 0915
                      7E                5A  7D 005EA         MOVQ    R10, -(SP)                                    : 0919
                                 1C     AE  9F 005ED         PUSHAB  STG_DESC                                      :
       00000000G      00                03  FB 005F0         CALLS   #3, DBG$NSAVE_STRING                          :
                      71                50  E9 005F7         BLBC    R0, 64$                                       :
                      50         14     AE  3C 005FA         MOVZWL  STG_DESC, R0                                  : 0927
                      50                57  C2 005FE         SUBL2   LENGTH, R0                                    :
                      68                50  A0 00601         ADDW2   R0, (R8)                                      :
              04      A8                50  C2 00604         SUBL2   R0, 4(R8)                                     : 0929
                                        0E  11 00608         BRB     60$                                           : 0862
                                0D00    8F  BB 0060A 59$:    PUSHR   #^M<R8,R10,R11>                               : 0938
       00000000G      00                03  FB 0060E         CALLS   #3, DBG$NSAVE_STRING                          :
```

```
                           53       50 E9 00615        BLBC    R0, 64$                               ;
          50          08   AE       04 C1 00618 60$:   ADDL3   #4, NOUN_NODE, R0                     ; 0948
                           60  10   AE D0 0061D        MOVL    16(SP), (R0)
 15       08   BC          08       08 ED 00621        CMPZV   #8, #8, @VERB_NODE, #21               ; 0954
                           03       12 00627           BNEQ    61$
                           0093     31 00629           BRW     66$
                           56       6A D0 0062C 61$:   MOVL    (R10), TMP_BUF1                       ; 0962
                           50       66 9A 0062F        MOVZBL  (TMP_BUF1), R0                        ; 0963
          01   A6          50  5C   8F 3A 00632        LOCC    #92, R0, 1(TMP_BUF1)
                           02       12 00638           BNEQ    62$
                           51       D4 0063A           CLRL    R1
                           11       51 E9 0063C 62$:   BLBC    R1, 63$
                           56       DD 0063F           PUSHL   TMP_BUF1                              ; 0966
                           01       DD 00641           PUSHL   #1
                    00028130        8F DD 00643        PUSHL   #164144
              00000000G    00       03 FB 00649        CALLS   #3, LIB$SIGNAL
                           50       66 9A 00650 63$:   MOVZBL  (TMP_BUF1), R0                        ; 0968
 06            00     01   A6       50 2D 00653        CMPC5   R0, 1(TMP_BUF1), #0, #6, P.ABQ
                    00000000'       EF    00659
                           5F       12 0065E           BNEQ    66$
                           0D00     8F BB 00660        PUSHR   #^M<R8,R10,R11>                       ; 0971
              00000000G    00       03 FB 00664        CALLS   #3, DBG$NSAVE_STRING
                           03       50 E8 0066B 64$:   BLBS    R0, 65$
                           0232     31 0066E           BRW     88$
                      04   AE       6A D0 00671 65$:   MOVL    (R10), TMP_BUF2                       ; 0976
                           50       66 9A 00675        MOVZBL  (TMP_BUF1), R0                        ; 0978
                           51  04   BE 9A 00678        MOVZBL  @TMP_BUF2, R1
                           50       51 C0 0067C        ADDL2   R1, R0
                           52  01   A0 9E 0067F        MOVAB   1(R0), R2
          50               52       04 C7 00683        DIVL3   #4, R2, R0
                               01   A0 9F 00687        PUSHAB  1(R0)
              00000000G    00       01 FB 0068A        CALLS   #1, DBG$GET_TEMPMEM
                           6A       50 D0 00691        MOVL    R0, (R10)
                           57       6A D0 00694        MOVL    (R10), R7                             ; 0979
                           67       52 D0 00697        MOVL    R2, (R7)
                           50       66 9A 0069A        MOVZBL  (TMP_BUF1), R0                        ; 0980
          01   A7     01   A6       50 28 0069D        MOVC3   R0, 1(TMP_BUF1), 1(R7)                ; 0981
                           50       66 9A 006A3        MOVZBL  (TMP_BUF1), R0                        ; 0983
                           50       57 C0 006A6        ADDL2   R7, R0
                      01   A0 00000000'  EF 90 006A9   MOVB    P.ABR, 1(R0)
                           51  04   BE 9A 006B1        MOVZBL  @TMP_BUF2, R1                         ; 0984
                 57   04   AE       01 C1 006B5        ADDL3   #1, TMP_BUF2, R7                      ; 0985
          02   A0          67       51 28 006BA        MOVC3   R1, (R7), 2(R0)
                                    01 DD 006BF 66$:   PUSHL   #1                                    ; 0991
                    00000000'       EF 9F 006C1        PUSHAB  DBG$CS_COMMA
                           58       DD 006C7           PUSHL   R8
              00000000G    00       03 FB 006C9        CALLS   #3, DBG$NMATCH
                           03       50 E9 006D0        BLBC    R0, 67$
                           FE51     31 006D3           BRW     52$
                               00   BE D4 006D6 67$:   CLRL    @LINK                                 ; 0996
 15       08   BC          08       08 ED 006D9        CMPZV   #8, #8, @VERB_NODE, #21               ; 1002
                           4E       13 006DF           BEQL    69$
                           02       DD 006E1           PUSHL   #2                                    ; 1005
                    00000000'       EF 9F 006E3        PUSHAB  DBG$CS_IN
                           58       DD 006E9           PUSHL   R8
              00000000G    00       03 FB 006EB        CALLS   #3, DBG$NMATCH
                           3A       50 E9 006F2        BLBC    R0, 69$                               ;
```

```
                                    01   DD 006F5          PUSHL   #1                                          ; 1008
                          00000000' EF   9F 006F7          PUSHAB  DBG$CS_CR
                                    58   DD 006FD          PUSHL   R8
                00000000G   00      03   FB 006FF          CALLS   #3, DBG$NMATCH
                            17      50   E8 00706          BLBS    R0, 68$
                                    5B   DD 00709          PUSHL   R11                                         ; 1012
          50        0C    AE      0C   C1 0070B          ADDL3   #12, NOUN_NODE, R0                          ; 1011
                                    50   DD 00710          PUSHL   R0
                                    58   DD 00712          PUSHL   R8
                00000000G   00      03   FB 00714          CALLS   #3, DBG$NPARSE_SCOPE_LIST
                            11      50   E8 0071B          BLBS    R0, 69$
                                    38   11 0071E          BRB     71$                                         ; 1014
                          000280D0  8F   DD 00720 68$:      PUSHL   #164048                                     ; 1019
                00000000G   00      01   FB 00726          CALLS   #1, DBG$NMAKE_ARG_VECT
                                    26   11 0072D          BRB     70$
                                    01   DD 0072F 69$:      PUSHL   #1                                          ; 1028
                          00000000' EF   9F 00731          PUSHAB  DBG$CS_CR
                                    58   DD 00737          PUSHL   R8
                00000000G   00      03   FB 00739          CALLS   #3, DBG$NMATCH
                            69      50   E8 00740          BLBS    R0, 76$
                                    58   DD 00743          PUSHL   R8                                          ; 1031
                00000000G   00      01   FB 00745          CALLS   #1, DBG$NNEXT_WORD
                                    50   DD 0074C          PUSHL   R0
                00000000G   00      01   FB 0074E          CALLS   #1, DBG$NSYNTAX_ERROR
                            6B      50   D0 00755 70$:      MOVL    R0, (R11)
                                0148 31   00758 71$:      BRW     88$                                         ; 1032
                            04 00000000G 00  E8 0075B 72$:  BLBS    DBG$GL_DEVELOPER, 73$                       ; 1040
                                    50   D4 00762          CLRL    R0
                                    11   11 00764          BRB     74$
                                    02   DD 00766 73$:      PUSHL   #2                                          ; 1041
                          00000000' EF   9F 00768          PUSHAB  DBG$CS_TASK
                                    58   DD 0076E          PUSHL   R8
                00000000G   00      03   FB 00770          CALLS   #3, DBG$NMATCH
                            01      50   D1 00777 74$:      CMPL    R0, #1                                      ; 1040
                                    14   12 0077A          BNEQ    75$
    08    BC        08      08      1E   F0 0077C          INSV    #30, #8, #8, @VERB_NODE                     ; 1043
                                08  AC   DD 00782          PUSHL   VERB_NODE                                   ; 1044
                                    58   DD 00785          PUSHL   R8
                00000000G   00      02   FB 00787          CALLS   #2, DBG$NPARSE_SHOW_TASK
                                    1C   11 0078E          BRB     76$                                         ; 0466
                                    04   DD 00790 75$:      PUSHL   #4                                          ; 1050
                          00000000' EF   9F 00792          PUSHAB  DBG$CS_TERMINAL
                                    58   DD 00798          PUSHL   R8
                00000000G   00      03   FB 0079A          CALLS   #3, DBG$NMATCH
                            01      50   D1 007A1          CMPL    R0, #1
                                    09   12 007A4          BNEQ    77$
    08    BC        08      08      19   F0 007A6          INSV    #25, #8, #8, @VERB_NODE                     ; 1051
                                0080 31   007AC 76$:      BRW     81$
                                    01   DD 007AF 77$:      PUSHL   #1                                          ; 1056
                          00000000' EF   9F 007B1          PUSHAB  DBG$CS_TRACE
                                    58   DD 007B7          PUSHL   R8
                00000000G   00      03   FB 007B9          CALLS   #3, DBG$NMATCH
                            01      50   D1 007C0          CMPL    R0, #1
                                    09   12 007C3          BNEQ    78$
    08    BC        08      08      0B   F0 007C5          INSV    #11, #8, #8, @VERB_NODE                     ; 1058
                                0087 31   007CB          BRW     84$                                         ; 1061
                                    02   DD 007CE 78$:      PUSHL   #2                                          ; 1065
```

```
                                00000000'  EF  9F 007D0          PUSHAB  DBG$CS_TYPE                                          ;
                                           58  DD 007D6          PUSHL   R8                                                    ;
                    00000000G   00         03  FB 007D8          CALLS   #3, DBG$NMATCH                                        ;
                                01         50  D1 007DF          CMPL    R0, #1                                                ;
                                           55  12 007E2          BNEQ    83$                                                   ;
                                           01  DD 007E4          PUSHL   #1                                                    ; 1072
                                00000000'  EF  9F 007E6          PUSHAB  DBG$CS_SLASH                                          ;
                                           58  DD 007EC          PUSHL   R8                                                    ;
                    00000000G   00         03  FB 007EE          CALLS   #3, DBG$NMATCH                                        ;
                                39         50  E9 007F5          BLBC    R0, 82$                                               ;
                                           01  DD 007F8          PUSHL   #1                                                    ; 1076
                                00000000'  EF  9F 007FA          PUSHAB  DBG$CS_OVERRIDE                                       ;
                                           58  DD 00800          PUSHL   R8                                                    ;
                    00000000G   00         03  FB 00802          CALLS   #3, DBG$NMATCH                                        ;
                                1D         50  E8 00809          BLBS    R0, 80$                                               ;
                                           01  DD 0080C  79$:     PUSHL   #1                                                    ; 1080
                                00000000'  EF  9F 0080E          PUSHAB  DBG$CS_CR                                             ;
                                           58  DD 00814          PUSHL   R8                                                    ;
                    00000000G   00         03  FB 00816          CALLS   #3, DBG$NMATCH                                        ;
                                6D         50  E9 0081D          BLBC    R0, 86$                                               ;
                                000280D0   8F  DD 00820          PUSHL   #164048                                               ; 1082
                                      FCCA 31  00826          BRW     50$                                                   ;
    08  BC          08        08         0D  F0 00829  80$:     INSV    #13, #8, #8, @VERB_NODE                               ; 1087
                                           76  11 0082F  81$:     BRB     89$                                                   ; 1072
    08  BC          08        08         0C  F0 00831  82$:     INSV    #12, #8, #8, @VERB_NODE                               ; 1091
                                           6E  11 00837          BRB     89$                                                   ; 0466
                                           01  DD 00839  83$:     PUSHL   #1                                                    ; 1095
                                00000000'  EF  9F 0083B          PUSHAB  DBG$CS_WATCH                                          ;
                                           58  DD 00841          PUSHL   R8                                                    ;
                    00000000G   00         03  FB 00843          CALLS   #3, DBG$NMATCH                                        ;
                                01         50  D1 0084A          CMPL    R0, #1                                                ;
                                           14  12 0084D          BNEQ    85$                                                   ;
    08  BC          08        08         0E  F0 0084F          INSV    #14, #8, #8, @VERB_NODE                               ; 1097
                                7E      08 AC  7D 00855  84$:     MOVQ    VERB_NODE, -(SP)                                      ; 1099
                                           58  DD 00859          PUSHL   R8                                                    ;
                    00000000G   00         03  FB 0085B          CALLS   #3, DBG$EVENT_SHOW_CANCEL_SYNTAX                      ;
                                           04  00862          RET                                                           ; 1098
                                           03  DD 00863  85$:     PUSHL   #3                                                    ; 1105
                                00000000'  EF  9F 00865          PUSHAB  DBG$CS_WINDOW                                         ;
                                           58  DD 0086B          PUSHL   R8                                                    ;
                    00000000G   00         03  FB 0086D          CALLS   #3, DBG$NMATCH                                        ;
                                01         50  D1 00874          CMPL    R0, #1                                                ;
                                           93  12 00877          BNEQ    79$                                                   ;
    08  BC          08        08         1A  F0 00879          INSV    #26, #8, #8, @VERB_NODE                               ; 1107
                                      08   AC  DD 0087F          PUSHL   VERB_NODE                                             ; 1108
                                           58  DD 00882          PUSHL   R8                                                    ;
                    00000000G   00         02  FB 00884          CALLS   #2, DBG$SCR_PARSE_SHOWIND_CMD                         ;
                                           1A  11 0088B          BRB     89$                                                   ; 0466
                                           58  DD 0088D  86$:     PUSHL   R8                                                    ; 1120
                    00000000G   00         01  FB 0088F          CALLS   #1, DBG$NNEXT_WORD                                    ;
                                           50  DD 00896          PUSHL   R0                                                    ;
                    00000000G   00         01  FB 00898          CALLS   #1, DBG$NSYNTAX_ERROR                                 ;
                          0C    BC         50  D0 0089F  87$:     MOVL    R0, @MESSAGE_VECT                                     ;
                                50         04  D0 008A3  88$:     MOVL    #4, R0                                                ; 1121
                                           04  008A6          RET                                                           ;
                                50         01  D0 008A7  89$:     MOVL    #1, R0                                                ; 1126
                                           04  008AA          RET                                                           ; 1128
```

```
; Routine Size:  2219 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  999     1129  1 ROUTINE dbg$nparse_show_key (input_desc, verb_node, message_vect) =
; 1000     1130  1 !++
; 1001     1131  1 ! Functional Description
; 1002     1132  1 !
; 1003     1133  1 !       This is the parse network for the SHOW KEY command.
; 1004     1134  1 !
; 1005     1135  1 ! Routine Inputs
; 1006     1136  1 !
; 1007     1137  1 !       input_desc      -       A pointer to a string descriptor for the
; 1008     1138  1 !                                  remaining input.
; 1009     1139  1 !       verb_node       -       A pointer to the verb node for the SHOW
; 1010     1140  1 !                                  KEY command, which will be the top-level
; 1011     1141  1 !                                  node in the command execution tree.
; 1012     1142  1 !       message_vect    -       A pointer to an error message vector.
; 1013     1143  1 !
; 1014     1144  1 ! Routine Outputs
; 1015     1145  1 !
; 1016     1146  1 !       A command execution tree is constructed starting at the verb
; 1017     1147  1 !       node:
; 1018     1148  1 !
; 1019     1149  1 !        ------      ------
; 1020     1150  1 !       ! VERB ! -> ! NOUN !
; 1021     1151  1 !        ------      ------
; 1022     1152  1 !           !                   The DBG$B_VERB_COMPOSITE field contains a
; 1023     1153  1 !        -------                value for the SHOW KEY command.
; 1024     1154  1 !       !ADVERB0!
; 1025     1155  1 !        -------               The noun node has the following information:
; 1026     1156  1 !           !
; 1027     1157  1 !        -------               DBG$L_NOUN_VALUE - A pointer to a descriptor
; 1028     1158  1 !       !ADVERB1!                      that contains the key-name.
; 1029     1159  1 !        -------
; 1030     1160  1 !           :
; 1031     1161  1 !        -------
; 1032     1162  1 !       !ADVERB3!             The adverb nodes appear as follows:
; 1033     1163  1 !        -------
; 1034     1164  1 !                             DBG$L_ADVERB_VALUE - Value or location of data
; 1035     1165  1 !                                    for this qualifier.
; 1036     1166  1 !                             DBG$L_ADVERB_LINK - Link to next Adverb-node.
; 1037     1167  1 !
; 1038     1168  1 !       The string descriptor is updated to point past the
; 1039     1169  1 !       input that has been parsed. A completion code is returned:
; 1040     1170  1 !
; 1041     1171  1 !       STS$K_SUCCESS - The input was successfully parsed.
; 1042     1172  1 !       STS$K_SEVERE -  There were errors during the parse. An error
; 1043     1173  1 !                       message vector is constructed and returned
; 1044     1174  1 !                       in message_vect.
; 1045     1175  1 !--
; 1046     1176  2     BEGIN
; 1047     1177  2
; 1048     1178  2     MAP
; 1049     1179  2         input_desc              : REF BLOCK [,BYTE],    ! String descriptor
; 1050     1180  2         verb_node               : REF dbg$verb_node;
; 1051     1181  2
; 1052     1182  2     BIND
; 1053     1183  2         dbg$cs_all              = UPLIT BYTE (3, 'ALL'),
; 1054     1184  2         dbg$cs_brief            = UPLIT BYTE (5, 'BRIEF'),
; 1055     1185  2         dbg$cs_directory        = UPLIT BYTE (9, 'DIRECTORY'),
```

