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

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

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

```
;     1          0001  0 MODULE DBGNPARSE (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1 !
;     5          0005  1 !**************************************************************************
;     6          0006  1 !*                                                                        *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                  *
;    10          0010  1 !*                                                                        *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                           *
;    17          0017  1 !*                                                                        *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                           *
;    21          0021  1 !*                                                                        *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !**************************************************************************
;    27          0027  1 !
;    28          0028  1 !
;    29          0029  1 ! WRITTEN BY
;    30          0030  1 !       David Plummer   April 15, 1980
;    31          0031  1 !
;    32          0032  1 ! MODIFIED BY
;    33          0033  1 !       Rich Title
;    34          0034  1 !       Vicki Holt
;    35          0035  1 !       Bert Beander
;    36          0036  1 !
;    37          0037  1 ! MODULE FUNCTION
;    38          0038  1 !       This module contains the highest level parse network, DBG$NPARSE_CMD, and
;    39          0039  1 !       several parsing associated routines. Legal command verbs are recognized by
;    40          0040  1 !       DBG$NPARSE_CMD and control is passed to the subnetwork responsible for parsing
;    41          0041  1 !       the rest of the input associated with the verb recognized. In total, the
;    42          0042  1 !       routines produce a command execution tree which is the version 3 debugg-
;    43          0043  1 !       form of intermediate code.
;    44          0044  1 !
;    45          0045  1 !       Also contained in this module are the routines DBG$NPARSE_ADDRESS and
;    46          0046  1 !       DBG$NPARSE_EXPRESSION which are the interfaces between the debugger
;    47          0047  1 !       parser and the Address Expression Interpreter and Expression Interpreter.
;    48          0048  1 !
;    49          0049  1
;    50          0050  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    51          0184  1
;    52          0185  1 FORWARD ROUTINE
;    53          0186  1     DBG$NPARSE_CMD,                     ! Highest level network for parsing
;    54          0187  1     DBG$NMATCH,                         ! String matching routine
;    55          0188  1     DBG$NNEXT_WORD,                     ! Produces a counted string from the input
;    56          0189  1     DBG$NPARSE_EXPRESSION,              ! Interface to expression interpreters
;    57          0190  1     DBG$NPARSE_ADDRESS,                 ! Interface to address expression interpreters
```

```
;    58          0191  1     DBG$NSAVE_DECIMAL_INTEGER,      ! Converts ASCII input to integer
;    59          0192  1     DBG$[illegible]_STRING,         ! Stores a string from input
;    60          0193  1     DBG[illegible]_DIR_LIST,        ! Parses a directory list.
;    61          0194  1     DBG$EXPAND_DEFINE_NAME,         ! Expands name defined with DEFINE
;    62          0195  1     DBG$SYNTAX_ERROR: NOVALUE;      ! Signal a syntax error in command
```

```
;    64          0196  1 EXTERNAL ROUTINE
;    65          0197  1     DBG$ADDR_EXP_INT,          ! Address Expression Interpreter
;    66          0198  1     DBG$DEF_SYM_FIND,          ! Look up a symbol in the
;    67          0199  1                                !     DEFINE symbol table
;    68          0200  1     DBG$GET_MEMORY,            ! Allocates space