```
;  1056          1186  2          dbg$cs_state            = UPLIT BYTE (5, 'STATE'),
;  1057          1187  2          dbg$cs_nostate          = UPLIT BYTE (7, 'NOSTATE'),
;  1058          1188  2          dbg$cs_left_paren       = UPLIT BYTE (1, dbg$k_left_parenthesis),
;  1059          1189  2          dbg$cs_right_paren      = UPLIT BYTE (1, dbg$k_right_parenthesis),
;  1060          1190  2          dbg$cs_comma            = UPLIT BYTE (1, dbg$k_comma),
;  1061          1191  2          dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
;  1062          1192  2          dbg$cs_equal            = UPLIT BYTE (1, dbg$k_equal),
;  1063          1193  2          dbg$cs_slash            = UPLIT BYTE (1, dbg$k_slash);
;  1064          1194  2
;  1065          1195  2      LITERAL
;  1066          1196  2          dbg$k_lowest_qualifier  = 0,    ! These correspond to the adverb nodes
;  1067          1197  2          dbg$k_directory         = 0,    ! of the tree that is being constructed.
;  1068          1198  2          dbg$k_all               = 1,
;  1069          1199  2          dbg$k_brief             = 2,
;  1070          1200  2          dbg$k_state             = 3,
;  1071          1201  2          dbg$k_highest_qualifier = 3;
;  1072          1202  2
;  1073          1203  2      LOCAL
;  1074          1204  2          all_flag                : INITIAL(FALSE),       ! True if /ALL
;  1075          1205  2          dir_flag                : INITIAL(FALSE),       ! True if /DIRECTORY
;  1076          1206  2          define_kind             : INITIAL(0),           ! Value of DELETE/KEY qualifier
;  1077          1207  2          noun_node               : REF dbg$noun_node,    ! Pointer to a noun node
;  1078          1208  2          new_noun_node           : REF dbg$noun_node,    ! Another pointer to a noun node
;  1079          1209  2          adverb_node             : REF dbg$adverb_node,  ! Pointer to a noun node
;  1080          1210  2          new_adverb_node         : REF dbg$adverb_node,  ! Another pointer to a adverb node
;  1081          1211  2          state_name_node         : REF dbg$state_name_node,      ! Pointer to a state-name node
;  1082          1212  2          new_state_name_node     : REF dbg$state_name_node,      ! Another pointer to a state-name node
;  1083          1213  2          ptr                     : REF VECTOR[,BYTE],    ! Points into input string
;  1084          1214  2          status,
;  1085          1215  2          temp_key_desc           : REF dbg$stg_desc,     ! String desc. for DELETE/KEY symbols
;  1086          1216  2          define_key_value;                               ! Value for the qualifier
;  1087          1217  2
;  1088          1218  2
;  1089          1219  2      ! Check whether we are on a system that allows keypad input.
;  1090          1220  2      !
;  1091          1221  2      IF NOT .dbg$gb_keypad_input
;  1092          1222  2      THEN
;  1093          1223  2          SIGNAL(dbg$_nokeydef);
;  1094          1224  2
;  1095          1225  2      ! Fill in the fact that this is a SHOW KEY command in the verb node.
;  1096          1226  2      ! And clear the noun link value.
;  1097          1227  2      !
;  1098          1228  2      verb_node [dbg$b_verb_composite] = show_key;
;  1099          1229  2      verb_node [dbg$l_verb_object_ptr] = 0;
;  1100          1230  2
;  1101          1231  2      ! Build adverb list with defaults.
;  1102          1232  2      !
;  1103          1233  2
;  1104          1234  2      new_adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);  ! Get first node
;  1105          1235  2
;  1106          1236  2      verb_node [dbg$l_verb_adverb_ptr] = .new_adverb_node;
;  1107          1237  2      adverb_node = .new_adverb_node;
;  1108          1238  2
;  1109          1239  2      adverb_node [dbg$b_adverb_literal] = dbg$k_lowest_qualifier;        ! Initialize first node
;  1110          1240  2      adverb_node [dbg$l_adverb_value]   = 0;
;  1111          1241  2      adverb_node [dbg$l_adverb_link]    = 0;
;  1112          1242  2
```

```
;  1113          1243  2     define_kind = dbg$k_lowest_qualifier + 1;
;  1114          1244  2     WHILE .define_kind LEQ dbg$k_highest_qualifier DO          ! Build rest of adverb list
;  1115          1245  3         BEGIN
;  1116          1246  3         new_adverb_node = dbg$get_tempmem(dbg$k_adverb_node_size);
;  1117          1247  3         adverb_node [dbg$l_adverb_link] = .new_adverb_node;
;  1118          1248  3         adverb_node = .new_adverb_node;
;  1119          1249  3
;  1120          1250  3         adverb_node [dbg$b_adverb_literal] = .define_kind;
;  1121          1251  3         IF .define_kind EQL dbg$k_state
;  1122          1252  3         THEN
;  1123          1253  4             BEGIN
;  1124          1254  4             ! For the adverb node that has state-name information, the default
;  1125          1255  4             ! is set up so that the adverb node points to a state-name node that
;  1126          1256  4             ! points to a descriptor that contains the state-name.
;  1127          1257  4             !
;  1128          1258  4             temp_key_desc = dbg$get_tempmem(2);
;  1129          1259  4             temp_key_desc [dsc$w_length]  = 0;
;  1130          1260  4             temp_key_desc [dsc$b_dtype]   = dsc$k_dtype_t;
;  1131          1261  4             temp_key_desc [dsc$b_class]   = dsc$k_class_d;
;  1132          1262  4             temp_key_desc [dsc$a_pointer] = 0;
;  1133          1263  4
;  1134          1264  4             ! This routine returns the current state name.
;  1135          1265  4             !
;  1136          1266  4             smg$set_default_state(dbg$gl_key_table_id, 0, .temp_key_desc);
;  1137          1267  4
;  1138          1268  4             state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  1139          1269  4             state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  1140          1270  4             state_name_node [dbg$l_state_name_link] = 0;
;  1141          1271  4             adverb_node [dbg$l_adverb_value] = .state_name_node;
;  1142          1272  4             END
;  1143          1273  3         ELSE
;  1144          1274  3             ! In all other cases, the default value is set to zero
;  1145          1275  3             !
;  1146          1276  3             adverb_node [dbg$l_adverb_value] = 0;
;  1147          1277  3
;  1148          1278  3         define_kind = .define_kind + 1;
;  1149          1279  2         END;
;  1150          1280  2     adverb_node [dbg$l_adverb_link] = 0;
;  1151          1281  2
;  1152          1282  2     WHILE (NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)) AND
;  1153          1283  2           (.input_desc [dsc$w_length] GTR 0) DO
;  1154          1284  2
;  1155          1285  3         BEGIN
;  1156          1286  3         IF dbg$nmatch(.input_desc, dbg$cs_slash, 1)
;  1157          1287  3         THEN
;  1158          1288  4             BEGIN
;  1159          1289  4
;  1160          1290  4             ! Find out what kind of qualifier it is
;  1161          1291  4             !
;  1162          1292  4             ! Initialize value
;  1163          1293  4
;  1164          1294  4             define_key_value = 0;
;  1165          1295  4
;  1166          1296  4             ! Set Define_key with qualifier code, and get value of define_key_value.
;  1167          1297  4
;  1168          1298  4             SELECTONE TRUE  OF
;  1169          1299  4                 SET
```

```
;  1170          1300  4
;  1171          1301  4                [dbg$nmatch(.input_desc, dbg$cs_all, 1)] :
;  1172          1302  5                    BEGIN
;  1173          1303  5                    ! Check if a key-name has already been found, this makes the
;  1174          1304  5                    ! /ALL invalid.
;  1175          1305  5                    !
;  1176          1306  5                    IF .verb_node [dbg$l_verb_object_ptr] NEQ 0
;  1177          1307  5                    THEN
;  1178          1308  5                        SIGNAL(dbg$_conflict);
;  1179          1309  5                    define_kind = dbg$k_all;
;  1180          1310  5                    define_key_value = 1;
;  1181          1311  5                    all_flag = TRUE;
;  1182          1312  4                    END;
;  1183          1313  4
;  1184          1314  4                [dbg$nmatch(.input_desc, dbg$cs_nostate, 3)] :
;  1185          1315  5                    BEGIN
;  1186          1316  5                    define_key_value = 0;
;  1187          1317  5                    define_kind = dbg$k_state;
;  1188          1318  4                    END;
;  1189          1319  4
;  1190          1320  4                [dbg$nmatch(.input_desc, dbg$cs_state, 1)] :
;  1191          1321  5                    BEGIN
;  1192          1322  5                    define_key_value = 0;
;  1193          1323  5                    define_kind = dbg$k_state;
;  1194          1324  5
;  1195          1325  5                    temp_key_desc = dbg$get_tempmem(2);
;  1196          1326  5                    temp_key_desc[dsc$w_length]  = 0;
;  1197          1327  5                    temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1198          1328  5                    temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  1199          1329  5                    temp_key_desc[dsc$a_pointer] = 0;
;  1200          1330  5
;  1201          1331  5                    ! Look for =
;  1202          1332  5                    !
;  1203          1333  5
;  1204          1334  5                    IF NOT dbg$nmatch (.input_desc, dbg$cs_equal, 1)
;  1205          1335  5                    THEN
;  1206          1336  5                        report_error;
;  1207          1337  5
;  1208          1338  5                    ! Look for a left paren
;  1209          1339  5                    !
;  1210          1340  5
;  1211          1341  5                    IF dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  1212          1342  5                    THEN
;  1213          1343  6                        BEGIN
;  1214          1344  6
;  1215          1345  6                        ! Pick up the first state name
;  1216          1346  6                        !
;  1217          1347  6                        status = dbg$read_key_info (.input_desc,
;  1218          1348  6                                                    .temp_key_desc,
;  1219          1349  6                                                    .message_vect);
;  1220          1350  6                        IF NOT .status
;  1221          1351  6                        THEN
;  1222          1352  6                            RETURN sts$k_severe;
;  1223          1353  6
;  1224          1354  6                        new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  1225          1355  6                        state_name_node = .new_state_name_node;
;  1226          1356  6                        state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
```

```
;  1227     1357  6                    state_name_node [dbg$l_state_name_link] = 0;
;  1228     1358  6                    define_key_value = .state_name_node;
;  1229     1359  6
;  1230     1360  6                    WHILE dbg$nmatch (.input_desc, dbg$cs_comma, 1) DO
;  1231     1361  7                        BEGIN
;  1232     1362  7                        temp_key_desc = dbg$get_tempmem(2);
;  1233     1363  7                        temp_key_desc[dsc$w_length]  = 0;
;  1234     1364  7                        temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1235     1365  7                        temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  1236     1366  7                        temp_key_desc[dsc$a_pointer] = 0;
;  1237     1367  7
;  1238     1368  7                        ! Pick up the next state name
;  1239     1369  7                        !
;  1240     1370  7                        status = dbg$read_key_info (.input_desc,
;  1241     1371  7                                                    .temp_key_desc,
;  1242     1372  7                                                    .message_vect);
;  1243     1373  7                        IF NOT .status
;  1244     1374  7                        THEN
;  1245     1375  7                            RETURN sts$k_severe;
;  1246     1376  7
;  1247     1377  7                        new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  1248     1378  7                        state_name_node [dbg$l_state_name_link] = .new_state_name_node;
;  1249     1379  7                        state_name_node = .new_state_name_node;
;  1250     1380  7                        state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  1251     1381  7                        state_name_node [dbg$l_state_name_link] = 0;
;  1252     1382  7
;  1253     1383  6                        END;
;  1254     1384  6
;  1255     1385  6                    ! Eat right paren
;  1256     1386  6                    !
;  1257     1387  6
;  1258     1388  6                    IF NOT dbg$nmatch (.input_desc, dbg$cs_right_paren, 1)
;  1259     1389  6                    THEN
;  1260     1390  6                        report_error;
;  1261     1391  6
;  1262     1392  6                    END
;  1263     1393  6
;  1264     1394  5                ELSE
;  1265     1395  6                    BEGIN
;  1266     1396  6
;  1267     1397  6                    ! Pick up the only state name
;  1268     1398  6                    !
;  1269     1399  6
;  1270     1400  6                    status = dbg$read_key_info (.input_desc,
;  1271     1401  6                                                .temp_key_desc,
;  1272     1402  6                                                .message_vect);
;  1273     1403  6                    IF NOT .status
;  1274     1404  6                    THEN
;  1275     1405  6                        RETURN sts$k_severe;
;  1276     1406  6
;  1277     1407  6                    new_state_name_node = dbg$get_tempmem(dbg$k_state_name_size);
;  1278     1408  6                    state_name_node = .new_state_name_node;
;  1279     1409  6                    state_name_node [dbg$l_state_name_ptr] = .temp_key_desc;
;  1280     1410  6                    state_name_node [dbg$l_state_name_link] = 0;
;  1281     1411  6                    define_key_value = .state_name_node;
;  1282     1412  6
;  1283     1413  5                    END;
```

```
;  1284          1414  5
;  1285          1415  4                        END;
;  1286          1416  4
;  1287          1417  4                    [dbg$nmatch(.input_desc, dbg$cs_brief, 1)] :
;  1288          1418  5                        BEGIN
;  1289          1419  5                        define_key_value = 1;
;  1290          1420  5                        define_kind = dbg$k_brief;
;  1291          1421  4                        END;
;  1292          1422  4
;  1293          1423  4                    [dbg$nmatch(.input_desc, dbg$cs_directory, 1)] :
;  1294          1424  5                        BEGIN
;  1295          1425  5                        IF .verb_node [dbg$l_verb_object_ptr] NEQ 0
;  1296          1426  5                        THEN
;  1297          1427  5                            SIGNAL(dbg$_conflict);
;  1298          1428  5                        dir_flag = TRUE;
;  1299          1429  5                        define_key_value = 1;
;  1300          1430  5                        define_kind = dbg$k_directory;
;  1301          1431  4                        END;
;  1302          1432  4
;  1303          1433  4                    [OTHERWISE] :
;  1304          1434  4                        report_error;
;  1305          1435  4                    TES;
;  1306          1436  4
;  1307          1437  4                ! Process the qualifier
;  1308          1438  4
;  1309          1439  4                IF .define_key_value NEQ 0
;  1310          1440  4                THEN
;  1311          1441  5                    BEGIN
;  1312          1442  5                    adverb_node = .verb_node [dbg$l_verb_adverb_ptr];
;  1313          1443  5                    WHILE (.adverb_node [dbg$b_adverb_literal] NEQ .define_kind) DO
;  1314          1444  5                        adverb_node = .adverb_node [dbg$l_adverb_link];
;  1315          1445  5                    adverb_node [dbg$l_adverb_value] = .define_key_value;
;  1316          1446  5
;  1317          1447  5                    ! If /STATE=xxxxx was specified, the link field is set to one.
;  1318          1448  5                    ! This is done to check for /STATE/DIR in the command,
;  1319          1449  5                    ! which is invalid.
;  1320          1450  5                    !
;  1321          1451  5
;  1322          1452  5                    IF .adverb_node [dbg$b_adverb_literal] EQL dbg$k_state
;  1323          1453  5                    THEN
;  1324          1454  5                        adverb_node [dbg$l_adverb_link] = 1;
;  1325          1455  4                    END;
;  1326          1456  4
;  1327          1457  4                END ! End of qualifier search code.
;  1328          1458  4
;  1329          1459  3            ELSE
;  1330          1460  3
;  1331          1461  3                ! Process key name
;  1332          1462  3
;  1333          1463  4                IF (.verb_node [dbg$l_verb_object_ptr] EQL 0) AND (NOT .all_flag) AND (NOT .dir_flag)
;  1334          1464  3                THEN
;  1335          1465  4                    BEGIN
;  1336          1466  4
;  1337          1467  4                    ! Get key name
;  1338          1468  4                    !
;  1339          1469  4
;  1340          1470  4                    temp_key_desc = dbg$get_tempmem(2);
```

```
;  1341          1471  4                 temp_key_desc[dsc$w_length]  = 0;
;  1342          1472  4                 temp_key_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1343          1473  4                 temp_key_desc[dsc$b_class]   = dsc$k_class_d;
;  1344          1474  4                 temp_key_desc[dsc$a_pointer] = 0;
;  1345          1475  4                 status = dbg$read_key_info (.input_desc,
;  1346          1476  4                                             .temp_key_desc,
;  1347          1477  4                                             .message_vect);
;  1348          1478  4                 IF NOT .status
;  1349          1479  4                 THEN
;  1350          1480  4                     RETURN sts$k_severe;
;  1351          1481  4
;  1352          1482  4                 ! Make noun node for key-name string
;  1353          1483  4                 !
;  1354          1484  4
;  1355          1485  4                 new_noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  1356          1486  4
;  1357          1487  4                 verb_node [dbg$l_verb_object_ptr] = .new_noun_node;
;  1358          1488  4                 noun_node = .new_noun_node;
;  1359          1489  4                 noun_node [dbg$l_noun_value] = .temp_key_desc;
;  1360          1490  4                 noun_node [dbg$l_adjective_ptr] = 0;
;  1361          1491  4                 noun_node [dbg$l_noun_link] = 0;
;  1362          1492  4                 END
;  1363          1493  4
;  1364          1494  3             ELSE
;  1365          1495  3                 report_error;
;  1366          1496  3
;  1367          1497  2         END;    ! End While
;  1368          1498  2
;  1369          1499  2         ! Check to see if key-name string or /ALL or /DIRECTORY has been entered.
;  1370          1500  2         ! If not, return a message and error status.
;  1371          1501  2         !
;  1372          1502  2
;  1373          1503  3         IF (.verb_node [dbg$l_verb_object_ptr] EQL 0) AND (NOT .all_flag) AND (NOT .dir_flag)
;  1374          1504  2         THEN
;  1375          1505  3             BEGIN
;  1376          1506  3             .message_vect = dbg$nmake_arg_vect(dbg$_needmore);
;  1377          1507  3             RETURN sts$k_severe;
;  1378          1508  2             END;
;  1379          1509  2
;  1380          1510  2         RETURN sts$k_success;
;  1381          1511  1         END;


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

                                                        03  00126 P.ABS:  .BYTE   3
                                                    4C  4C  41  00127          .ASCII  \ALL\                ;
                                                        05  0012A P.ABT:  .BYTE   5
                                                46  45  49  52  42  0012B          .ASCII  \BRIEF\              ;
                                                        09  00130 P.ABU:  .BYTE   9
                                            59  52  4F  54  43  45  52  49  44  00131          .ASCII  \DIRECTORY\          ;
                                                        05  0013A P.ABV:  .BYTE   5
                                                45  54  41  54  53  0013B          .ASCII  \STATE\              ;
                                                        07  00140 P.ABW:  .BYTE   7
                                                45  54  41  54  53  4F  4E  00141          .ASCII  \NOSTATE\            ;
                                                        28  01  00148 P.ABX:  .BYTE   1, 40                    ;
```

```
                                29  01  0014A P.ABY:   .BYTE   1, 41                                          ;
                                2C  01  0014C P.ABZ:   .BYTE   1, 44                                          ;
                                0D  01  0014E P.ACA:   .BYTE   1, 13                                          ;
                                3D  01  00150 P.ACB:   .BYTE   1, 61                                          ;
                                2F  01  00152 P.ACC:   .BYTE   1, 47                                          ;

                                                DBG$CS_ALL=          P.ABS
                                                DBG$CS_BRIEF=        P.ABT
                                                DBG$CS_DIRECTORY=    P.ABU
                                                DBG$CS_STATE=        P.ABV
                                                DBG$CS_NOSTATE=      P.ABW
                                                DBG$CS_LEFT_PAREN=   P.ABX
                                                DBG$CS_RIGHT_PAREN=  P.ABY
                                                DBG$CS_COMMA=        P.ABZ
                                                DBG$CS_CR=           P.ACA
                                                DBG$CS_EQUAL=        P.ACB
                                                DBG$CS_SLASH=        P.ACC


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

                                        OFFC 00000 DBG$NPARSE_SHOW_KEY:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11            ; 1129
                        5E                      0C  C2 00002            SUBL2   #12, SP                         ;
                                                7E  7C 00005            CLRQ    DIR_FLAG                        ; 1176
                                                5A  D4 00007            CLRL    DEFINE_KIND                     ;
                        0D 00000000G            00  E8 00009            BLBS    DBG$GB_KEYPAD_INPUT, 1$         ; 1221
                           00028D18             8F  DD 00010            PUSHL   #167192                         ; 1223
                00000000G  00                   01  FB 00016            CALLS   #1, LIB$SIGNAL                  ;
                        59              08      AC  D0 0001D 1$:        MOVL    VERB_NODE, R9                   ; 1228
                     01 A9                      1B  90 00021            MOVB    #27, 1(R9)                      ;
                     0C AE              08      A9  9E 00025            MOVAB   8(R9), 12(SP)                   ; 1229
                                        0C      BE  D4 0002A            CLRL    @12(SP)                         ;
                                                03  DD 0002D            PUSHL   #3                              ; 1234
                00000000G  00                   01  FB 0002F            CALLS   #1, DBG$GET_TEMPMEM             ;
                        53                      50  D0 00036            MOVL    R0, NEW_ADVERB_NODE             ;
                     04 A9                      53  D0 00039            MOVL    NEW_ADVERB_NODE, 4(R9)          ; 1236
                        55                      53  D0 0003D            MOVL    NEW_ADVERB_NODE, ADVERB_NODE    ; 1237
                                                65  94 00040            CLRB    (ADVERB_NODE)                   ; 1239
                                        04      A5  7C 00042            CLRQ    4(ADVERB_NODE)                  ; 1240
                        5A                      01  D0 00045            MOVL    #1, DEFINE_KIND                 ; 1243
                        03                      5A  D1 00048 2$:        CMPL    DEFINE_KIND, #3                 ; 1244
                                                61  14 0004B            BGTR    5$                              ;
                                                03  DD 0004D            PUSHL   #3                              ; 1246
                00000000G  00                   01  FB 0004F            CALLS   #1, DBG$GET_TEMPMEM             ;
                        53                      50  D0 00056            MOVL    R0, NEW_ADVERB_NODE             ;
                     08 A5                      53  D0 00059            MOVL    NEW_ADVERB_NODE, 8(ADVERB_NODE) ; 1247
                        55                      53  D0 0005D            MOVL    NEW_ADVERB_NODE, ADVERB_NODE    ; 1248
                        65                      5A  90 00060            MOVB    DEFINE_KIND, (ADVERB_NODE)      ; 1250
                        03                      5A  D1 00063            CMPL    DEFINE_KIND, #3                 ; 1251
                                                3F  12 00066            BNEQ    3$                              ;
                                                02  DD 00068            PUSHL   #2                              ; 1258
                00000000G  00                   01  FB 0006A            CALLS   #1, DBG$GET_TEMPMEM             ;
                        52                      50  D0 00071            MOVL    R0, TEMP_KEY_DESC               ;
                        62 020E0000             8F  D0 00074            MOVL    #34471936, (TEMP_KEY_DESC)      ; 1259
                                        04      A2  D4 0007B            CLRL    4(TEMP_KEY_DESC)                ; 1262
                                                52  DD 0007E            PUSHL   TEMP_KEY_DESC                   ; 1266
```

```
                        7E  D4 00080          CLRL    -(SP)                                           ;
                00000000G   00  9F 00082          PUSHAB  DBG$GL_KEY_TABLE_ID                             ;
00000000G   00              03  FB 00088          CALLS   #3, SMG$SET_DEFAULT_STATE                       ;
                            02  DD 0008F          PUSHL   #2                                              ; 1268
00000000G   00              01  FB 00091          CALLS   #1, DBG$GET_TEMPMEM                             ;
                54          50  D0 00098          MOVL    R0, STATE_NAME_NODE                             ;
                64          52  D0 0009B          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)                ; 1269
                      04    A4  D4 0009E          CLRL    4(STATE_NAME_NODE)                              ; 1270
          04    A5            54  D0 000A1          MOVL    STATE_NAME_NODE, 4(ADVERB_NODE)                 ; 1271
                            03  11 000A5          BRB     4$                                              ; 1251
                      04    A5  D4 000A7  3$:     CLRL    4(ADVERB_NODE)                                  ; 1276
                            5A  D6 000AA  4$:     INCL    DEFINE_KIND                                     ; 1278
                            9A  11 000AC          BRB     2$                                              ; 1244
                      08    A5  D4 000AE  5$:     CLRL    8(ADVERB_NODE)                                  ; 1280
                53          04  AC D0 000B1          MOVL    INPUT_DESC, R3                              ; 1282
                            01  DD 000B5  6$:     PUSHL   #1                                              ;
                00000000'   EF  9F 000B7          PUSHAB  DBG$CS_CR                                       ;
                            53  DD 000BD          PUSHL   R3                                              ;
00000000G   00              03  FB 000BF          CALLS   #3, DBG$NMATCH                                  ;
                03          50  E9 000C6          BLBC    R0, 8$                                          ;
                        029E    31 000C9  7$:     BRW     37$                                             ;
                            63  B5 000CC  8$:     TSTW    (R3)                                            ; 1283
                            F9  13 000CE          BEQL    7$                                              ;
                            01  DD 000D0          PUSHL   #1                                              ; 1286
                00000000'   EF  9F 000D2          PUSHAB  DBG$CS_SLASH                                    ;
                            53  DD 000D8          PUSHL   R3                                              ;
00000000G   00              03  FB 000DA          CALLS   #3, DBG$NMATCH                                  ;
                03          50  E8 000E1          BLBS    R0, 9$                                          ;
                        022B    31 000E4          BRW     33$                                             ;
                            56  D4 000E7  9$:     CLRL    DEFINE_KEY_VALUE                                ; 1294
                            01  DD 000E9          PUSHL   #1                                              ; 1301
                00000000'   EF  9F 000EB          PUSHAB  DBG$CS_ALL                                      ;
                            53  DD 000F1          PUSHL   R3                                              ;
00000000G   00              03  FB 000F3          CALLS   #3, DBG$NMATCH                                  ;
                01          50  D1 000FA          CMPL    R0, #1                                          ;
                            1E  12 000FD          BNEQ    11$                                             ;
                      0C    BE  D5 000FF          TSTL    @12(SP)                                         ; 1306
                            0D  13 00102          BEQL    10$                                             ;
                00028158    8F  DD 00104          PUSHL   #164184                                         ; 1308
00000000G   00              01  FB 0010A          CALLS   #1, LIB$SIGNAL                                  ;
                5A          01  D0 00111 10$:     MOVL    #1, DEFINE_KIND                                 ; 1309
                56          01  D0 00114          MOVL    #1, DEFINE_KEY_VALUE                            ; 1310
          04    AE            01  D0 00117          MOVL    #1, ALL_FLAG                                    ; 1311
                            1B  11 0011B          BRB     12$                                             ; 1298
                            03  DD 0011D 11$:     PUSHL   #3                                              ; 1314
                00000000'   EF  9F 0011F          PUSHAB  DBG$CS_NOSTATE                                  ;
                            53  DD 00125          PUSHL   R3                                              ;
00000000G   00              03  FB 00127          CALLS   #3, DBG$NMATCH                                  ;
                01          50  D1 0012E          CMPL    R0, #1                                          ;
                            08  12 00131          BNEQ    13$                                             ;
                            56  D4 00133          CLRL    DEFINE_KEY_VALUE                                ; 1316
                5A          03  D0 00135          MOVL    #3, DEFINE_KIND                                 ; 1317
                        01B3    31 00138 12$:     BRW     30$                                             ; 1298
                            01  DD 0013B 13$:     PUSHL   #1                                              ; 1320
                00000000'   EF  9F 0013D          PUSHAB  DBG$CS_STATE                                    ;
                            53  DD 00143          PUSHL   R3                                              ;
00000000G   00              03  FB 00145          CALLS   #3, DBG$NMATCH                                  ;
```

```
                    01              50  D1 0014C          CMPL    R0, #1                                  ;
                                    03  13 0014F          BEQL    14$                                     ;
                                  014C  31 00151          BRW     26$                                     ;
                                    56  D4 00154 14$:     CLRL    DEFINE_KEY_VALUE                        ; 1322
                                    5A  03  D0 00156      MOVL    #3, DEFINE_KIND                         ; 1323
                                        02  DD 00159      PUSHL   #2                                      ; 1325
00000000G                           00  01  FB 0015B      CALLS   #1, DBG$GET_TEMPMEM                     ;
                                    52  50  D0 00162      MOVL    R0, TEMP_KEY_DESC                       ;
                                    62 020E0000 8F  D0 00165  MOVL  #34471936, (TEMP_KEY_DESC)          ; 1326
                                            04  A2  D4 0016C  CLRL  4(TEMP_KEY_DESC)                    ; 1329
                                        01  DD 0016F      PUSHL   #1                                      ; 1334
                                00000000' EF  9F 00171      PUSHAB  DBG$CS_EQUAL                        ;
                                        53  DD 00177      PUSHL   R3                                      ;
00000000G                           00  03  FB 00179      CALLS   #3, DBG$NMATCH                          ;
                                    2C  50  E8 00180      BLBS    R0, 17$                                 ;
                                        01  DD 00183 15$:     PUSHL   #1                                  ; 1335
                                00000000' EF  9F 00185      PUSHAB  DBG$CS_CR                           ;
                                        53  DD 0018B      PUSHL   R3                                      ;
00000000G                           00  03  FB 0018D      CALLS   #3, DBG$NMATCH                          ;
                                    03  50  E9 00194      BLBC    R0, 16$                                 ;
                                  01DC  31 00197          BRW     38$                                     ;
                                        53  DD 0019A 16$:     PUSHL   R3                                  ;
00000000G                           00  01  FB 0019C      CALLS   #1, DBG$NNEXT_WORD                      ;
                                        50  DD 001A3      PUSHL   R0                                      ;
00000000G                           00  01  FB 001A5      CALLS   #1, DBG$NSYNTAX_ERROR                   ;
                                  01D4  31 001AC          BRW     39$                                     ;
                                        01  DD 001AF 17$:     PUSHL   #1                                  ; 1341
                                00000000' EF  9F 001B1      PUSHAB  DBG$CS_LEFT_PAREN                   ;
                                        53  DD 001B7      PUSHL   R3                                      ;
00000000G                           00  03  FB 001B9      CALLS   #3, DBG$NMATCH                          ;
                                    03  50  E8 001C0      BLBS    R0, 18$                                 ;
                                  00A7  31 001C3          BRW     23$                                     ;
                                    0C  AC  DD 001C6 18$:     PUSHL   MESSAGE_VECT                      ; 1349
                                        52  DD 001C9      PUSHL   TEMP_KEY_DESC                           ; 1348
                                        53  DD 001CB      PUSHL   R3                                      ; 1347
00000000G                           00  03  FB 001CD      CALLS   #3, DBG$READ_KEY_INFO                   ;
                                    5B  50  D0 001D4      MOVL    R0, STATUS                              ;
                                    59  5B  E9 001D7      BLBC    STATUS, 20$                             ; 1350
                                        02  DD 001DA      PUSHL   #2                                      ; 1354
00000000G                           00  01  FB 001DC      CALLS   #1, DBG$GET_TEMPMEM                     ;
                                08  AE  50  D0 001E3      MOVL    R0, NEW_STATE_NAME_NODE                 ;
                                    54  08  AE  D0 001E7  MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE    ; 1355
                                    64  52  D0 001EB      MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)        ; 1356
                                        04  A4  D4 001EE  CLRL    4(STATE_NAME_NODE)                      ; 1357
                                    56  54  D0 001F1      MOVL    STATE_NAME_NODE, DEFINE_KEY_VALUE       ; 1358
                                    58  04  A4  9E 001F4  MOVAB   4(R4), R8                               ; 1378
                                        01  DD 001F8 19$:     PUSHL   #1                                  ; 1360
                                00000000' EF  9F 001FA      PUSHAB  DBG$CS_COMMA                        ;
                                        53  DD 00200      PUSHL   R3                                      ;
00000000G                           00  03  FB 00202      CALLS   #3, DBG$NMATCH                          ;
                                    4A  50  E9 00209      BLBC    R0, 21$                                 ;
                                        02  DD 0020C      PUSHL   #2                                      ; 1362
00000000G                           00  01  FB 0020E      CALLS   #1, DBG$GET_TEMPMEM                     ;
                                    52  50  D0 00215      MOVL    R0, TEMP_KEY_DESC                       ;
                                    62 020E0000 8F  D0 00218  MOVL  #34471936, (TEMP_KEY_DESC)          ; 1363
                                            04  A2  D4 0021F  CLRL  4(TEMP_KEY_DESC)                    ; 1366
                                            0C  AC  DD 00222  PUSHL MESSAGE_VECT                        ; 1372
```

```
                                52  DD 00225          PUSHL   TEMP_KEY_DESC                                    ; 1371
                                53  DD 00227          PUSHL   R3                                              ; 1370
00000000G   00                  03  FB 00229          CALLS   #3, DBG$READ_KEY_INFO                           ;
            5B                  50  D0 00230          MOVL    R0, STATUS                                      ;
            48                  5B  E9 00233 20$:     BLBC    STATUS, 24$                                     ; 1373
                                02  DD 00236          PUSHL   #2                                              ; 1377
00000000G   00                  01  FB 00238          CALLS   #1, DBG$GET_TEMPMEM                             ;
      08    AE                  50  D0 0023F          MOVL    R0, NEW_STATE_NAME_NODE                         ;
            68          08      AE  D0 00243          MOVL    NEW_STATE_NAME_NODE, (R8)                       ; 1378
            54          08      AE  D0 00247          MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE            ; 1379
            64                  52  D0 0024B          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)                ; 1380
            58          04      A4  9E 0024E          MOVAB   4(R4), R8                                       ; 1381
                                68  D4 00252          CLRL    (R8)                                            ;
                                A2  11 00254          BRB     19$                                             ; 1360
                                01  DD 00256 21$:     PUSHL   #1                                              ; 1388
                    00000000'   EF  9F 00258          PUSHAB  DBG$CS_RIGHT_PAREN                              ;
                                53  DD 0025E          PUSHL   R3                                              ;
00000000G   00                  03  FB 00260          CALLS   #3, DBG$NMATCH                                  ;
            52                  50  E8 00267          BLBS    R0, 27$                                         ;
                          FF16  31  D0 0026A 22$:     BRW     15$                                             ; 1389
                      0C  AC    DD  0026D 23$:     PUSHL   MESSAGE_VECT                                    ; 1402
                                52  DD 00270          PUSHL   TEMP_KEY_DESC                                   ; 1401
                                53  DD 00272          PUSHL   R3                                              ; 1400
00000000G   00                  03  FB 00274          CALLS   #3, DBG$READ_KEY_INFO                           ;
            5B                  50  D0 0027B          MOVL    R0, STATUS                                      ;
            03                  5B  E8 0027E 24$:     BLBS    STATUS, 25$                                     ; 1403
                          0103  31  00281            BRW     40$                                             ;
                                02  DD 00284 25$:     PUSHL   #2                                              ; 1407
00000000G   00                  01  FB 00286          CALLS   #1, DBG$GET_TEMPMEM                             ;
      08    AE                  50  D0 0028D          MOVL    R0, NEW_STATE_NAME_NODE                         ;
            54          08      AE  D0 00291          MOVL    NEW_STATE_NAME_NODE, STATE_NAME_NODE            ; 1408
            64                  52  D0 00295          MOVL    TEMP_KEY_DESC, (STATE_NAME_NODE)                ; 1409
                        04      A4  D4 00298          CLRL    4(STATE_NAME_NODE)                              ; 1410
            56                  54  D0 0029B          MOVL    STATE_NAME_NODE, DEFINE_KEY_VALUE               ; 1411
                                4E  11 0029E          BRB     30$                                             ; 1298
                                01  DD 002A0 26$:     PUSHL   #1                                              ; 1417
                    00000000'   EF  9F 002A2          PUSHAB  DBG$CS_BRIEF                                    ;
                                53  DD 002A8          PUSHL   R3                                              ;
00000000G   00                  03  FB 002AA          CALLS   #3, DBG$NMATCH                                  ;
            01                  50  D1 002B1          CMPL    R0, #1                                          ;
                                08  12 002B4          BNEQ    28$                                             ;
            56                  01  D0 002B6          MOVL    #1, DEFINE_KEY_VALUE                            ; 1419
            5A                  02  D0 002B9          MOVL    #2, DEFINE_KIND                                 ; 1420
                                30  11 002BC 27$:     BRB     30$                                             ; 1298
                                01  DD 002BE 28$:     PUSHL   #1                                              ; 1423
                    00000000'   EF  9F 002C0          PUSHAB  DBG$CS_DIRECTORY                                ;
                                53  DD 002C6          PUSHL   R3                                              ;
00000000G   00                  03  FB 002C8          CALLS   #3, DBG$NMATCH                                  ;
            01                  50  D1 002CF          CMPL    R0, #1                                          ;
                                96  12 002D2          BNEQ    22$                                             ;
                          0C    BE  D5 002D4          TSTL    @12(SP)                                         ; 1425
                                0D  13 002D7          BEQL    29$                                             ;
                    00028158    8F  DD 002D9          PUSHL   #164184                                         ; 1427
00000000G   00                  01  FB 002DF          CALLS   #1, LIB$SIGNAL                                  ;
            6E                  01  D0 002E6 29$:     MOVL    #1, DIR_FLAG                                    ; 1428
            56                  01  D0 002E9          MOVL    #1, DEFINE_KEY_VALUE                            ; 1429
                                5A  D4 002EC          CLRL    DEFINE_KIND                                     ; 1430
```