;    69          0201  1     DBG$GET_TEMPMEM,           ! Allocates and lists dynamic storage
;    70          0202  1     DBG$EXP_INT,               ! Expression Interpreter
;    71          0203  1     DBG$NPARSE_ALLOCATE,       ! ALLOCATE command parse network
;    72          0204  1     DBG$NPARSE_AT_SIGN,        ! @ filespec parse network
;    73          0205  1     DBG$NPARSE_ATTACH,         ! ATTACH command parse network
;    74          0206  1     DBG$NPARSE_CALL,           ! CALL command parse network
;    75          0207  1     DBG$NPARSE_CANCEL,         ! CANCEL command parse network
;    76          0208  1     DBG$NPARSE_DECLARE,        ! DECLARE command parse network
;    77          0209  1     DBG$NPARSE_DEFINE,         ! DEFINE command parse network
;    78          0210  1     DBG$NPARSE_DELETE,         ! DELETE/KEY command
;    79          0211  1     DBG$NPARSE_DEPOSIT,        ! DEPOSIT command parse network
;    80          0212  1     DBG$NPARSE_DUMP,           ! DUMP command parse network
;    81          0213  1     DBG$NPARSE_EDIT,           ! EDIT command parse network
;    82          0214  1     DBG$NPARSE_EVALUATE,       ! EVALUATE command parse network
;    83          0215  1     DBG$NPARSE_EXAMINE,        ! EXAMINE command parse network
;    84          0216  1     DBG$NPARSE_EXIT,           ! EXIT command parse network
;    85          0217  1     DBG$NPARSE_EXITLOOP,       ! EXITLOOP command parse network
;    86          0218  1     DBG$NPARSE_FOR,            ! FOR command parse network
;    87          0219  1     DBG$NPARSE_GO,             ! GO command parse network
;    88          0220  1     DBG$NPARSE_HELP,           ! HELP command parse network
;    89          0221  1     DBG$NPARSE_IF,             ! IF command parse network
;    90          0222  1     DBG$NPARSE_REPEAT,         ! REPEAT command parse network
;    91          0223  1     DBG$NPARSE_SEARCH,         ! SEARCH command parse network
;    92          0224  1     DBG$NPARSE_SET,            ! SET command parse network
;    93          0225  1     DBG$NPARSE_SHOW,           ! SHOW command parse network
;    94          0226  1     DBG$NPARSE_SPAWN,          ! SPAWN command parse network
;    95          0227  1     DBG$NPARSE_STEP,           ! STEP command parse network
;    96          0228  1     DBG$NPARSE_SYMBOLIZE,      ! SYMBOLIZE command parse network
;    97          0229  1     DBG$NPARSE_TYPE,           ! TYPE command parse network
;    98          0230  1     DBG$NPARSE_UNDEFINE,       ! UNDEFINE command parse network
;    99          0231  1     DBG$NPARSE_WHILE,          ! WHILE command parse network
;   100          0232  1     DBG$NMAKE_ARG_VECT,        ! Constructs a message vector
;   101          0233  1     DBG$NOUT_INFO,             ! Outputs an info message
;   102          0234  1     DBG$NREAD_NAME,            ! Reads a name that is
;   103          0235  1                                !     a potential DEFINEd name
;   104          0236  1     DBG$NSET_LAST_TYPLEN,      ! Sets type and length of current location
;   105          0237  1     DBG$NSYNTAX_ERROR,         ! Constructs a message vector for a syntax error
;   106          0238  1     DBG$REL_MEMORY,            ! Release memory from pool
;   107          0239  1     DBG$SCR_PARSE_DISPLAY_CMD: NOVALUE,  ! Parse the DISPLAY command
;   108          0240  1     DBG$SCR_PARSE_SAVE_CMD: NOVALUE,     ! Parse the SAVE command
;   109          0241  1     DBG$SCR_PARSE_SCROLL_CMD: NOVALUE,   ! Parse the SCROLL command
;   110          0242  1     DBG$SCR_PARSE_SELECT_CMD: NOVALUE,   ! parse the SELECT command
;   111          0243  1     DBG$STA_SETCONTEXT : NOVALUE;        ! Sets registers context
;   112          0244  1
;   113          0245  1 EXTERNAL
;   114          0246  1     DBG$GL_DEVELOPER: BITVECTOR[],       ! Developer flags
;   115          0247  1     DBG$GL_EDIT_ENABLED,                 ! Flag saying whether the EDIT
;   116          0248  1                                          !    command is enabled.
;   117          0249  1     DBG$GB_RADIX: VECTOR[3, BYTE],       ! Radix settings
;   118          0250  1     DBG$GB_EXC_BRE_FLAG: BYTE,           ! Flag set if we are in exception break
;   119          0251  1     DBG$GB_LANGUAGE: BYTE,               ! Current language code
;   120          0252  1     DBG$GL_SCREEN_NOGO,                  ! Flag to disable STEP, GO, and CALL in
```

```
;  121          0253  1                                    !       screen display command lists
;  122          0254  1      DBG$GB_VERB: BYTE;              ! Holds command verb value
;  123          0255  1
;  124          0256  1  LITERAL
;  125          0257  1      ALLOCATE_VERB = DBG$K_ALLOCATE_VERB,! Code for ALLOCATE command
;  126          0258  1      AT_SIGN_VERB = DBG$K_AT_SIGN_VERB,  ! Code for indirect command file execution
;  127          0259  1      ATTACH_VERB = DBG$K_ATTACH_VERB,    ! Code for ATTACH command
;  128          0260  1      CALL_VERB = DBG$K_CALL_VERB,        ! Code for CALL command
;  129          0261  1      CANCEL_VERB = DBG$K_CANCEL_VERB,    ! Code for CANCEL command
;  130          0262  1      DECLARE_VERB = DBG$K_DECLARE_VERB,  ! Code for DECLARE command
;  131          0263  1      DEFINE_VERB = DBG$K_DEFINE_VERB,    ! Code for DEFINE command
;  132          0264  1      DELETE_VERB = DBG$K_DELETE_VERB,    ! Code for DELETE command
;  133          0265  1      DEPOSIT_VERB = DBG$K_DEPOSIT_VERB,  ! Code for DEPOSIT command
;  134          0266  1      DISPLAY_VERB = DBG$K_DISPLAY_VERB,  ! Code for DISPLAY command
;  135          0267  1      DUMP_VERB = DBG$K_DUMP_VERB,        ! Code for DUMP command
;  136          0268  1      EDIT_VERB = DBG$K_EDIT_VERB,        ! Code for EDIT command
;  137          0269  1      EVALUATE_VERB = DBG$K_EVALUATE_VERB,! Code for EVALUATE command
;  138          0270  1      EXAMINE_VERB = DBG$K_EXAMINE_VERB,  ! Code for EXAMINE command
;  139          0271  1      EXIT_VERB = DBG$K_EXIT_VERB,        ! Code for EXIT command
;  140          0272  1      EXITLOOP_VERB = DBG$K_EXITLOOP_VERB,! Code for EXITLOOP command
;  141          0273  1      FOR_VERB = DBG$K_FOR_VERB,          ! Code for FOR command
;  142          0274  1      GO_VERB = DBG$K_GO_VERB,            ! Code for GO command
;  143          0275  1      HELP_VERB = DBG$K_HELP_VERB,        ! Code for HELP command
;  144          0276  1      IF_VERB = DBG$K_IF_VERB,            ! Code for IF command
;  145          0277  1      REPEAT_VERB = DBG$K_REPEAT_VERB,    ! Code for REPEAT command
;  146          0278  1      SAVE_VERB = DBG$K_SAVE_VERB,        ! Code for SAVE command
;  147          0279  1      SCROLL_VERB = DBG$K_SCROLL_VERB,    ! Code for SCROLL command
;  148          0280  1      SEARCH_VERB = DBG$K_SEARCH_VERB,    ! Code for SEARCH command
;  149          0281  1      SELECT_VERB = DBG$K_SELECT_VERB,    ! Code for SELECT command
;  150          0282  1      SET_VERB = DBG$K_SET_VERB,          ! Code for SET command
;  151          0283  1      SHOW_VERB = DBG$K_SHOW_VERB,        ! Code for SHOW command
;  152          0284  1      SPAWN_VERB = DBG$K_SPAWN_VERB,      ! Code for SPAWN command
;  153          0285  1      STEP_VERB = DBG$K_STEP_VERB,        ! Code for STEP command
;  154          0286  1      SYMBOLIZE_VERB =
;  155          0287  1                     DBG$K_SYMBOLIZE_VERB,! Code for SYMBOLIZE command
;  156          0288  1      TYPE_VERB = DBG$K_TYPE_VERB,        ! Code for TYPE command
;  157          0289  1      UNDEFINE_VERB = DBG$K_UNDEFINE_VERB,! Code for UNDEFINE command
;  158          0290  1      WHILE_VERB = DBG$K_WHILE_VERB,      ! Code for WHILE command
;  159          0291  1      WORD_SIZE = 80;                     ! Maximum word size
;  160          0292  1
;  161          0293  1  OWN
;  162          0294  1      WORD_BUF : VECTOR [WORD_SIZE + 1, BYTE]; ! Buffer for counted string word
```

```
;  164     0295  1 GLOBAL ROUTINE DBG$NPARSE_CMD (INPUT_DESC, VERB_NODE_PTR, MESSAGE_VECT) =
;  165     0296  1 !