```
                                                 56  D5  002EE 30$:    TSTL    DEFINE_KEY_VALUE                          ; 1439
                                                 75  13  002F0         BEQL    36$                                       ;
                                      55    04   A9  D0  002F2         MOVL    4(R9), ADVERB_NODE                        ; 1442
          5A            65            08        00  ED  002F6 31$:    CMPZV   #0, #8, (ADVERB_NODE), DEFINE_KIND        ; 1443
                                                 06  13  002FB         BEQL    32$                                       ;
                                      55    08   A5  D0  002FD         MOVL    8(ADVERB_NODE), ADVERB_NODE               ; 1444
                                                 F3  11  00301         BRB     31$                                       ;
                                04    A5          56  D0  00303 32$:    MOVL    DEFINE_KEY_VALUE, 4(ADVERB_NODE)          ; 1445
                                      03          65  91  00307         CMPB    (ADVERB_NODE), #3                         ; 1452
                                                 5B  12  0030A         BNEQ    36$                                       ;
                                08    A5          01  D0  0030C         MOVL    #1, 8(ADVERB_NODE)                        ; 1454
                                                 55  11  00310         BRB     36$                                       ; 1286
                                          0C    BE  D5  00312 33$:    TSTL    @12(SP)                                   ; 1463
                                                 04  12  00315         BNEQ    34$                                       ;
                                      03    04   AE  E9  00317         BLBC    ALL_FLAG, 35$                             ;
                                                FE65 31  0031B 34$:    BRW     15$                                       ;
                                      FA          6E  E8  0031E 35$:    BLBS    DIR_FLAG, 34$                             ;
                                                 02  DD  00321         PUSHL   #2                                        ; 1470
                          00000000G   00          01  FB  00323         CALLS   #1, DBG$GET_TEMPMEM                       ;
                                      52          50  D0  0032A         MOVL    R0, TEMP_KEY_DESC                         ;
                                      62 020E0000   8F  D0  0032D         MOVL    #34471936, (TEMP_KEY_DESC)                ; 1471
                                           04     A2  D4  00334         CLRL    4(TEMP_KEY_DESC)                          ; 1474
                                           0C     AC  DD  00337         PUSHL   MESSAGE_VECT                              ; 1477
                                                 52  DD  0033A         PUSHL   TEMP_KEY_DESC                             ; 1476
                                                 53  DD  0033C         PUSHL   R3                                        ; 1475
                          00000000G   00          03  FB  0033E         CALLS   #3, DBG$READ_KEY_INFO                     ;
                                      5B          50  D0  00345         MOVL    R0, STATUS                                ;
                                      3C          5B  E9  00348         BLBC    STATUS, 40$                               ; 1478
                                                 04  DD  0034B         PUSHL   #4                                        ; 1485
                          00000000G   00          01  FB  0034D         CALLS   #1, DBG$GET_TEMPMEM                       ;
                                10    AE          50  D0  00354         MOVL    R0, NEW_NOUN_NODE                         ;
                                0C    BE    10    AE  D0  00358         MOVL    NEW_NOUN_NODE, @12(SP)                    ; 1487
                                      57    10    AE  D0  0035D         MOVL    NEW_NOUN_NODE, NOUN_NODE                  ; 1488
                                      67          52  D0  00361         MOVL    TEMP_KEY_DESC, (NOUN_NODE)                ; 1489
                                           04     A7  7C  00364         CLRQ    4(NOUN_NODE)                              ; 1490
                                                FD4B 31  00367 36$:    BRW     6$                                        ; 1463
                                           0C    BE  D5  0036A 37$:    TSTL    @12(SP)                                   ; 1503
                                                 1C  12  0036D         BNEQ    41$                                       ;
                                      18    04   AE  E8  0036F         BLBS    ALL_FLAG, 41$                             ;
                                      15          6E  E8  00373         BLBS    DIR_FLAG, 41$                             ;
                                      000280D0    8F  DD  00376 38$:    PUSHL   #164048                                   ; 1506
                          00000000G   00          01  FB  0037C         CALLS   #1, DBG$NMAKE_ARG_VECT                    ;
                                0C    BC          50  D0  00383 39$:    MOVL    R0, @MESSAGE_VECT                         ;
                                      50          04  D0  00387 40$:    MOVL    #4, R0                                    ; 1507
                                                 04  0038A         RET                                               ;
                                      50          01  D0  0038B 41$:    MOVL    #1, R0                                    ; 1510
                                                 04  0038E         RET                                               ; 1511

; Routine Size:  911 bytes,    Routine Base:  DBG$CODE + 08AB

; 1382          1512  1
```

```
;  1384     1513  1 GLOBAL ROUTINE DBG$NEXECUTE_SHOW (VERB_NODE, MESSAGE_VECT) =
;  1385     1514  1
;  1386     1515  1 !++
;  1387     1516  1 ! FUNCTIONAL DESCRIPTION:
;  1388     1517  1 !
;  1389     1518  1 !       This routine accepts a command execution tree as input and performs the
;  1390     1519  1 !       semantic actions associated with the SHOW command. Version 2 debugger
;  1391     1520  1 !       routines and data structures are utilyzed during command execution.
;  1392     1521  1 !
;  1393     1522  1 ! FORMAL PARAMETERS:
;  1394     1523  1 !
;  1395     1524  1 !       VERB_NODE -             The first node in the command execution tree
;  1396     1525  1 !
;  1397     1526  1 !       MESSAGE_VECT -          The address of a longword to contain the address
;  1398     1527  1 !                               of a message argument vector
;  1399     1528  1 !
;  1400     1529  1 ! IMPLICIT INPUTS:
;  1401     1530  1 !
;  1402     1531  1 !       NONE
;  1403     1532  1 !
;  1404     1533  1 ! IMPLICIT OUTPUTS:
;  1405     1534  1 !
;  1406     1535  1 !       Semantic actions corresponding to the input command are performed. That
;  1407     1536  1 !       is, various states of the debugger are displayed.
;  1408     1537  1 !
;  1409     1538  1 ! ROUTINE VALUE:        unsigned longword integer completion code
;  1410     1539  1 !
;  1411     1540  1 ! COMPLETION CODES:
;  1412     1541  1 !
;  1413     1542  1 !       STS$K_SEVERE (4) -      The command could not be executed.
;  1414     1543  1 !
;  1415     1544  1 !       STS$K_SUCCESS (1) -     The parsed command was executed.
;  1416     1545  1 !
;  1417     1546  1 ! SIDE EFFECTS:
;  1418     1547  1 !
;  1419     1548  1 !       Output concerning the state of the debugger is displayed to the user.
;  1420     1549  1 !
;  1421     1550  1 !--
;  1422     1551  1
;  1423     1552  2     BEGIN
;  1424     1553  2
;  1425     1554  2     MAP
;  1426     1555  2         VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to command Verb Node
;  1427     1556  2
;  1428     1557  2
;  1429     1558  2
;  1430     1559  2     ! Transfer control to a subnetwork on the basis of the composite verb
;  1431     1560  2     !
;  1432     1561  2     CASE .VERB_NODE[DBG$B_VERB_COMPOSITE] FROM MIN_SHOW TO MAX_SHOW OF
;  1433     1562  2             SET
;  1434     1563  2
;  1435     1564  2             [show_break] :
;  1436     1565  2                 RETURN DBG$EVENT_SHOW_CANCEL_SEMANTICS (.VERB_NODE,
;  1437     1566  2                                                         .MESSAGE_VECT
;  1438     1567  2                                                        );
;  1439     1568  2
;  1440     1569  2             [show_calls] :
```

```
;  1441     1570  3                BEGIN
;  1442     1571  3                LOCAL
;  1443     1572  3                    EXC_TYPE,               ! Exception type (trap=1, fault=2)
;  1444     1573  3                    NOUN_NODE : REF dbg$noun_node;
;  1445     1574  3
;  1446     1575  3                noun_node = .verb_node [dbg$l_verb_object_ptr];
;  1447     1576  3
;  1448     1577  3                ! exception type is based on whether the last exception
;  1449     1578  3                ! was a fault, break or step-end
;  1450     1579  3                !
;  1451     1580  3                IF      .dbg$runframe [dbg$v_at_fault] OR
;  1452     1581  3                        .dbg$runframe [dbg$v_at_break] OR
;  1453     1582  3                        .dbg$runframe [dbg$v_at_step_end]
;  1454     1583  3                THEN    exc_type = fault_exc
;  1455     1584  3                ELSE    exc_type = trap_exc;
;  1456     1585  3
;  1457     1586  3                dbg$traceback (.dbg$runframe [dbg$l_user_pc],
;  1458     1587  3                               .dbg$runframe [dbg$l_user_fp],
;  1459     1588  3                               .EXC_TYPE, .NOUN_NODE[DBG$L_NOUN_VALUE]);
;  1460     1589  3                END;
;  1461     1590  2
;  1462     1591  2            [show_define] :
;  1463     1592  3                BEGIN
;  1464     1593  3                dbg$show_define();
;  1465     1594  3                END;
;  1466     1595  2
;  1467     1596  2            [show_developer] :
;  1468     1597  3                BEGIN
;  1469     1598  3                dbg$print (
;  1470     1599  3                        UPLIT BYTE (%ASCIC 'Developer Longword (in hex): !XL'),
;  1471     1600  3                        .dbg$gl_developer);
;  1472     1601  3                dbg$newline();
;  1473     1602  3                END;
;  1474     1603  2
;  1475     1604  2
;  1476     1605  2            ! Execute the SHOW DISPLAY command.
;  1477     1606  2            !
;  1478     1607  2            [SHOW_DISPLAY]:
;  1479     1608  2                DBG$SCR_EXECUTE_SHODISP_CMD(.VERB_NODE);
;  1480     1609  2
;  1481     1610  2
;  1482     1611  2            ! Execute the SHOW KEY command.
;  1483     1612  2            !
;  1484     1613  2            [show_key]:
;  1485     1614  3                BEGIN
;  1486     1615  3
;  1487     1616  3                LOCAL
;  1488     1617  3                    status;
;  1489     1618  3
;  1490     1619  3                status = dbg$nexecute_show_key(.verb_node, .message_vect);
;  1491     1620  3                IF NOT .status
;  1492     1621  3                THEN
;  1493     1622  3                    RETURN sts$k_severe;
;  1494     1623  3                END;
;  1495     1624  2
;  1496     1625  2            ! Execute the SHOW LANGUAGE command.
;  1497     1626  2            !
```

```
;  1498        1627  2              [show_language] :
;  1499        1628  3                  BEGIN
;  1500        1629  3                  $fao_tt_out ('language: !AC', dbg$language (.dbg$gb_language));
;  1501        1630  2                  END;
;  1502        1631  2
;  1503        1632  2              [show_log] :
;  1504        1633  3                  BEGIN
;  1505        1634  3                  dbg$nshow_output (2);      ! 2 stands for ''show log'' parameter
;  1506        1635  2                  END;
;  1507        1636  2
;  1508        1637  2              [show_margins] :
;  1509        1638  3                  BEGIN
;  1510        1639  3                  dbg$nshow_margins();
;  1511        1640  2                  END;
;  1512        1641  2
;  1513        1642  2              [show_max_source_files] :
;  1514        1643  3                  BEGIN
;  1515        1644  3                  dbg$nshow_max_source_files();
;  1516        1645  2                  END;
;  1517        1646  2
;  1518        1647  2              [show_mode] :
;  1519        1648  3                  BEGIN
;  1520        1649  3                  dbg$show_mode ();
;  1521        1650  2                  END;
;  1522        1651  2
;  1523        1652  2              [show_module] :
;  1524        1653  3                  BEGIN
;  1525        1654  3                  dbg$show_module ();
;  1526        1655  2                  END;
;  1527        1656  2
;  1528        1657  2              [show_output] :
;  1529        1658  3                  BEGIN
;  1530        1659  3                  dbg$nshow_output (1);      ! 1 stands for ''full rep''
;  1531        1660  2                  END;
;  1532        1661  2
;  1533        1662  2              [show_radix] :
;  1534        1663  3                  BEGIN
;  1535        1664  3                  ! Parameter EQL 0  =>  show the SET RADIX radix
;  1536        1665  3                  dbg$show_radix(0);
;  1537        1666  2                  END;
;  1538        1667  2
;  1539        1668  2              [show_radix_override]:
;  1540        1669  3                  BEGIN
;  1541        1670  3                  ! Parameter EQL 1  =>  show the SET RADIX/OVERRIDE radix
;  1542        1671  3                  dbg$show_radix(1);
;  1543        1672  2                  END;
;  1544        1673  2
;  1545        1674  2              [show_scope] :
;  1546        1675  3                  BEGIN
;  1547        1676  3                  dbg$rst_showscope ();
;  1548        1677  2                  END;
;  1549        1678  2
;  1550        1679  2              [show_search] :
;  1551        1680  3                  BEGIN
;  1552        1681  3                  dbg$show_search ();
;  1553        1682  2                  END;
;  1554        1683  2
```

```
;  1555     1684  2
;  1556     1685  2                ! Execute the SHOW SELECT command.
;  1557     1686  2                !
;  1558     1687  2                [SHOW_SELECT]:
;  1559     1688  2                    DBG$SCR_EXECUTE_SHOSEL_CMD(.VERB_NODE);
;  1560     1689  2
;  1561     1690  2
;  1562     1691  2                ! Execute the SHOW SOURCE command.
;  1563     1692  2                !
;  1564     1693  2                [SHOW_SOURCE] :
;  1565     1694  2                    DBG$SRC_SHOW_SOURCE();
;  1566     1695  2
;  1567     1696  2
;  1568     1697  2                [show_step] :
;  1569     1698  3                    BEGIN
;  1570     1699  3                    dbg$show_step ();
;  1571     1700  2                    END;
;  1572     1701  2
;  1573     1702  2                [show_symbol, show_symbol_defined]:
;  1574     1703  3                    BEGIN
;  1575     1704  3                    LOCAL
;  1576     1705  3                        addr_flag,
;  1577     1706  3                        flags,
;  1578     1707  3                        global_flag,
;  1579     1708  3                        type_flag,
;  1580     1709  3                        noun_node: REF dbg$noun_node,
;  1581     1710  3                        name_list: REF VECTOR[,LONG],
;  1582     1711  3                        status;
;  1583     1712  3
;  1584     1713  3                    ! Recover the flags.
;  1585     1714  3                    !
;  1586     1715  3                    noun_node = .verb_node [dbg$l_verb_object_ptr];
;  1587     1716  3                    flags = .noun_node[dbg$l_adjective_ptr];
;  1588     1717  3                    addr_flag = .flags/4;
;  1589     1718  3                    global_flag = (.flags mod 4)/2;
;  1590     1719  3                    type_flag = .flags mod 2;
;  1591     1720  3
;  1592     1721  3                    ! Show the non-defined symbols.
;  1593     1722  3                    !
;  1594     1723  3                    IF .verb_node[dbg$b_verb_composite] EQL show_symbol_defined
;  1595     1724  3                    THEN
;  1596     1725  4                        BEGIN
;  1597     1726  4                        status = FALSE;
;  1598     1727  4                        addr_flag = TRUE;
;  1599     1728  4                        type_flag = TRUE;
;  1600     1729  4                        END
;  1601     1730  4
;  1602     1731  3                    ELSE
;  1603     1732  3                        status = dbg$sta_showsymbol(.verb_node);
;  1604     1733  3
;  1605     1734  3
;  1606     1735  3                    ! Show the defined symbols as long as no IN clause was
;  1607     1736  3                    ! specified.
;  1608     1737  3                    !
;  1609     1738  3                    name_list = .noun_node[dbg$l_noun_value];
;  1610     1739  3                    WHILE .name_list NEQ 0 do
;  1611     1740  4                        BEGIN
```

```
;  1612     1741  4                    status = .name_list[2] GTR 0;
;  1613     1742  4                    IF .noun_node[dbg$l_noun_value2] EQL 0
;  1614     1743  4                    THEN
;  1615     1744  5                        BEGIN
;  1616     1745  5                        IF NOT dbg$dump_define( .name_list[1],
;  1617     1746  5                                                .addr_flag,
;  1618     1747  5                                                .global_flag,
;  1619     1748  5                                                .type_flag,
;  1620     1749  5                                                .status,
;  1621     1750  5                                                .message_vect)
;  1622     1751  5                        THEN
;  1623     1752  5                            RETURN sts$k_severe;
;  1624     1753  5                        END
;  1625     1754  4                    ELSE
;  1626     1755  5                        BEGIN
;  1627     1756  5                        IF NOT .status
;  1628     1757  5                        THEN
;  1629     1758  5                            SIGNAL(dbg$_symnotfnd, 1, .name_list[1]);
;  1630     1759  4                        END;
;  1631     1760  4
;  1632     1761  4                    name_list = .name_list[0];
;  1633     1762  3                    END;
;  1634     1763  2                END;
;  1635     1764  2
;  1636     1765  2
;  1637     1766  2            ! Execute the SHOW TASK command.
;  1638     1767  2            !
;  1639     1768  2            [SHOW_TASK]:
;  1640     1769  3                BEGIN
;  1641     1770  3                DBG$NEXECUTE_SHOW_TASK(.VERB_NODE);
;  1642     1771  2                END;
;  1643     1772  2
;  1644     1773  2
;  1645     1774  2            ! Execute the SHOW TERMINAL command.
;  1646     1775  2            !
;  1647     1776  2            [SHOW_TERMINAL]:
;  1648     1777  3                BEGIN
;  1649     1778  3                DBG$PRINT(UPLIT BYTE(%ASCIC 'terminal width: !SL'),
;  1650     1779  3                                                .DBG$SRC_TERM_WIDTH);
;  1651     1780  3                DBG$NEWLINE();
;  1652     1781  2                END;
;  1653     1782  2
;  1654     1783  2
;  1655     1784  2            ! Execute the SHOW TRACE command.
;  1656     1785  2            !
;  1657     1786  2            [SHOW_TRACE]:
;  1658     1787  2                RETURN DBG$EVENT_SHOW_CANCEL_SEMANTICS (.VERB_NODE,
;  1659     1788  2                                                        .MESSAGE_VECT);
;  1660     1789  2
;  1661     1790  2            [show_type] :
;  1662     1791  3                BEGIN
;  1663     1792  3                dbg$show_type (default);
;  1664     1793  2                END;
;  1665     1794  2
;  1666     1795  2            [show_type_override] :
;  1667     1796  3                BEGIN
;  1668     1797  3                dbg$show_type (override);
```

```
;  1669          1798  2                END;
;  1670          1799  2
;  1671          1800  2            [SHOW_WATCH] :
;  1672          1801  2                RETURN DBG$EVENT_SHOW_CANCEL_SEMANTICS(.VERB_NODE,
;  1673          1802  2                                                       .MESSAGE_VECT);
;  1674          1803  2
;  1675          1804  2            ! Execute the SHOW WINDOW command.
;  1676          1805  2            !
;  1677          1806  2            [SHOW_WINDOW]:
;  1678          1807  2                DBG$SCR_EXECUTE_SHOWIND_CMD(.VERB_NODE);
;  1679          1808  2
;  1680          1809  2
;  1681          1810  2            ! Any other SHOW command code is an internal DEBUG error.
;  1682          1811  2            !
;  1683          1812  2            [INRANGE, OUTRANGE] :
;  1684          1813  2                $DBG_ERROR('DBGNSHOW\NEXECUTE_SHOW');
;  1685          1814  2
;  1686          1815  2            TES;
;  1687          1816  2
;  1688          1817  2        RETURN STS$K_SUCCESS;
;  1689          1818  2
;  1690          1819  1        END;


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

67  6E  6F  4C  20  72  65  70  6F  6C  65  76  65  44  20  00154 P.ACD:  .ASCII  \ Developer Longword (in hex): !XL\                ;
20  3A  29  78  65  68  20  6E  69  28  20  64  72  6F  77  00163                                                              ;
                                    4C  58  21              00172                                                              ;
                                                    0D      00175 P.ACE:  .BYTE   13                                           ;
        43  41  21  20  3A  65  67  61  75  67  6E  61  6C  00176          .ASCII  \language: !AC\                               ;
68  74  64  69  77  20  6C  61  6E  69  6D  72  65  74  13  00183 P.ACF:  .ASCII  <19>\terminal width: !SL\                      ;
                                    4C  53  21  20  3A      00192                                                              ;
43  45  58  45  4E  5C  57  4F  48  53  4E  47  42  44  16  00197 P.ACG:  .ASCII  <22>\DBGNSHOW\<92>\NEXECUTE_SHOW\            ;
                        57  4F  48  53  5F  45  54  55      001A6                                                              ;


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

                                                07FC 00000          .ENTRY  DBG$NEXECUTE_SHOW, Save R2,R3,R4,R5,R6,R7,-        ; 1513
                                                                            R8,R9,R10                                          ;
                                    5A 00000000G   00   9E 00002            MOVAB   LIB$SIGNAL, R10                            ;
                                    59 00000000'   EF   9E 00009            MOVAB   P.ACG, R9                                  ;
                                    58 00000000G   00   9E 00010            MOVAB   DBG$RUNFRAME+72, R8                        ;
                                    52             04   AC  D0 00017            MOVL    VERB_NODE, R2                         ; 1561
                                    01             01   A2  8F 0001B            CASEB   1(R2), #1, #29                        ;
                1D                                          00020 1$:           .WORD   47$-1$,-                              ;
        00A7            003C            004B            01E3                            3$-1$,-                               ;
        00E7            00DE            00D5            00C3                    00028   2$-1$,-                               ;
        01D4            01E3            0123            00FD                    00030   11$-1$,-                              ;
        00C7            011A            01E3            01D8                    00038   13$-1$,-                              ;
        0077            012D            0106            00CE                    00040   16$-1$,-                              ;
        010F            008C            0080            012D                    00048   17$-1$,-                              ;
        00F0            0097            01F0            01BB                    00050   18$-1$,-                              ;
                                        01B0            00F4                    00058   25$-1$,-                              ;
```

```
                                                                31$-1$,-                                                                ;
                                                                47$-1$,-                                                                ;
                                                                44$-1$,-                                                                ;
                                                                45$-1$,-                                                                ;
                                                                47$-1$,-                                                                ;
                                                                29$-1$,-                                                                ;
                                                                14$-1$,-                                                                ;
                                                                15$-1$,-                                                                ;
                                                                26$-1$,-                                                                ;
                                                                33$-1$,-                                                                ;
                                                                7$-1$,-                                                                 ;
                                                                33$-1$,-                                                                ;
                                                                8$-1$,-                                                                 ;
                                                                9$-1$,-                                                                 ;
                                                                28$-1$,-                                                                ;
                                                                42$-1$,-                                                                ;
                                                                48$-1$,-                                                                ;
                                                                10$-1$,-                                                                ;
                                                                21$-1$,-                                                                ;
                                                                22$-1$,-                                                                ;
                                                                41$-1$                                                                  ;
                                59  DD 0005C 2$:        PUSHL   R9                                                                      ; 1813
                                01  DD 0005E            PUSHL   #1                                                                      ;
                          00028362  8F  DD 00060        PUSHL   #164706                                                                 ;
                                6A  03  FB 00066        CALLS   #3, LIB$SIGNAL                                                          ;
                                76  11  00069        BRB     12$                                                                     ;
                                51        08  A2  D0 0006B 3$:  MOVL    8(R2), NOUN_NODE                                                ; 1575
                          08        01  A8  05  E0 0006F        BBS     #5, DBG$RUNFRAME+73, 4$                                         ; 1580
                                05  68  E8 00074        BLBS    DBG$RUNFRAME+72, 4$                                                     ; 1581
                          05        01  A8  04  E1 00077        BBC     #4, DBG$RUNFRAME+73, 5$                                         ; 1582
                                50  02  D0 0007C 4$:  MOVL    #2, EXC_TYPE                                                            ; 1583
                                03  11  0007F        BRB     6$                                                                      ;
                                50  01  D0 00081 5$:  MOVL    #1, EXC_TYPE                                                            ; 1584
                                61  DD 00084 6$:  PUSHL   (NOUN_NODE)                                                             ; 1588
                                50  DD 00086        PUSHL   EXC_TYPE                                                                ;
                                      F0  A8  DD 00088        PUSHL   DBG$RUNFRAME+56                                                 ; 1587
                                      F8  A8  DD 0008B        PUSHL   DBG$RUNFRAME+64                                                 ; 1586
                00000000G  00       04  FB 0008E        CALLS   #4, DBG$TRACEBACK                                                       ;
                                77  11  00095        BRB     20$                                                                     ; 1561
                00000000G  00       00  FB 00097 7$:  CALLS   #0, DBG$SHOW_DEFINE                                                     ; 1593
                                7B  11  0009E        BRB     24$                                                                     ; 1561
                          00000000G   00  DD 000A0 8$:  PUSHL   DBG$GL_DEVELOPER                                                    ; 1600
                                      BD  A9  9F 000A6        PUSHAB  P.ACD                                                           ; 1599
                                    0138  31  000A9        BRW     43$                                                                     ;
                                52  DD 000AC 9$:  PUSHL   R2                                                                      ; 1608
                00000000G  00       01  FB 000AE        CALLS   #1, DBG$SCR_EXECUTE_SHODISP_CMD                                         ;
                                76  11  000B5        BRB     27$                                                                     ;
                                      08  AC  DD 000B7 10$:  PUSHL   MESSAGE_VECT                                                    ; 1619
                                52  DD 000BA        PUSHL   R2                                                                      ;
                          0000V CF  02  FB 000BC        CALLS   #2, DBG$NEXECUTE_SHOW_KEY                                               ;
                                7D  50  E8 000C1        BLBS    STATUS, 30$                                                             ; 1620
                                    00EF  31  000C4        BRW     38$                                                                     ; 1622
                                7E 00000000G  00  9A 000C7 11$:  MOVZBL  DBG$GB_LANGUAGE, -(SP)                                          ; 1629
                00000000G  00       01  FB 000CE        CALLS   #1, DBG$LANGUAGE                                                        ;
                                50  DD 000D5        PUSHL   R0                                                                      ;
                                      DE  A9  9F 000D7        PUSHAB  P.ACE                                                           ;
                00000000G  00       02  FB 000DA        CALLS   #2, DBG$FAO_OUT                                                         ;
```

```
                                     67  11 000E1 12$:    BRB     32$                                             ; 1561
                                     02  DD 000E3 13$:    PUSHL   #2                                              ; 1634
                                     22  11 000E5         BRB     19$                                             ;
                              0000V  CF  00  FB 000E7 14$:    CALLS   #0, DBG$NSHOW_MARGINS                       ; 1639
                                     5C  11 000EC         BRB     32$                                             ; 1561
                              0000V  CF  00  FB 000EE 15$:    CALLS   #0, DBG$NSHOW_MAX_SOURCE_FILES              ; 1644
                                     55  11 000F3         BRB     32$                                             ; 1561
                          00000000G  00  00  FB 000F5 16$:    CALLS   #0, DBG$SHOW_MODE                           ; 1649
                                     4C  11 000FC         BRB     32$                                             ; 1561
                          00000000G  00  00  FB 000FE 17$:    CALLS   #0, DBG$SHOW_MODULE                         ; 1654
                                     43  11 00105         BRB     32$                                             ; 1561
                                     01  DD 00107 18$:    PUSHL   #1                                              ; 1659
                              0000V  CF  01  FB 00109 19$:    CALLS   #1, DBG$NSHOW_OUTPUT                        ;
                                     3A  11 0010E 20$:    BRB     32$                                             ; 1561
                                     7E  D4 00110 21$:    CLRL    -(SP)                                           ; 1665
                                     02  11 00112         BRB     23$                                             ;
                                     01  DD 00114 22$:    PUSHL   #1                                              ; 1671
                              0000V  CF  01  FB 00116 23$:    CALLS   #1, DBG$SHOW_RADIX                          ;
                                     2D  11 0011B 24$:    BRB     32$                                             ; 1561
                          00000000G  00  00  FB 0011D 25$:    CALLS   #0, DBG$RST_SHOWSCOPE                       ; 1676
                                     24  11 00124         BRB     32$                                             ; 1561
                          00000000G  00  00  FB 00126 26$:    CALLS   #0, DBG$SHOW_SEARCH                         ; 1681
                                     1B  11 0012D 27$:    BRB     32$                                             ; 1561
                                     52  DD 0012F 28$:    PUSHL   R2                                              ; 1688
                          00000000G  00  01  FB 00131         CALLS   #1, DBG$SCR_EXECUTE_SHOSEL_CMD              ;
                                     10  11 00138         BRB     32$                                             ;
                          00000000G  00  00  FB 0013A 29$:    CALLS   #0, DBG$SRC_SHOW_SOURCE                     ; 1694
                                     07  11 00141 30$:    BRB     32$                                             ;
                          00000000G  00  00  FB 00143 31$:    CALLS   #0, DBG$SHOW_STEP                           ; 1699
                                 00CC  31 0014A 32$:    BRW     49$                                             ; 1561
                                 53  08  A2  D0 0014D 33$:    MOVL    8(R2), NOUN_NODE                            ; 1715
                                 51  04  A3  D0 00151         MOVL    4(NOUN_NODE), FLAGS                         ; 1716
                              56         51  04  C7 00155         DIVL3   #4, FLAGS, ADDR_FLAG                    ; 1717
                           7E 00         51  01  7A 00159         EMUL    #1, FLAGS, #0, -(SP)                    ; 1718
                           50 50         8E  04  7B 0015E         EDIV    #4, (SP)+, R0, R0                       ;
                              57         50  02  C7 00163         DIVL3   #2, R0, GLOBAL_FLAG                     ;
                           7E 00         51  01  7A 00167         EMUL    #1, FLAGS, #0, -(SP)                    ; 1719
                           54 54         8E  02  7B 0016C         EDIV    #2, (SP)+, TYPE_FLAG, TYPE_FLAG         ;
                                 15  01  A2  91 00171         CMPB    1(R2), #21                                  ; 1723
                                     08  12 00175         BNEQ    34$                                             ;
                                     56  01  D0 00177         MOVL    #1, ADDR_FLAG                               ; 1727
                                     54  01  7D 0017A         MOVQ    #1, TYPE_FLAG                               ; 1728
                                     0C  11 0017D         BRB     35$                                             ; 1723
                                     52  DD 0017F 34$:    PUSHL   R2                                              ; 1732
                          00000000G  00  01  FB 00181         CALLS   #1, DBG$STA_SHOWSYMBOL                      ;
                                     55  50  D0 00188         MOVL    R0, STATUS                                  ;
                                     52  63  D0 0018B 35$:    MOVL    (NOUN_NODE), NAME_LIST                      ; 1738
                                     BA  13 0018E 36$:    BEQL    32$                                             ; 1739
                                     50  D4 00190         CLRL    R0                                              ; 1741
                                 08  A2  D5 00192         TSTL    8(NAME_LIST)                                    ;
                                     02  15 00195         BLEQ    37$                                             ;
                                     50  D6 00197         INCL    R0                                              ;
                                     55  50  D0 00199 37$:    MOVL    R0, STATUS                                  ;
                                 0C  A3  D5 0019C         TSTL    12(NOUN_NODE)                                   ; 1742
                                     19  12 0019F         BNEQ    39$                                             ;
                                 08  AC  DD 001A1         PUSHL   MESSAGE_VECT                                    ; 1750
                                     30  BB 001A4         PUSHR   #^M<R4,R5>                                      ; 1748
```

```
                7E              56  7D 001A6          MOVQ    ADDR_FLAG, -(SP)                                ; 1746
                          04    A2  DD 001A9          PUSHL   4(NAME_LIST)                                    ; 1745
00000000G       00              06  FB 001AC          CALLS   #6, DBG$DUMP_DEFINE
                15              50  E8 001B3          BLBS    R0, 40$
                50              04  D0 001B6 38$:     MOVL    #4, R0                                          ; 1752
                                04  001B9            RET
                0E              55  E8 001BA 39$:     BLBS    STATUS, 40$                                     ; 1756
                          04    A2  DD 001BD          PUSHL   4(NAME_LIST)                                    ; 1758
                                01  DD 001C0          PUSHL   #1
                    000286BB    8F  DD 001C2          PUSHL   #165563
                6A              03  FB 001C8          CALLS   #3, LIB$SIGNAL
                52              62  D0 001CB 40$:     MOVL    (NAME_LIST), NAME_LIST                          ; 1761
                                BE  11 001CE          BRB     36$                                             ; 1739
                                52  DD 001D0 41$:     PUSHL   R2                                              ; 1770
00000000G       00              01  FB 001D2          CALLS   #1, DBG$NEXECUTE_SHOW_TASK
                                3E  11 001D9          BRB     49$                                             ; 1561
                00000000G       00  DD 001DB 42$:     PUSHL   DBG$SRC_TERM_WIDTH                              ; 1779
                          EC    A9  9F 001E1          PUSHAB  P.ACF                                           ; 1778
00000000G       00              02  FB 001E4 43$:     CALLS   #2, DBG$PRINT
00000000G       00              00  FB 001EB          CALLS   #0, DBG$NEWLINE                                 ; 1780
                                25  11 001F2          BRB     49$                                             ; 1561
                                7E  D4 001F4 44$:     CLRL    -(SP)                                           ; 1792
                                02  11 001F6          BRB     46$
                                01  DD 001F8 45$:     PUSHL   #1                                              ; 1797
00000000G       00              01  FB 001FA 46$:     CALLS   #1, DBG$SHOW_TYPE
                                16  11 00201          BRB     49$                                             ; 1561
                          08    AC  DD 00203 47$:     PUSHL   MESSAGE_VECT                                    ; 1802
                                52  DD 00206          PUSHL   R2                                              ; 1801
00000000G       00              02  FB 00208          CALLS   #2, DBG$EVENT_SHOW_CANCEL_SEMANTICS
                                    04 0020F          RET
                                52  DD 00210 48$:     PUSHL   R2                                              ; 1807
                00000000G       00  FB 00212          CALLS   #1, DBG$SCR_EXECUTE_SHOWIND_CMD
                                50  D0 00219 49$:     MOVL    #1, R0                                          ; 1817
                                    04 0021C          RET                                                     ; 1819

; Routine Size:  541 bytes,    Routine Base:  DBG$CODE + 0C3A
```

```
;  1692          1820  1 GLOBAL ROUTINE dbg$nexecute_show_key (verb_node, message_vect) =
;  1693          1821  1 !++
;  1694          1822  1 ! Functional Description
;  1695          1823  1 !
;  1696          1824  1 !       This routine performs the action associated with the SHOW KEY command.
;  1697          1825  1 !
;  1698          1826  1 ! Routine Inputs
;  1699          1827  1 !
;  1700          1828  1 !       verb_node -     The head of a command execution tree. This is built
;  1701          1829  1 !                       by the routine DBG$NPARSE_SHOW_KEY, and its structure
;  1702          1830  1 !                       is described in the header of that routine.
;  1703          1831  1 !       message_vect -  An error message vector.
;  1704          1832  1 !
;  1705          1833  1 ! Routine Outputs
;  1706          1834  1 !
;  1707          1835  1 !       Information about key definitions are output to the screen.
;  1708          1836  1 !
;  1709          1837  1 !       The routine value is one of:
;  1710          1838  1 !       sts$k_success - Success code.
;  1711          1839  1 !       sts$k_severe -  Error. An error message vector is contructed.
;  1712          1840  1 !--
;  1713          1841  2     BEGIN
;  1714          1842  2
;  1715          1843  2     MAP
;  1716          1844  2         verb_node : REF dbg$verb_node;
;  1717          1845  2
;  1718          1846  2     LITERAL
;  1719          1847  2         v_key_noecho    = 0,
;  1720          1848  2         v_key_terminate = 1,
;  1721          1849  2         v_key_lockstate = 2,
;  1722          1850  2         v_key_setstate  = 4;
;  1723          1851  2
;  1724          1852  2     LOCAL
;  1725          1853  2         noun_node               : REF dbg$noun_node,    ! Points to a noun node
;  1726          1854  2         adverb_node             : REF dbg$adverb_node,  ! Points to an adverb node
;  1727          1855  2         found,
;  1728          1856  2         dir_flag,
;  1729          1857  2         all_flag,
;  1730          1858  2         brief_flag,
;  1731          1859  2         show_status,
;  1732          1860  2         attributes              : BITVECTOR [32],
;  1733          1861  2         context                 : INITIAL(0),
;  1734          1862  2         temp_if_state_address   : REF dbg$state_name_node,
;  1735          1863  2         if_state_desc_address   : REF dbg$state_name_node,
;  1736          1864  2         desc_ptr                : REF dbg$stg_desc,
;  1737          1865  2
;  1738          1866  2         key_name_desc           : dbg$stg_desc,
;  1739          1867  2         if_state_name_desc      : dbg$stg_desc,
;  1740          1868  2         equiv_name_desc         : dbg$stg_desc,
;  1741          1869  2         state_name_desc         : dbg$stg_desc;
;  1742          1870  2
;  1743          1871  2
;  1744          1872  2         ! Macro to initialize descriptors
;  1745          1873  2         !
;  1746          1874  2         MACRO
;  1747        M 1875  2             initialize_desc (temp_desc) =
;  1748        M 1876  2                 BEGIN
```