;  166     0297  1 ! FUNCTIONAL DESCRIPTION:
;  167     0298  1 !
;  168     0299  1 !       Highest level command parsing ATN network. This routine recognizes the
;  169     0300  1 !       verb portion of the input command and transfers control to the appropriate
;  170     0301  1 !       ATN subnetwork to parse the rest of the command.
;  171     0302  1 !
;  172     0303  1 ! FORMAL PARAMETERS:
;  173     0304  1 !
;  174     0305  1 !       INPUT_DESC      - Standard VAX string descriptor of the input command.
;  175     0306  1 !
;  176     0307  1 !       VERB_NODE_PTR   - Pointer to the verb (head) node of the command
;  177     0308  1 !                         execution tree.
;  178     0309  1 !
;  179     0310  1 !       MESSAGE_VECT    - Address of a longword to contain the address of
;  180     0311  1 !                         a message argument vector.
;  181     0312  1 !
;  182     0313  1 ! IMPLICIT INPUTS:
;  183     0314  1 !
;  184     0315  1 !       NONE
;  185     0316  1 !
;  186     0317  1 ! IMPLICIT OUTPUTS:
;  187     0318  1 !
;  188     0319  1 !       The command execution tree is constructed and verb_node_ptr is set to point
;  189     0320  1 !       to the dynamically allocated verb node which is the head node of the tree.
;  190     0321  1 !
;  191     0322  1 ! ROUTINE VALUE:
;  192     0323  1 !
;  193     0324  1 !       unsigned longword integer completion code
;  194     0325  1 !
;  195     0326  1 ! COMPLETION CODES:
;  196     0327  1 !
;  197     0328  1 !       STS$K_SEVERE (4) -      unsuccessful parse
;  198     0329  1 !
;  199     0330  1 !       STS$K_SUCCESS (1) -     successful parse of the input command
;  200     0331  1 !
;  201     0332  1
;  202     0333  2     BEGIN
;  203     0334  2
;  204     0335  2     MAP
;  205     0336  2         INPUT_DESC : REF DBG$STG_DESC;  ! Input string descriptor
;  206     0337  2
;  207     0338  2
;  208     0339  2     ! Define strings used at this level of parsing
;  209     0340  2     !