```
;  1749        M 1877  2                temp_desc [dsc$w_length] = 0;
;  1750        M 1878  2                temp_desc [dsc$b_dtype]  = dsc$k_dtype_t;
;  1751        M 1879  2                temp_desc [dsc$b_class]  = dsc$k_class_d;
;  1752        M 1880  2                temp_desc [dsc$a_pointer]= 0;
;  1753          1881  2                END %;
;  1754          1882  2
;  1755          1883  2            !+
;  1756          1884  2            !       We will set up the noun and verb pointers and proceed to walk
;  1757          1885  2            ! down the adverb list with the knowledge that the qualifier information
;  1758          1886  2            ! is in order.  After checking the qualifiers, a call is made to the
;  1759          1887  2            ! routine SMG$GET_KEY_DEF to get the key information; if the /ALL
;  1760          1888  2            ! qualifier exits then calls are made to SMG$LIST_KEY_DEFS to get all
;  1761          1889  2            ! the key definitions in the table.  A call is also made for each
;  1762          1890  2            ! state specified by the State qualifier.
;  1763          1891  2            !       Then exit successfully, unless some error was found on the way.
;  1764          1892  2            !-
;  1765          1893  2
;  1766          1894  2            noun_node   = .verb_node [dbg$l_verb_object_ptr];
;  1767          1895  2            adverb_node = .verb_node [dbg$l_verb_adverb_ptr];
;  1768          1896  2
;  1769          1897  2            ! DIRECTORY qualifier
;  1770          1898  2
;  1771          1899  2            dir_flag = .adverb_node [dbg$l_adverb_value];
;  1772          1900  2            adverb_node = .adverb_node [dbg$l_adverb_link];
;  1773          1901  2
;  1774          1902  2            ! ALL qualifier
;  1775          1903  2
;  1776          1904  2            all_flag = .adverb_node [dbg$l_adverb_value];
;  1777          1905  3            IF (.dir_flag) AND (.all_flag)
;  1778          1906  2            THEN
;  1779          1907  2                SIGNAL(dbg$_conflict);
;  1780          1908  2            adverb_node = .adverb_node [dbg$l_adverb_link];
;  1781          1909  2
;  1782          1910  2            ! BRIEF qualifier
;  1783          1911  2
;  1784          1912  2            brief_flag = .adverb_node [dbg$l_adverb_value];
;  1785          1913  3            IF (.dir_flag) AND (.brief_flag)
;  1786          1914  2            THEN
;  1787          1915  2                SIGNAL(dbg$_conflict);
;  1788          1916  2            adverb_node = .adverb_node [dbg$l_adverb_link];
;  1789          1917  2
;  1790          1918  2            ! STATE qualifier (Note: if /STATE=xxxxx was specified the link field will be one)
;  1791          1919  2
;  1792          1920  2            if_state_desc_address = .adverb_node [dbg$l_adverb_value];
;  1793          1921  3            IF (.dir_flag) AND (.adverb_node [dbg$l_adverb_link] EQL 1)
;  1794          1922  2            THEN
;  1795          1923  2                SIGNAL(dbg$_conflict);
;  1796          1924  2            adverb_node = .adverb_node [dbg$l_adverb_link];
;  1797          1925  2
;  1798          1926  2            ! If the /DIRECTORY qualifier exists
;  1799          1927  2            !
;  1800          1928  2
;  1801          1929  2            IF .dir_flag
;  1802          1930  2            THEN
;  1803          1931  3                BEGIN
;  1804          1932  3                if_state_desc_address = 0;
;  1805          1933  3                temp_if_state_address = 0;
```

```
;  1806          1934  2          END;
;  1807          1935  2
;  1808          1936  2      WHILE .dir_flag DO      ! i.e. while true do
;  1809          1937  3          BEGIN
;  1810          1938  3          initialize_desc (if_state_name_desc);
;  1811          1939  3
;  1812          1940  3          ! Get a state name of some key.
;  1813          1941  3          !
;  1814          1942  3          show_status = smg$list_key_defs (dbg$gl_key_table_id,
;  1815          1943  3                                           context,
;  1816          1944  3                                           0,
;  1817          1945  3                                           if_state_name_desc);
;  1818          1946  3          IF NOT .show_status
;  1819          1947  3          THEN
;  1820          1948  3              IF .show_status EQL smg$_nomorekeys
;  1821          1949  3              THEN
;  1822          1950  4                  BEGIN
;  1823          1951  4                  ! Output list of states, if no more new key definitions
;  1824          1952  4                  ! can be found.
;  1825          1953  4                  !
;  1826          1954  4                  WHILE .if_state_desc_address NEQ 0 DO
;  1827          1955  5                      BEGIN
;  1828          1956  5                      dbg$print (UPLIT BYTE (%ASCIC '!AS'),
;  1829          1957  5                                 .if_state_desc_address [dbg$l_state_name_ptr]);
;  1830          1958  5                      dbg$newline();
;  1831          1959  5                      if_state_desc_address = .if_state_desc_address [dbg$l_state_name_link];
;  1832          1960  4                      END;
;  1833          1961  4                  RETURN sts$k_success;
;  1834          1962  4                  END
;  1835          1963  3              ELSE
;  1836          1964  3                  SIGNAL(dbg$_shokeyerr);
;  1837          1965  3
;  1838          1966  3          found = FALSE;
;  1839          1967  3          WHILE .temp_if_state_address NEQ 0 DO
;  1840          1968  4              BEGIN
;  1841          1969  4              ! Look to see if a key has already been found with this state
;  1842          1970  4              ! If so, advance the pointer to the next state-node, else, a
;  1843          1971  4              ! key with this state has been found and a new node need not
;  1844          1972  4              ! be added to the list of state names.
;  1845          1973  4              !
;  1846          1974  4              desc_ptr = .temp_if_state_address [dbg$l_state_name_ptr];
;  1847          1975  4              IF 0 EQL str$compare_eql(.desc_ptr, if_state_name_desc)
;  1848          1976  4              THEN
;  1849          1977  5                  BEGIN
;  1850          1978  5                  found = TRUE;
;  1851          1979  5                  EXITLOOP;
;  1852          1980  5                  END
;  1853          1981  4              ELSE
;  1854          1982  4                  temp_if_state_address = .temp_if_state_address [dbg$l_state_name_link];
;  1855          1983  3              END;
;  1856          1984  3          IF NOT .found
;  1857          1985  3          THEN
;  1858          1986  4              BEGIN
;  1859          1987  4              ! If a key has not been found that has this state yet,
;  1860          1988  4              ! add the state to a list for output.
;  1861          1989  4              !
;  1862          1990  4              temp_if_state_address = dbg$get_tempmem(dbg$k_state_name_size);
```

```
;  1863     1991  4                temp_if_state_address [dbg$l_state_name_link] = .if_state_desc_address;
;  1864     1992  4                if_state_desc_address = .temp_if_state_address;
;  1865     1993  4
;  1866     1994  4                desc_ptr = dbg$get_tempmem(2);
;  1867     1995  4                desc_ptr [dsc$w_length]  = .if_state_name_desc [dsc$w_length];
;  1868     1996  4                desc_ptr [dsc$b_dtype]   = dsc$k_dtype_t;
;  1869     1997  4                desc_ptr [dsc$b_class]   = dsc$k_class_d;
;  1870     1998  4                desc_ptr [dsc$a_pointer] = .if_state_name_desc [dsc$a_pointer];
;  1871     1999  4
;  1872     2000  4                temp_if_state_address [dbg$l_state_name_ptr] = .desc_ptr;
;  1873     2001  3                END;
;  1874     2002  2            END;
;  1875     2003  2
;  1876     2004  2        ! If the /DIRECTORY qualifier does not exist
;  1877     2005  2        !
;  1878     2006  2
;  1879     2007  2        WHILE .if_state_desc_address NEQ 0 DO
;  1880     2008  3            BEGIN
;  1881     2009  3            ! Output the keys for each seperate state that is specified
;  1882     2010  3            !
;  1883     2011  3            dbg$newline();
;  1884     2012  3            dbg$print(UPLIT BYTE (%ASCIC '!AS keypad definitions:'),
;  1885     2013  3                      .if_state_desc_address [dbg$l_state_name_ptr]);
;  1886     2014  3            dbg$newline();
;  1887     2015  3            context = 0;
;  1888     2016  3            WHILE TRUE DO
;  1889     2017  4                BEGIN
;  1890     2018  4                ! Look at each key that has been defined and determine whether
;  1891     2019  4                ! it has the right state to be output in this list.
;  1892     2020  4                !
;  1893     2021  4
;  1894     2022  4                initialize_desc(key_name_desc);
;  1895     2023  4                initialize_desc(if_state_name_desc);
;  1896     2024  4                initialize_desc(state_name_desc);
;  1897     2025  4                initialize_desc(equiv_name_desc);
;  1898     2026  4
;  1899     2027  4                show_status = smg$list_key_defs (dbg$gl_key_table_id,
;  1900     2028  4                                                 context,
;  1901     2029  4                                                 key_name_desc,
;  1902     2030  4                                                 if_state_name_desc,
;  1903     2031  4                                                 attributes,
;  1904     2032  4                                                 equiv_name_desc,
;  1905     2033  4                                                 state_name_desc);
;  1906     2034  4                IF NOT .show_status
;  1907     2035  4                THEN
;  1908     2036  4                    IF .show_status EQL smg$_nomorekeys
;  1909     2037  4                    THEN
;  1910     2038  4                        EXITLOOP
;  1911     2039  4                    ELSE
;  1912     2040  4                        SIGNAL(dbg$_shokeyerr);
;  1913     2041  4
;  1914     2042  4                ! If all keys are to be output, or if just a key with this name,
;  1915     2043  4                ! go into this condition.
;  1916     2044  4                !
;  1917     2045  5                IF (.all_flag) OR (0 EQL str$compare_eql(key_name_desc, .noun_node [dbg$l_noun_value]))
;  1918     2046  4                THEN
;  1919     2047  4                    ! If the state-names match, then this key is to be output.
```

```
;  1920          2048  4          !
;  1921          2049  4          IF 0 EQL str$compare_eql(if_state_name_desc, .if_state_desc_address [dbg$l_state_name_pt
;  1922          2050  4          THEN
;  1923          2051  5              BEGIN
;  1924          2052  5              ! Print out key information
;  1925          2053  5              !
;  1926          2054  5              dbg$print(UPLIT BYTE (%ASCIC '  !AS = "!AS"'), key_name_desc, equiv_name_desc);
;  1927          2055  5              IF NOT .brief_flag
;  1928          2056  5              THEN
;  1929          2057  6                  BEGIN
;  1930          2058  6                  IF .attributes [v_key_noecho]
;  1931          2059  6                  THEN
;  1932          2060  6                      dbg$print(UPLIT BYTE (%ASCIC '  (noecho'))
;  1933          2061  6                  ELSE
;  1934          2062  6                      dbg$print(UPLIT BYTE (%ASCIC '  (echo'));
;  1935          2063  6                  IF .attributes [v_key_terminate]
;  1936          2064  6                  THEN
;  1937          2065  6                      dbg$print(UPLIT BYTE (%ASCIC ',terminate'))
;  1938          2066  6                  ELSE
;  1939          2067  6                      dbg$print(UPLIT BYTE (%ASCIC ',noterminate'));
;  1940          2068  6                  IF .attributes [v_key_lockstate]
;  1941          2069  6                  THEN
;  1942          2070  6                      dbg$print(UPLIT BYTE (%ASCIC ',lock'))
;  1943          2071  6                  ELSE
;  1944          2072  6                      dbg$print(UPLIT BYTE (%ASCIC ',nolock'));
;  1945          2073  6                  IF .attributes [v_key_setstate]
;  1946          2074  6                  THEN
;  1947          2075  6                      dbg$print(UPLIT BYTE (%ASCIC ',state=!AS)'), state_name_desc)
;  1948          2076  6                  ELSE
;  1949          2077  6                      dbg$print(UPLIT BYTE (%ASCIC ')'));
;  1950          2078  5                  END;
;  1951          2079  5              dbg$newline();
;  1952          2080  4              END;
;  1953          2081  3          END;
;  1954          2082  3          ! Move the state-name-pointer ahead one
;  1955          2083  3          !
;  1956          2084  3          if_state_desc_address = .if_state_desc_address [dbg$l_state_name_link];
;  1957          2085  2          END;
;  1958          2086  2
;  1959          2087  2      RETURN sts$k_success;
;  1960          2088  1      END;


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

                                                                53  41  21  03  001AE P.ACH:  .ASCII  <3>\!AS\
66  65  64  20  64  61  70  79  65  6B  20  53  41  21  17  001B2 P.ACI:  .ASCII  <23>\!AS keypad definitions:\
                        3A  73  6E  6F  69  74  69  6E  69  001C1
    22  53  41  21  22  20  3D  20  53  41  21  20  20  0D  001CA P.ACJ:  .ASCII  <13>\  !AS = "!AS"\
                    6F  68  63  65  6F  6E  28  20  20  09  001D8 P.ACK:  .ASCII  <9>\  (noecho\
                    6F  68  63  65  6F  6E  28  20  20  07  001E2 P.ACL:  .ASCII  <7>\  (echo\
                65  74  61  6E  69  6D  72  65  74  2C  0A  001EA P.ACM:  .ASCII  <10>\,terminate\
        65  74  61  6E  69  6D  72  65  74  6F  6E  2C  0C  001F5 P.ACN:  .ASCII  <12>\,noterminate\
                                6B  63  6F  6C  6F  6E  2C  05  00202 P.ACO:  .ASCII  <5>\,lock\
                        6B  63  6F  6C  6F  6E  2C  07  00208 P.ACP:  .ASCII  <7>\,nolock\
            29  53  41  21  3D  65  74  61  74  73  2C  0B  00210 P.ACQ:  .ASCII  <11>\,state=!AS)\
```

```
                              29  01  0021C P.ACR:  .ASCII  <1>\)\                                          ;

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

                                OFFC 00000          .ENTRY  DBG$NEXECUTE_SHOW_KEY, Save R2,R3,R4,R5,R6,-; 1820
                                                            R7,R8,R9,R10,R11                              ;
            5B 00000000'  EF  9E 00002              MOVAB   P.ACH, R11                                    ;
            5E            38  C2 00009              SUBL2   #56, SP                                       ;
                     04   AE  D4 0000C              CLRL    CONTEXT                                       ; 1841
            50       04   AC  D0 0000F              MOVL    VERB_NODE, R0                                 ; 1894
            5A       08   A0  D0 00013              MOVL    8(R0), NOUN_NODE                              ;
            52       04   A0  D0 00017              MOVL    4(R0), ADVERB_NODE                            ; 1895
            57       04   A2  D0 0001B              MOVL    4(ADVERB_NODE), DIR_FLAG                      ; 1899
            52       08   A2  D0 0001F              MOVL    8(ADVERB_NODE), ADVERB_NODE                   ; 1900
            59       04   A2  D0 00023              MOVL    4(ADVERB_NODE), ALL_FLAG                      ; 1904
            10            57  E9 00027              BLBC    DIR_FLAG, 1$                                  ; 1905
            0D            59  E9 0002A              BLBC    ALL_FLAG, 1$                                  ;
               00028158   8F  DD 0002D              PUSHL   #164184                                       ; 1907
 00000000G  00            01  FB 00033              CALLS   #1, LIB$SIGNAL                                ;
            52       08   A2  D0 0003A 1$:          MOVL    8(ADVERB_NODE), ADVERB_NODE                   ; 1908
            58       04   A2  D0 0003E              MOVL    4(ADVERB_NODE), BRIEF_FLAG                    ; 1912
            10            57  E9 00042              BLBC    DIR_FLAG, 2$                                  ; 1913
            0D            58  E9 00045              BLBC    BRIEF_FLAG, 2$                                ;
               00028158   8F  DD 00048              PUSHL   #164184                                       ; 1915
 00000000G  00            01  FB 0004E              CALLS   #1, LIB$SIGNAL                                ;
            52       08   A2  D0 00055 2$:          MOVL    8(ADVERB_NODE), ADVERB_NODE                   ; 1916
            54       04   A2  D0 00059              MOVL    4(ADVERB_NODE), IF_STATE_DESC_ADDRESS         ; 1920
            13            57  E9 0005D              BLBC    DIR_FLAG, 3$                                  ; 1921
            01       08   A2  D1 00060              CMPL    8(ADVERB_NODE), #1                            ;
                          0D  12 00064              BNEQ    3$                                            ;
               00028158   8F  DD 00066              PUSHL   #164184                                       ; 1923
 00000000G  00            01  FB 0006C              CALLS   #1, LIB$SIGNAL                                ;
            52       08   A2  D0 00073 3$:          MOVL    8(ADVERB_NODE), ADVERB_NODE                   ; 1924
            02            57  E9 00077              BLBC    DIR_FLAG, 4$                                  ; 1929
                          53  7C 0007A              CLRQ    TEMP_IF_STATE_ADDRESS                         ; 1933
            03            57  E8 0007C 4$:          BLBS    DIR_FLAG, 5$                                  ; 1936
                        00B6  31 0007F              BRW     14$                                           ;
        20  AE 020E0000   8F  D0 00082 5$:          MOVL    #34471936, IF_STATE_NAME_DESC                 ; 1938
                     24   AE  D4 0008A              CLRL    IF_STATE_NAME_DESC+4                          ;
                     20   AE  9F 0008D              PUSHAB  IF_STATE_NAME_DESC                            ; 1942
                          7E  D4 00090              CLRL    -(SP)                                         ;
                     0C   AE  9F 00092              PUSHAB  CONTEXT                                       ;
               00000000G  00  9F 00095              PUSHAB  DBG$GL_KEY_TABLE_ID                           ;
 00000000G  00            04  FB 0009B              CALLS   #4, SMG$LIST_KEY_DEFS                         ;
            56            50  D0 000A2              MOVL    R0, SHOW_STATUS                               ;
            35            56  E8 000A5              BLBS    SHOW_STATUS, 9$                               ; 1946
 00000000G  8F            56  D1 000A8              CMPL    SHOW_STATUS, #SMG$_NOMOREKEYS                 ; 1948
                          1F  12 000AF              BNEQ    8$                                            ;
                          54  D5 000B1 6$:          TSTL    IF_STATE_DESC_ADDRESS                         ; 1954
                          03  12 000B3              BNEQ    7$                                            ;
                        01AA  31 000B5              BRW     30$                                           ;
                          64  DD 000B8 7$:          PUSHL   (IF_STATE_DESC_ADDRESS)                       ; 1957
                          5B  DD 000BA              PUSHL   R11                                           ; 1956
 00000000G  00            02  FB 000BC              CALLS   #2, DBG$PRINT                                 ;
 00000000G  00            00  FB 000C3              CALLS   #0, DBG$NEWLINE                               ; 1958
```

```
                54        04   A4  DO 000CA          MOVL    4(IF_STATE_DESC_ADDRESS), -                     ; 1959
                                                             IF_STATE_DESC_ADDRESS
                          E1   11 000CE              BRB     6$                                              ; 1954
            00028120      8F   DD 000D0 8$:         PUSHL   #164128                                         ; 1964
00000000G   00            01   FB 000D6              CALLS   #1, LIB$SIGNAL
                          55   D4 000DD 9$:         CLRL    FOUND                                           ; 1966
                          53   D5 000DF 10$:        TSTL    TEMP_IF_STATE_ADDRESS                           ; 1967
                          1E   13 000E1              BEQL    12$
                52        63   D0 000E3              MOVL    (TEMP_IF_STATE_ADDRESS), DESC_PTR               ; 1974
                      20  AE   9F 000E6              PUSHAB  IF_STATE_NAME_DESC                              ; 1975
                          52   DD 000E9              PUSHL   DESC_PTR
00000000G   00            02   FB 000EB              CALLS   #2, STR$COMPARE_EQL
                          50   D5 000F2              TSTL    R0
                          05   12 000F4              BNEQ    11$
                55        01   D0 000F6              MOVL    #1, FOUND                                       ; 1978
                          06   11 000F9              BRB     12$                                             ; 1977
                53        04   A3  DO 000FB 11$:     MOVL    4(TEMP_IF_STATE_ADDRESS), -                     ; 1982
                                                             TEMP_IF_STATE_ADDRESS
                          DE   11 000FF              BRB     10$                                             ; 1967
                31        55   E8 00101 12$:        BLBS    FOUND, 13$                                      ; 1984
                          02   DD 00104              PUSHL   #2                                              ; 1990
00000000G   00            01   FB 00106              CALLS   #1, DBG$GET_TEMPMEM
                53        50   D0 0010D              MOVL    R0, TEMP_IF_STATE_ADDRESS
          04    A3        54   D0 00110              MOVL    IF_STATE_DESC_ADDRESS, -                        ; 1991
                                                             4(TEMP_IF_STATE_ADDRESS)
                54        53   D0 00114              MOVL    TEMP_IF_STATE_ADDRESS, -                        ; 1992
                                                             IF_STATE_DESC_ADDRESS
                          02   DD 00117              PUSHL   #2                                              ; 1994
00000000G   00            01   FB 00119              CALLS   #1, DBG$GET_TEMPMEM
                52        50   D0 00120              MOVL    R0, DESC_PTR
                62        20   AE   B0 00123         MOVW    IF_STATE_NAME_DESC, (DESC_PTR)                  ; 1995
          02    A2      020E   8F   B0 00127         MOVW    #526, 2(DESC_PTR)                               ; 1996
          04    A2        24   AE   D0 0012D         MOVL    IF_STATE_NAME_DESC+4, 4(DESC_PTR)               ; 1998
                63        52   D0 00132              MOVL    DESC_PTR, (TEMP_IF_STATE_ADDRESS)               ; 2000
                        FF44   31 00135 13$:         BRW     4$                                              ; 1936
                          54   D5 00138 14$:         TSTL    IF_STATE_DESC_ADDRESS                           ; 2007
                          03   12 0013A              BNEQ    15$
                        0123   31 0013C              BRW     30$
00000000G   00            00   FB 0013F 15$:         CALLS   #0, DBG$NEWLINE                                 ; 2011
                          64   DD 00146              PUSHL   (IF_STATE_DESC_ADDRESS)                         ; 2013
                      04  AB   9F 00148              PUSHAB  P.ACI                                           ; 2012
00000000G   00            02   FB 0014B              CALLS   #2, DBG$PRINT
00000000G   00            00   FB 00152              CALLS   #0, DBG$NEWLINE                                 ; 2014
                      04  AE   D4 00159              CLRL    CONTEXT                                         ; 2015
          2C    AE  020E0000   8F   D0 0015C 16$:     MOVL    #34471936, KEY_NAME_DESC                        ; 2022
                      30  AE   D4 00164              CLRL    KEY_NAME_DESC+4
          20    AE  020E0000   8F   D0 00167         MOVL    #34471936, IF_STATE_NAME_DESC                   ; 2023
                      24  AE   D4 0016F              CLRL    IF_STATE_NAME_DESC+4
          08    AE  020E0000   8F   D0 00172         MOVL    #34471936, STATE_NAME_DESC                      ; 2024
                      0C  AE   D4 0017A              CLRL    STATE_NAME_DESC+4
          14    AE  020E0000   8F   D0 0017D         MOVL    #34471936, EQUIV_NAME_DESC                      ; 2025
                      18  AE   D4 00185              CLRL    EQUIV_NAME_DESC+4
                      08  AE   9F 00188              PUSHAB  STATE_NAME_DESC                                 ; 2027
                      18  AE   9F 0018B              PUSHAB  EQUIV_NAME_DESC
                      08  AE   9F 0018E              PUSHAB  ATTRIBUTES
                      2C  AE   9F 00191              PUSHAB  IF_STATE_NAME_DESC
                      3C  AE   9F 00194              PUSHAB  KEY_NAME_DESC
```

```
                          18   AE  9F 00197          PUSHAB  CONTEXT                                  ;
                00000000G      00  9F 0019A          PUSHAB  DBG$GL_KEY_TABLE_ID                      ;
00000000G       00             07  FB 001A0          CALLS   #7, SMG$LIST_KEY_DEFS                    ;
                56             50  D0 001A7          MOVL    R0, SHOW_STATUS                          ;
                19             56  E8 001AA          BLBS    SHOW_STATUS, 18$                         ; 2034
00000000G       8F             56  D1 001AD          CMPL    SHOW_STATUS, #SMG$_NOMOREKEYS            ; 2036
                               03  12 001B4          BNEQ    17$                                      ;
                            00A2   31 001B6          BRW     29$                                      ;
                00028120       8F  DD 001B9 17$:     PUSHL   #164128                                  ; 2040
00000000G       00             01  FB 001BF          CALLS   #1, LIB$SIGNAL                           ;
                10             59  E8 001C6 18$:     BLBS    ALL_FLAG, 19$                            ; 2045
                               6A  DD 001C9          PUSHL   (NOUN_NODE)                              ;
                          30   AE  9F 001CB          PUSHAB  KEY_NAME_DESC                            ;
00000000G       00             02  FB 001CE          CALLS   #2, STR$COMPARE_EQL                      ;
                               50  D5 001D5          TSTL    R0                                       ;
                               83  12 001D7          BNEQ    16$                                      ;
                               64  DD 001D9 19$:     PUSHL   (IF_STATE_DESC_ADDRESS)                  ; 2049
                          24   AE  9F 001DB          PUSHAB  IF_STATE_NAME_DESC                       ;
00000000G       00             02  FB 001DE          CALLS   #2, STR$COMPARE_EQL                      ;
                               50  D5 001E5          TSTL    R0                                       ;
                               6F  12 001E7          BNEQ    28$                                      ;
                          14   AE  9F 001E9          PUSHAB  EQUIV_NAME_DESC                          ; 2054
                          30   AE  9F 001EC          PUSHAB  KEY_NAME_DESC                            ;
                          1C   AB  9F 001EF          PUSHAB  P.ACJ                                    ;
00000000G       00             03  FB 001F2          CALLS   #3, DBG$PRINT                            ;
                55             58  E8 001F9          BLBS    BRIEF_FLAG, 27$                          ; 2055
                05             6E  E9 001FC          BLBC    ATTRIBUTES, 20$                          ; 2058
                          2A   AB  9F 001FF          PUSHAB  P.ACK                                    ; 2060
                               03  11 00202          BRB     21$                                      ;
                          34   AB  9F 00204 20$:     PUSHAB  P.ACL                                    ; 2062
     00000000G  00             01  FB 00207 21$:     CALLS   #1, DBG$PRINT                            ;
05                  6E         01  E1 0020E          BBC     #1, ATTRIBUTES, 22$                      ; 2063
                          3C   AB  9F 00212          PUSHAB  P.ACM                                    ; 2065
                               03  11 00215          BRB     23$                                      ;
                          47   AB  9F 00217 22$:     PUSHAB  P.ACN                                    ; 2067
     00000000G  00             01  FB 0021A 23$:     CALLS   #1, DBG$PRINT                            ;
05                  6E         02  E1 00221          BBC     #2, ATTRIBUTES, 24$                      ; 2068
                          54   AB  9F 00225          PUSHAB  P.ACO                                    ; 2070
                               03  11 00228          BRB     25$                                      ;
                          5A   AB  9F 0022A 24$:     PUSHAB  P.ACP                                    ; 2072
     00000000G  00             01  FB 0022D 25$:     CALLS   #1, DBG$PRINT                            ;
0F                  6E         04  E1 00234          BBC     #4, ATTRIBUTES, 26$                      ; 2073
                          08   AE  9F 00238          PUSHAB  STATE_NAME_DESC                          ; 2075
                          62   AB  9F 0023B          PUSHAB  P.ACQ                                    ;
     00000000G  00             02  FB 0023E          CALLS   #2, DBG$PRINT                            ;
                               0A  11 00245          BRB     27$                                      ;
                          6E   AB  9F 00247 26$:     PUSHAB  P.ACR                                    ; 2077
     00000000G  00             01  FB 0024A          CALLS   #1, DBG$PRINT                            ;
     00000000G  00             00  FB 00251 27$:     CALLS   #0, DBG$NEWLINE                          ; 2079
                            FF01   31 00258 28$:     BRW     16$                                      ; 2016
                54         04   A4  D0 0025B 29$:     MOVL    4(IF_STATE_DESC_ADDRESS), -              ; 2084
                                                                IF_STATE_DESC_ADDRESS                    ;
                            FED6   31 0025F          BRW     14$                                      ; 2007
                50             01  D0 00262 30$:     MOVL    #1, R0                                   ; 2087
                                   04 00265          RET                                              ; 2088

; Routine Size:  614 bytes,    Routine Base:  DBG$CODE + 0E57
```

```
;  1962          2089  1 GLOBAL ROUTINE DBG$NSHOW_MARGINS : NOVALUE =
;  1963          2090  1 !++
;  1964          2091  1 ! FUNCTION
;  1965          2092  1 !
;  1966          2093  1 !       This routine implements the SHOW MARGINS command.
;  1967          2094  1 !
;  1968          2095  1 ! INPUTS
;  1969          2096  1 !
;  1970          2097  1 !       The global variables DBG$SRC_LEFT_MARGIN and DBG$SRC_RIGHT_MARGIN.
;  1971          2098  1 !
;  1972          2099  1 ! OUTPUTS
;  1973          2100  1 !
;  1974          2101  1 !       Margin settings are displayed at the terminal.
;  1975          2102  1 !
;  1976          2103  1 !--
;  1977          2104  2     BEGIN
;  1978          2105  2
;  1979          2106  2     ! Set up the output buffer
;  1980          2107  2     !
;  1981          2108  2     dbg$flushbuf();
;  1982          2109  2
;  1983          2110  2     dbg$print (UPLIT BYTE(%ASCIC 'left margin: !UL , right margin: !UL'),
;  1984          2111  2                .dbg$src_left_margin, .dbg$src_right_margin);
;  1985          2112  2     dbg$newline();
;  1986          2113  2
;  1987          2114  1     END; ! dbg$nshow_margins


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

21 20 3A 6E 69 67 72 61 6D 20 74 66 65 6C 24  0021E P.ACS:  .ASCII  \$left margin: !UL , right margin: !UL\      ;
67 72 61 6D 20 74 68 67 69 72 20 2C 20 4C 55  0022D                                                           ;
                       4C 55 21 20 3A 6E 69   0023C                                                           ;


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

                                                 0000 00000     .ENTRY  DBG$NSHOW_MARGINS, Save nothing        ; 2089
                    00000000G  00                00  FB 00002  CALLS   #0, DBG$FLUSHBUF                      ; 2108
                               00000000G         00  DD 00009  PUSHL   DBG$SRC_RIGHT_MARGIN                  ; 2111
                               00000000G         00  DD 0000F  PUSHL   DBG$SRC_LEFT_MARGIN                   ;
                               00000000'         EF  9F 00015  PUSHAB  P.ACS                                ; 2110
                    00000000G  00                03  FB 0001B  CALLS   #3, DBG$PRINT                        ;
                    00000000G  00                00  FB 00022  CALLS   #0, DBG$NEWLINE                      ; 2112
                                                 04 00029      RET                                          ; 2114

; Routine Size:  42 bytes,    Routine Base:  DBG$CODE + 10BD
```

```
;  1989          2115  1 GLOBAL ROUTINE DBG$NSHOW_MAX_SOURCE_FILES : NOVALUE =
;  1990          2116  1 !++
;  1991          2117  1 ! FUNCTION
;  1992          2118  1 !
;  1993          2119  1 !       This routine implements the SHOW MAX_SOURCE_FILES command.
;  1994          2120  1 !
;  1995          2121  1 ! INPUTS
;  1996          2122  1 !
;  1997          2123  1 !       The global variable DBG$SRC_MAX_SOURCE_FILES.
;  1998          2124  1 !
;  1999          2125  1 ! OUTPUTS
;  2000          2126  1 !
;  2001          2127  1 !       The value is displayed at the terminal.
;  2002          2128  1 !
;  2003          2129  1 !--
;  2004          2130  2     BEGIN
;  2005          2131  2
;  2006          2132  2     ! Set up the output buffer
;  2007          2133  2     !
;  2008          2134  2     dbg$flushbuf();
;  2009          2135  2
;  2010          2136  2     dbg$print (UPLIT BYTE(%ASCIC 'max_source_files: !UL'),
;  2011          2137  2                .dbg$src_max_files);
;  2012          2138  2     dbg$newline();
;  2013          2139  2
;  2014          2140  1     END; ! dbg$nshow_max_source_files


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
6C  69  66  5F  65  63  72  75  6F  73  5F  78  61  6D  15  00243 P.ACT:  .ASCII  <21>\max_source_files: !UL\                    ;
                                4C  55  21  20  3A  73  65  00252                                                            ;


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

                                                          0000 00000        .ENTRY  DBG$NSHOW_MAX_SOURCE_FILES, Save nothing    ; 2115
                        00000000G  00                  00  FB 00002        CALLS   #0, DBG$FLUSHBUF                            ; 2134
                                          00000000G    00  DD 00009        PUSHL   DBG$SRC_MAX_FILES                           ; 2137
                                          00000000'    EF  9F 0000F        PUSHAB  P.ACT                                       ; 2136
                        00000000G  00                  02  FB 00015        CALLS   #2, DBG$PRINT                               ;
                        00000000G  00                  00  FB 0001C        CALLS   #0, DBG$NEWLINE                             ; 2138
                                                       04 00023        RET                                                 ; 2140

; Routine Size:  36 bytes,    Routine Base:  DBG$CODE + 10E7
```

```
;  2016     2141  1 GLOBAL ROUTINE DBG$NSHOW_OUTPUT (FULL_REP) : NOVALUE =
;  2017     2142  1 !
;  2018     2143  1 ! FUNCTION
;  2019     2144  1 !       This routine prints the output for the SHOW OUTPUT and SHOW LOG
;  2020     2145  1 !       commands.  The output for the SHOW LOG command is a subset of the
;  2021     2146  1 !       output for the SHOW OUTPUT command, for which reason both commands
;  2022     2147  1 !       are handled in the same routine.
;  2023     2148  1 !
;  2024     2149  1 ! INPUTS
;  2025     2150  1 !       FULL_REP - Equals 1 for a SHOW OUTPUT report and equals 2 for SHOW LOG.
;  2026     2151  1 !
;  2027     2152  1 ! OUTPUTS
;  2028     2153  1 !       NONE
;  2029     2154  1 !
;  2030     2155  1
;  2031     2156  2     BEGIN
;  2032     2157  2
;  2033     2158  2     BIND
;  2034     2159  2         DEFLOG_NAME = UPLIT BYTE ('DEBUG.LOG'),
;  2035     2160  2         DEFLOG_SIZE = %CHARCOUNT ('DEBUG.LOG');
;  2036     2161  2
;  2037     2162  2     LOCAL
;  2038     2163  2         FNAME_LEN,                      ! Length of log file's file name
;  2039     2164  2         FNAME_PTR;                      ! Pointer to log file's file name
;  2040     2165  2
;  2041     2166  2
;  2042     2167  2
;  2043     2168  2     ! If a SHOW OUTPUT command was entered, we print the full representation of
;  2044     2169  2     ! the output settings.  We start by printing the settings of the VERIFY,
;  2045     2170  2     ! TERMINAL, and SCREEN_LOG switches.
;  2046     2171  2     !
;  2047     2172  2     IF .FULL_REP
;  2048     2173  2     THEN
;  2049     2174  2         BEGIN
;  2050     2175  2
;  2051     2176  2
;  2052     2177  2         ! Print the "verify" or "noverify" switch setting.
;  2053     2178  2         !
;  2054     2179  2         IF NOT .DBG$GB_DEF_OUT[OUT_VERIFY]
;  2055     2180  2         THEN
;  2056     2181  2             DBG$PRINT(UPLIT BYTE(%ASCIC 'no'), 0);
;  2057     2182  2
;  2058     2183  2         DBG$PRINT(UPLIT BYTE(%ASCIC 'verify, '), 0);
;  2059     2184  2
;  2060     2185  2
;  2061     2186  2         ! Print the "terminal" or "noterminal" switch setting.
;  2062     2187  2         !
;  2063     2188  2         IF NOT .DBG$GB_DEF_OUT[OUT_TERM]
;  2064     2189  2         THEN
;  2065     2190  2             DBG$PRINT(UPLIT BYTE(%ASCIC 'no'), 0);
;  2066     2191  2
;  2067     2192  2         DBG$PRINT(UPLIT BYTE(%ASCIC 'terminal, '), 0);
;  2068     2193  2
;  2069     2194  2
;  2070     2195  2         ! Print the "screen_log" or "noscreen_log" switch setting.
;  2071     2196  2         !
;  2072     2197  3         IF NOT .DBG$GB_DEF_OUT[OUT_SCREEN]
```