;  210     0341  2     BIND
;  211     0342  2         DBG$CS_ALLOCATE  = UPLIT BYTE(%ASCIC 'ALLOCATE'),
;  212     0343  2         DBG$CS_AT_SIGN   = UPLIT BYTE(1, DBG$K_AT_SIGN),
;  213     0344  2         DBG$CS_ATTACH    = UPLIT BYTE(%ASCIC 'ATTACH'),
;  214     0345  2         DBG$CS_CALL      = UPLIT BYTE(%ASCIC 'CALL'),
;  215     0346  2         DBG$CS_CANCEL    = UPLIT BYTE(%ASCIC 'CANCEL'),
;  216     0347  2         DBG$CS_DECLARE   = UPLIT BYTE(%ASCIC 'DECLARE'),
;  217     0348  2         DBG$CS_DEFINE    = UPLIT BYTE(%ASCIC 'DEFINE'),
;  218     0349  2         DBG$CS_DELETE    = UPLIT BYTE(%ASCIC 'DELETE'),
;  219     0350  2         DBG$CS_DEPOSIT   = UPLIT BYTE(%ASCIC 'DEPOSIT'),
;  220     0351  2         DBG$CS_DISPLAY   = UPLIT BYTE(%ASCIC 'DISPLAY'),
```

```
;  221          0352  2          DBG$CS_DUMP      = UPLIT BYTE(%ASCIC 'DUMP'),
;  222          0353  2          DBG$CS_EDIT      = UPLIT BYTE(%ASCIC 'EDIT'),
;  223          0354  2          DBG$CS_EVALUATE  = UPLIT BYTE(%ASCIC 'EVALUATE'),
;  224          0355  2          DBG$CS_EXAMINE   = UPLIT BYTE(%ASCIC 'EXAMINE'),
;  225          0356  2          DBG$CS_EXIT      = UPLIT BYTE(%ASCIC 'EXIT'),
;  226          0357  2          DBG$CS_EXITLOOP  = UPLIT BYTE(%ASCIC 'EXITLOOP'),
;  227          0358  2          DBG$CS_FOR       = UPLIT BYTE(%ASCIC 'FOR'),
;  228          0359  2          DBG$CS_GO        = UPLIT BYTE(%ASCIC 'GO'),
;  229          0360  2          DBG$CS_HELP      = UPLIT BYTE(%ASCIC 'HELP'),
;  230          0361  2          DBG$CS_IF        = UPLIT BYTE(%ASCIC 'IF'),
;  231          0362  2          DBG$CS_REPEAT    = UPLIT BYTE(%ASCIC 'REPEAT'),
;  232          0363  2          DBG$CS_SAVE      = UPLIT BYTE(%ASCIC 'SAVE'),
;  233          0364  2          DBG$CS_SCROLL    = UPLIT BYTE(%ASCIC 'SCROLL'),
;  234          0365  2          DBG$CS_SEARCH    = UPLIT BYTE(%ASCIC 'SEARCH'),
;  235          0366  2          DBG$CS_SELECT    = UPLIT BYTE(%ASCIC 'SELECT'),
;  236          0367  2          DBG$CS_SET       = UPLIT BYTE(%ASCIC 'SET'),
;  237          0368  2          DBG$CS_SHOW      = UPLIT BYTE(%ASCIC 'SHOW'),
;  238          0369  2          DBG$CS_SPAWN     = UPLIT BYTE(%ASCIC 'SPAWN'),
;  239          0370  2          DBG$CS_STEP      = UPLIT BYTE(%ASCIC 'STEP'),
;  240          0371  2          DBG$CS_SYMBOLIZE = UPLIT BYTE(%ASCIC 'SYMBOLIZE'),
;  241          0372  2          DBG$CS_TYPE      = UPLIT BYTE(%ASCIC 'TYPE'),
;  242          0373  2          DBG$CS_UNDEFINE  = UPLIT BYTE(%ASCIC 'UNDEFINE'),
;  243          0374  2          DBG$CS_WHILE     = UPLIT BYTE(%ASCIC 'WHILE'),
;  244          0375  2          DBG$CS_CR        = UPLIT BYTE(1, DBG$K_CAR_RETURN);
;  245          0376  2
;  246          0377  2      LOCAL
;  247          0378  2          VERB_NODE : REF DBG$VERB_NODE;  ! Verb node (head of tree)
;  248          0379  2
;  249          0380  2
;  250          0381  2      ! Construct and link the verb node
;  251          0382  2      !
;  252          0383  2      VERB_NODE = DBG$GET_TEMPMEM(DBG$K_VERB_NODE_SIZE);
;  253          0384  2      .VERB_NODE_PTR = .VERB_NODE;
;  254          0385  2      DBG$GB_VERB = 0;
;  255          0386  2
;  256          0387  2
;  257          0388  2      ! Set registers to current context as a default
;  258          0389  2      !