```
;  2073          2198  3          THEN
;  2074          2199  3              DBG$PRINT(UPLIT BYTE(%ASCIC 'no'), 0);
;  2075          2200  3
;  2076          2201  3          DBG$PRINT(UPLIT BYTE(%ASCIC 'screen_log, '), 0);
;  2077          2202  2          END;
;  2078          2203  2
;  2079          2204  2
;  2080          2205  2      ! Now print whether we are logging or not and print the name of the current
;  2081          2206  2      ! log file.  If log file has been specified, we report the file name in the
;  2082          2207  2      ! NAM block; otherwise, we use the default log-file file name.  Note that
;  2083          2208  2      ! this output is done for both the SHOW LOG and SHOW OUTPUT commands.
;  2084          2209  2      !
;  2085          2210  2      IF .DBG$GL_LOGFAB[FAB$W_IFI] LEQ 0
;  2086          2211  2      THEN
;  2087          2212  2          BEGIN
;  2088          2213  3          FNAME_PTR = DEFLOG_NAME;
;  2089          2214  3          FNAME_LEN = DEFLOG_SIZE;
;  2090          2215  3          END
;  2091          2216  2
;  2092          2217  2      ELSE
;  2093          2218  2          BEGIN
;  2094          2219  3          FNAME_PTR = .DBG$GL_LOGNAM[NAM$L_RSA];
;  2095          2220  3          FNAME_LEN = .DBG$GL_LOGNAM[NAM$B_RSL];
;  2096          2221  3          END;
;  2097          2222  2
;  2098          2223  2      IF NOT .DBG$GB_DEF_OUT[OUT_LOG]
;  2099          2224  2      THEN
;  2100          2225  2          DBG$PRINT(UPLIT BYTE(%ASCIC 'not '), 0);
;  2101          2226  2
;  2102          2227  2      DBG$PRINT(UPLIT BYTE(%ASCIC 'logging to !AD'), .FNAME_LEN, .FNAME_PTR);
;  2103          2228  2
;  2104          2229  2
;  2105          2230  2      ! Finally close out the print line and return.
;  2106          2231  2      !
;  2107          2232  2      DBG$NEWLINE();
;  2108          2233  2      RETURN;
;  2109          2234  2
;  2110          2235  1      END;


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

                                  47  4F  4C  2E  47  55  42  45  44  00259 P.ACU:  .ASCII  \DEBUG.LOG\                          ;
                                                                  6F  6E  02  00262 P.ACV:  .ASCII  <2>\no\
                                  20  2C  79  66  69  72  65  76  08  00265 P.ACW:  .ASCII  <8>\verify, \
                                                                  6F  6E  02  0026E P.ACX:  .ASCII  <2>\no\
                              20  2C  6C  61  6E  69  6D  72  65  74  0A  00271 P.ACY:  .ASCII  <10>\terminal, \
                                                                  6F  6E  02  0027C P.ACZ:  .ASCII  <2>\no\
                          20  2C  67  6F  6C  5F  6E  65  65  72  63  73  0C  0027F P.ADA:  .ASCII  <12>\screen_log, \
                                                                  20  74  6F  6E  04  0028C P.ADB:  .ASCII  <4>\not \
                  44  41  21  20  6F  74  20  67  6E  69  67  67  6F  6C  0E  00291 P.ADC:  .ASCII  <14>\logging to !AD\          ;

                                                                                  DEFLOG_NAME=        P.ACU
                                                                                  DEFLOG_SIZE=        9
```

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

                                 007C 00000     .ENTRY  DBG$NSHOW_OUTPUT, Save R2,R3,R4,R5,R6       ; 2141
56 00000000G   00    9E 00002     MOVAB   DBG$GB_DEF_OUT+2, R6
55 00000000G   00    9E 00009     MOVAB   DBG$PRINT, R5
54 00000000'   EF    9E 00010     MOVAB   P.ACV, R4
3A       04    AC    E9 00017     BLBC    FULL_REP, 4$                                            ; 2172
07             66    E8 0001B     BLBS    DBG$GB_DEF_OUT+2, 1$                                    ; 2179
               7E    D4 0001E     CLRL    -(SP)                                                   ; 2181
               54    DD 00020     PUSHL   R4
65             02    FB 00022     CALLS   #2, DBG$PRINT
               7E    D4 00025 1$: CLRL    -(SP)                                                   ; 2183
          03   A4    9F 00027     PUSHAB  P.ACW
65             02    FB 0002A     CALLS   #2, DBG$PRINT
08       FF    A6    E8 0002D     BLBS    DBG$GB_DEF_OUT+1, 2$                                    ; 2188
               7E    D4 00031     CLRL    -(SP)                                                   ; 2190
          0C   A4    9F 00033     PUSHAB  P.ACX
65             02    FB 00036     CALLS   #2, DBG$PRINT
               7E    D4 00039 2$: CLRL    -(SP)                                                   ; 2192
          0F   A4    9F 0003B     PUSHAB  P.ACY
65             02    FB 0003E     CALLS   #2, DBG$PRINT
08       01    A6    E8 00041     BLBS    DBG$GB_DEF_OUT+3, 3$                                    ; 2197
               7E    D4 00045     CLRL    -(SP)                                                   ; 2199
          1A   A4    9F 00047     PUSHAB  P.ACZ
65             02    FB 0004A     CALLS   #2, DBG$PRINT
               7E    D4 0004D 3$: CLRL    -(SP)                                                   ; 2201
          1D   A4    9F 0004F     PUSHAB  P.ADA
65             02    FB 00052     CALLS   #2, DBG$PRINT
   00000000G   00    B5 00055 4$: TSTW    DBG$GL_LOGFAB+2                                         ; 2210
               09    12 0005B     BNEQ    5$
52       F7    A4    9E 0005D     MOVAB   DEFLOG_NAME, FNAME_PTR                                  ; 2213
53             09    D0 00061     MOVL    #9, FNAME_LEN                                           ; 2214
               0F    11 00064     BRB     6$                                                      ; 2210
50 00000000G   00    D0 00066 5$: MOVL    DBG$GL_LOGNAM, R0                                       ; 2219
52       04    A0    D0 0006D     MOVL    4(R0), FNAME_PTR
53       03    A0    9A 00071     MOVZBL  3(R0), FNAME_LEN                                        ; 2220
08       FE    A6    E8 00075 6$: BLBS    DBG$GB_DEF_OUT, 7$                                      ; 2223
               7E    D4 00079     CLRL    -(SP)                                                   ; 2225
          2A   A4    9F 0007B     PUSHAB  P.ADB
65             02    FB 0007E     CALLS   #2, DBG$PRINT
               52    DD 00081 7$: PUSHL   FNAME_PTR                                               ; 2227
               53    DD 00083     PUSHL   FNAME_LEN
          2F   A4    9F 00085     PUSHAB  P.ADC
65             03    FB 00088     CALLS   #3, DBG$PRINT
   00000000G   00          00 FB 0008B     CALLS   #0, DBG$NEWLINE                                 ; 2232
                                04 00092     RET                                                 ; 2235

; Routine Size:  147 bytes,    Routine Base:  DBG$CODE + 110B
```

```
;  2112          2236  1 GLOBAL ROUTINE DBG$SHOW_RADIX(OVERRIDE_FLAG): NOVALUE =
;  2113          2237  1 !
;  2114          2238  1 ! FUNCTION
;  2115          2239  1 !       Displays the radix that was set by SET RADIX or SET RADIX/OVERRIDE.
;  2116          2240  1 !
;  2117          2241  1 ! INPUTS
;  2118          2242  1 !       OVERRIDE_FLAG - 1 if SHOW RADIX/OVERRIDE was specified
;  2119          2243  1 !
;  2120          2244  1 ! OUTPUTS
;  2121          2245  1 !       The radix setting is written to the output stream.
;  2122          2246  1 !
;  2123          2247  2     BEGIN
;  2124          2248  2     LOCAL
;  2125          2249  2         RADIX;
;  2126          2250  2     ROUTINE DISPLAY_RADIX (IN_RADIX) : NOVALUE =
;  2127          2251  3         BEGIN
;  2128          2252  3         LOCAL
;  2129          2253  3             RADIX;
;  2130          2254  3         RADIX = DBG$NGET_TRANS_RADIX (.IN_RADIX);
;  2131          2255  3         CASE .RADIX FROM DBG$K_DEFAULT TO DBG$K_HEX OF
;  2132          2256  3             SET
;  2133          2257  3             [DBG$K_BINARY]:
;  2134          2258  3                 DBG$PRINT (UPLIT BYTE( %ASCIC 'binary'));
;  2135          2259  3             [DBG$K_OCTAL]:
;  2136          2260  3                 DBG$PRINT (UPLIT BYTE( %ASCIC 'octal'));
;  2137          2261  3             [DBG$K_DECIMAL]:
;  2138          2262  3                 DBG$PRINT (UPLIT BYTE (%ASCIC 'decimal'));
;  2139          2263  3             [DBG$K_HEX]:
;  2140          2264  3                 DBG$PRINT (UPLIT BYTE (%ASCIC 'hexadecimal'));
;  2141          2265  3             [INRANGE, OUTRANGE]:
;  2142          2266  3                 $DBG_ERROR('DBGNSHOW\DBG$SHOW_RADIX');
;  2143          2267  3             TES;
;  2144          2268  2         END;


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

                                79  72  61  6E  69  62  06  002A0 P.ADD:  .ASCII  <6>\binary\                                ;
                                    6C  61  74  63  6F  05  002A7 P.ADE:  .ASCII  <5>\octal\                                 ;
                            6C  61  6D  69  63  65  64  07  002AD P.ADF:  .ASCII  <7>\decimal\                                 ;
                        6C  61  6D  69  63  65  64  61  78  65  68  0B  002B5 P.ADG:  .ASCII  <11>\hexadecimal\                    ;
53  24  47  42  44  5C  57  4F  48  53  4E  47  42  44  17  002C1 P.ADH:  .ASCII  <23>\DBGNSHOW\<92>\DBG$SHOW_RADIX\           ;
                        58  49  44  41  52  5F  57  4F  48  002D0                                                              ;


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

                                                    0004 00000 DISPLAY_RADIX:
                                                                .WORD   Save R2                                             ; 2250
                                52 00000000'    EF  9E 00002            MOVAB   P.ADH, R2                                   ;
                                           04   AC  DD 00009            PUSHL   IN_RADIX                                    ; 2254
                        00000000G   00          01  FB 0000C            CALLS   #1, DBG$NGET_TRANS_RADIX                    ;
                    0F              01          50  CF 00013            CASEL   RADIX, #1, #T5                              ; 2255
            0020            0020        0032        0020    00017 1$:   .WORD   2$-1$,-                                     ;
            0037            0020        0020        0020    0001F               3$-1$,-                                     ;
```

```
                0020            0020            003C            0020        00027                   2$-1$,-                                 ;
                0041            0020            0020            0020        0002F                   2$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    4$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    5$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    2$-1$,-                                 ;
                                                                                                    6$-1$                                   ;
                                                    52  DD  00037 2$:       PUSHL   R2                                                  ; 2266
                                                    01  DD  00039           PUSHL   #1
                                        00028362    8F  DD  0003B           PUSHL   #164706
                        00000000G   00              03  FB  00041           CALLS   #3, LIB$SIGNAL
                                                    04  00048             RET
                                                DF  A2  9F  00049 3$:       PUSHAB  P.ADD                                               ; 2258
                                                    0D  11  0004C           BRB     7$
                                                E6  A2  9F  0004E 4$:       PUSHAB  P.ADE                                               ; 2260
                                                    08  11  00051           BRB     7$
                                                EC  A2  9F  00053 5$:       PUSHAB  P.ADF                                               ; 2262
                                                    03  11  00056           BRB     7$
                                                F4  A2  9F  00058 6$:       PUSHAB  P.ADG                                               ; 2264
                        00000000G   00              01  FB  0005B 7$:       CALLS   #1, DBG$PRINT
                                                    04  00062             RET                                                             ; 2268

; Routine Size:  99 bytes,    Routine Base:  DBG$CODE + 119E

;  2145            2269  2     IF NOT .OVERRIDE_FLAG
;  2146            2270  2     THEN
;  2147            2271  3         BEGIN
;  2148            2272  3         RADIX = .DBG$GB_RADIX[DBG$B_RADIX_INPUT];
;  2149            2273  3         DBG$PRINT(UPLIT BYTE(%ASCIC 'input radix : '));
;  2150            2274  3         DISPLAY_RADIX (.RADIX);
;  2151            2275  3         DBG$NEWLINE();
;  2152            2276  3         RADIX = .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT_OVER];
;  2153            2277  3         IF .RADIX EQL DBG$K_DEFAULT
;  2154            2278  3         THEN
;  2155            2279  3             RADIX = .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT];
;  2156            2280  3         DBG$PRINT(UPLIT BYTE(%ASCIC 'output radix: '));
;  2157            2281  3         DISPLAY_RADIX (.RADIX);
;  2158            2282  3         IF .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT_OVER] NEQ DBG$K_DEFAULT
;  2159            2283  3         THEN
;  2160            2284  3             DBG$PRINT(UPLIT BYTE(%ASCIC ' (override)'));
;  2161            2285  3         DBG$NEWLINE();
;  2162            2286  3         END
;  2163            2287  2     ELSE
;  2164            2288  3         BEGIN
;  2165            2289  3         DBG$PRINT(UPLIT BYTE(%ASCIC 'output override radix: '));
;  2166            2290  3         RADIX = .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT_OVER];
;  2167            2291  3         IF .RADIX NEQ DBG$K_DEFAULT
;  2168            2292  3         THEN
;  2169            2293  3             DISPLAY_RADIX(.RADIX)
```

```
;  2170          2294  3          ELSE
;  2171          2295  3              DBG$PRINT(UPLIT BYTE(%ASCIC 'none'));
;  2172          2296  3          DBG$NEWLINE();
;  2173          2297  2          END;
;  2174          2298  1      END;



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

20 3A 20 78 69 64 61 72 20 74 75 70 6E 69 0E  002D9 P.ADI:   .ASCII  <14>\input radix : \                    ;
20 3A 78 69 64 61 72 20 74 75 70 74 75 6F 0E  002E8 P.ADJ:   .ASCII  <14>\output radix: \                   ;
      29 65 64 69 72 72 65 76 6F 28 20 0B     002F7 P.ADK:   .ASCII  <11>\ (override)\                     ;
64 69 72 72 65 76 6F 20 74 75 70 74 75 6F 17  00303 P.ADL:   .ASCII  <23>\output override radix: \          ;
                  20 3A 78 69 64 61 72 20 65  00312
                        65 6E 6F 6E 04        0031B P.ADM:   .ASCII  <4>\none\                            ;



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

                                                00FC 00000      .ENTRY  DBG$SHOW_RADIX, Save R2,R3,R4,R5,R6,R7      ; 2236
                                    57      98  AF   9E 00002  MOVAB   DISPLAY_RADIX, R7                          ;
                                    56 00000000G 00   9E 00006  MOVAB   DBG$NEWLINE, R6                          ;
                                    55 00000000G 00   9E 0000D  MOVAB   DBG$PRINT, R5                           ;
                                    54 00000000G 00   9E 00014  MOVAB   DBG$GB_RADIX+2, R4                      ;
                                    53 00000000' EF   9E 0001B  MOVAB   P.ADI, R3                               ;
                                    32      04   AC   E8 00022  BLBS    OVERRIDE_FLAG, 2$                     ; 2269
                                    52      FE   A4   9A 00026  MOVZBL  DBG$GB_RADIX, RADIX                   ; 2272
                                                 53   DD 0002A  PUSHL   R3                                     ; 2273
                                    65           01   FB 0002C  CALLS   #1, DBG$PRINT                         ;
                                                 52   DD 0002F  PUSHL   RADIX                                  ; 2274
                                    67           01   FB 00031  CALLS   #1, DISPLAY_RADIX                     ;
                                    66           00   FB 00034  CALLS   #0, DBG$NEWLINE                       ; 2275
                                    52           64   9A 00037  MOVZBL  DBG$GB_RADIX+2, RADIX                  ; 2276
                                    01           52   D1 0003A  CMPL    RADIX, #1                             ; 2277
                                                 04   12 0003D  BNEQ    1$                                    ;
                                    52      FF   A4   9A 0003F  MOVZBL  DBG$GB_RADIX+1, RADIX                  ; 2279
                                            0F   A3   9F 00043  1$:     PUSHAB  P.ADJ                         ; 2280
                                    65           01   FB 00046  CALLS   #1, DBG$PRINT                         ;
                                                 52   DD 00049  PUSHL   RADIX                                  ; 2281
                                    67           01   FB 0004B  CALLS   #1, DISPLAY_RADIX                     ;
                                    01           64   91 0004E  CMPB    DBG$GB_RADIX+2, #1                    ; 2282
                                                 20   13 00051  BEQL    5$                                    ;
                                            1E   A3   9F 00053  PUSHAB  P.ADK                                 ; 2284
                                                 18   11 00056  BRB     4$                                    ;
                                            2A   A3   9F 00058  2$:     PUSHAB  P.ADL                         ; 2289
                                    65           01   FB 0005B  CALLS   #1, DBG$PRINT                         ;
                                    52           64   9A 0005E  MOVZBL  DBG$GB_RADIX+2, RADIX                  ; 2290
                                    01           52   D1 00061  CMPL    RADIX, #1                             ; 2291
                                                 07   13 00064  BEQL    3$                                    ;
                                                 52   DD 00066  PUSHL   RADIX                                  ; 2293
                                    67           01   FB 00068  CALLS   #1, DISPLAY_RADIX                     ;
                                                 06   11 0006B  BRB     5$                                    ;
                                            42   A3   9F 0006D  3$:     PUSHAB  P.ADM                         ; 2295
                                    65           01   FB 00070  4$:     CALLS   #1, DBG$PRINT                 ;
                                    66           00   FB 00073  5$:     CALLS   #0, DBG$NEWLINE               ; 2296
```

```
                                        04 00076        RET                                             ; 2298

; Routine Size:  119 bytes,    Routine Base:  DBG$CODE + 1201

; 2175          2299  1 END             ! End of module
; 2176          2300  0 ELUDOM



                                                                .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  DBG$PLIT                      800  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                     4728  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                           Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619        15         0        1000        00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1
;                                        32         0         0           7        00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1
;                                      1545        46         2          97        00:02.0
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                       418         1         0          31        00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1
;                                       386        15         3          22        00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1
;                                       150         8         5          12        00:00.3



;                               COMMAND QUALIFIERS

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

; Size:          4728 code + 800 data bytes
; Run Time:        01:21.2
; Elapsed Time:    03:51.4
; Lines/CPU Min:    1699
; Lexemes/CPU-Min: 11570
; Memory Used:  541 pages
; Compilation Complete
```

DBGNTYPE
LIS

DBGNSHOW
LIS

DBGPARSER
LIS