;  259          0390  2      DBG$STA_SETCONTEXT (0);
;  260          0391  2
;  261          0392  2
;  262          0393  2      ! Try to recognize a legal verb. If one is found, transfer control
;  263          0394  2      ! to a command parse subnetwork.
;  264          0395  2      !
;  265          0396  3      IF (SELECTONE TRUE OF
;  266          0397  3              SET
;  267          0398  3
;  268          0399  3              [DBG$NMATCH (.INPUT_DESC, DBG$CS_ALLOCATE, 2)]:
;  269          0400  4                  BEGIN
;  270          0401  4                  VERB_NODE [DBG$B_VERB_LITERAL] = ALLOCATE_VERB;
;  271          0402  4                  DBG$GB_VERB = DBG$K_ALLOCATE_VERB;
;  272          0403  4                  DBG$NPARSE_ALLOCATE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  273          0404  3                  END;
;  274          0405  3
;  275          0406  3              [DBG$NMATCH (.INPUT_DESC, DBG$CS_AT_SIGN, 1)]:
;  276          0407  4                  BEGIN
;  277          0408  4                  VERB_NODE [DBG$B_VERB_LITERAL] = AT_SIGN_VERB;
```

```
;  278          0409  4              DBG$GB_VERB = DBG$K_AT_SIGN_VERB;
;  279          0410  4              DBG$NPARSE_AT_SIGN (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  280          0411  3              END;
;  281          0412
;  282          0413  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_ATTACH, 3)]:
;  283          0414  4              BEGIN
;  284          0415  4              VERB_NODE [DBG$B_VERB_LITERAL] = ATTACH_VERB;
;  285          0416  4              DBG$GB_VERB = DBG$K_ATTACH_VERB;
;  286          0417  4              DBG$NPARSE_ATTACH (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  287          0418  3              END;
;  288          0419
;  289          0420
;  290          0421  3          ! Check for the CALL command.  If this is the CALL command, we
;  291          0422  3          ! signal an error if we are currently in an exception break
;  292          0423  3          ! (we cannot do a CALL if we will restart the program by re-
;  293          0424  3          ! signalling instead of continuing) or if we are currently
;  294          0425  3          ! executing the DEBUG command list of a screen display (which
;  295          0426  3          ! would cause an infinite loop of screen display updating).
;  296          0427  3          ! Otherwise we parse the command normally.
;  297          0428  3          !
;  298          0429  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_CALL, 3)]:
;  299          0430  4              BEGIN
;  300          0431  4              VERB_NODE [DBG$B_VERB_LITERAL] = CALL_VERB;
;  301          0432  4              DBG$GB_VERB = DBG$K_CALL_VERB;
;  302          0433  4              IF .DBG$GB_EXC_BRE_FLAG THEN SIGNAL(DBG$_STEPFROEXC);
;  303          0434  4              IF .DBG$GL_SCREEN_NOGO THEN SIGNAL(DBG$_NOSTEPGO);
;  304          0435  4              DBG$NPARSE_CALL (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  305          0436  3              END.
;  306          0437  3
;  307          0438  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_CANCEL, 3)]:
;  308          0439  4              BEGIN
;  309          0440  4              VERB_NODE [DBG$B_VERB_LITERAL] = CANCEL_VERB;
;  310          0441  4              DBG$GB_VERB = DBG$K_CANCEL_VERB;
;  311          0442  4              DBG$NPARSE_CANCEL (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  312          0443  3              END;
;  313          0444  3
;  314          0445  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DECLARE, 3)]:
;  315          0446  4              BEGIN
;  316          0447  4              VERB_NODE [DBG$B_VERB_LITERAL] = DECLARE_VERB;
;  317          0448  4              DBG$GB_VERB = DBG$K_DECLARE_VERB;
;  318          0449  4              DBG$NPARSE_DECLARE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  319          0450  3              END;
;  320          0451  3
;  321          0452  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DEFINE, 3)]:
;  322          0453  4              BEGIN
;  323          0454  4              VERB_NODE [DBG$B_VERB_LITERAL] = DEFINE_VERB;
;  324          0455  4              DBG$GB_VERB = DBG$K_DEFINE_VERB;
;  325          0456  4              DBG$NPARSE_DEFINE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  326          0457  3              END;
;  327          0458  3
;  328          0459  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DELETE, 3)]:
;  329          0460  4              BEGIN
;  330          0461  4              VERB_NODE [DBG$B_VERB_LITERAL] = DELETE_VERB;
;  331          0462  4              DBG$GB_VERB = DBG$K_DELETE_VERB;
;  332          0463  4              DBG$NPARSE_DELETE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  333          0464  3              END;
;  334          0465  3
```

```
;  335          0466  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DEPOSIT, 1)]:
;  336          0467  4              BEGIN
;  337          0468  4              VERB_NODE [DBG$B_VERB_LITERAL] = DEPOSIT_VERB;
;  338          0469  4              DBG$GB_VERB = DBG$K_DEPOSIT_VERB;
;  339          0470  4              DBG$NPARSE_DEPOSIT (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  340          0471  3              END;
;  341          0472  3
;  342          0473  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DISPLAY, 3)]:
;  343          0474  4              BEGIN
;  344          0475  4              VERB_NODE [DBG$B_VERB_LITERAL] = DISPLAY_VERB;
;  345          0476  4              DBG$GB_VERB = DBG$K_DISPLAY_VERB;
;  346          0477  4              DBG$SCR_PARSE_DISPLAY_CMD(.INPUT_DESC, FALSE, .VERB_NODE);
;  347          0478  4              TRUE
;  348          0479  3              END;
;  349          0480  3
;  350          0481  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DUMP, 2)]:
;  351          0482  4              BEGIN
;  352          0483  4              VERB_NODE [DBG$B_VERB_LITERAL] = DUMP_VERB;
;  353          0484  4              DBG$GB_VERB = DBG$K_DUMP_VERB;
;  354          0485  4              DBG$NPARSE_DUMP (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  355          0486  3              END;
;  356          0487  3
;  357          0488  3
;  358          0489  3          ! The EDIT command is our interface to the EDITH text editor.
;  359          0490  3          ! We allow this command if a certain global flag is set.
;  360          0491  3          ! That way, the command can be disabled for version 4.0,
;  361          0492  3          ! and enabled with a patch in a future release. The
;  362          0493  3          ! all-purpose developer bit 0 also enables this command,
;  363          0494  3          ! so that developers can easily test it.
;  364          0495  3          !
;  365          0496  4          [  (IF .DBG$GL_EDIT_ENABLED OR .DBG$GL_DEVELOPER[0]
;  366          0497  4              THEN
;  367          0498  4                  DBG$NMATCH (.INPUT_DESC, DBG$CS_EDIT, 2)
;  368          0499  4              ELSE
;  369          0500  3                  FALSE)]:
;  370          0501  4              BEGIN
;  371          0502  4              VERB_NODE [DBG$B_VERB_LITERAL] = EDIT_VERB;
;  372          0503  4              DBG$GB_VERB = DBG$K_EDIT_VERB;
;  373          0504  4              DBG$NPARSE_EDIT (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  374          0505  3              END;
;  375          0506  3
;  376          0507  3
;  377          0508  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_EVALUATE, 2)]:
;  378          0509  4              BEGIN
;  379          0510  4              VERB_NODE [DBG$B_VERB_LITERAL] = EVALUATE_VERB;
;  380          0511  4              DBG$GB_VERB = DBG$K_EVALUATE_VERB;
;  381          0512  4              DBG$NPARSE_EVALUATE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  382          0513  3              END;
;  383          0514  3
;  384          0515  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_EXAMINE, 1)]:
;  385          0516  4              BEGIN
;  386          0517  4              VERB_NODE [DBG$B_VERB_LITERAL] = EXAMINE_VERB;
;  387          0518  4              DBG$GB_VERB = DBG$K_EXAMINE_VERB;
;  388          0519  4              DBG$NPARSE_EXAMINE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  389          0520  3              END;
;  390          0521  3
;  391          0522  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_EXIT, 3)]:
```

```
;  392          0523  4              BEGIN
;  393          0524  4              VERB_NODE [DBG$B_VERB_LITERAL] = EXIT_VERB;
;  394          0525  4              DBG$GB_VERB = DBG$K_EXIT_VERB;
;  395          0526  4              DBG$NPARSE_EXIT (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  396          0527  3              END;
;  397          0528  3
;  398          0529  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_EXITLOOP, 5)]:
;  399          0530  4              BEGIN
;  400          0531  4              VERB_NODE [DBG$B_VERB_LITERAL] = EXITLOOP_VERB;
;  401          0532  4              DBG$GB_VERB = DBG$K_EXITLOOP_VERB;
;  402          0533  4              DBG$NPARSE_EXITLOOP (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  403          0534  3              END;
;  404          0535  3
;  405          0536  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_FOR, 1)]:
;  406          0537  4              BEGIN
;  407          0538  4              VERB_NODE [DBG$B_VERB_LITERAL] = FOR_VERB;
;  408          0539  4              DBG$GB_VERB = DBG$K_FOR_VERB;
;  409          0540  4              DBG$NPARSE_FOR (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  410          0541  3              END;
;  411          0542  3
;  412          0543  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_GO, 1)]:
;  413          0544  4              BEGIN
;  414          0545  4              VERB_NODE [DBG$B_VERB_LITERAL] = GO_VERB;
;  415          0546  4              DBG$GB_VERB = DBG$K_GO_VERB;
;  416          0547  4              IF .DBG$GL_SCREEN_NOGO THEN SIGNAL(DBG$_NOSTEPGO);
;  417          0548  4              DBG$NPARSE_GO (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  418          0549  3              END;
;  419          0550  3
;  420          0551  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_HELP, 1)]:
;  421          0552  4              BEGIN
;  422          0553  4              VERB_NODE [DBG$B_VERB_LITERAL] = HELP_VERB;
;  423          0554  4              DBG$GB_VERB = DBG$K_HELP_VERB;
;  424          0555  4              DBG$NPARSE_HELP (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  425          0556  3              END;
;  426          0557  3
;  427          0558  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_IF, 1)]:
;  428          0559  4              BEGIN
;  429          0560  4              VERB_NODE [DBG$B_VERB_LITERAL] = IF_VERB;
;  430          0561  4              DBG$GB_VERB = DBG$K_IF_VERB;
;  431          0562  4              DBG$NPARSE_IF (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  432          0563  3              END;
;  433          0564  3
;  434          0565  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_REPEAT, 2)]:
;  435          0566  4              BEGIN
;  436          0567  4              VERB_NODE [DBG$B_VERB_LITERAL] = REPEAT_VERB;
;  437          0568  4              DBG$GB_VERB = DBG$K_REPEAT_VERB;
;  438          0569  4              DBG$NPARSE_REPEAT (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  439          0570  3              END;
;  440          0571  3
;  441          0572  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_SAVE, 3)]:
;  442          0573  4              BEGIN
;  443          0574  4              VERB_NODE [DBG$B_VERB_LITERAL] = SAVE_VERB;
;  444          0575  4              DBG$GB_VERB = DBG$K_SAVE_VERB;
;  445          0576  4              DBG$SCR_PARSE_SAVE_CMD(.INPUT_DESC, .VERB_NODE);
;  446          0577  4              TRUE
;  447          0578  3              END;
;  448          0579  3
```

```
;  449          0580  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_SCROLL, 3)]:
;  450          0581  4              BEGIN
;  451          0582  4              VERB_NODE [DBG$B_VERB_LITERAL] = SCROLL_VERB;
;  452          0583  4              DBG$GB_VERB = DBG$K_SCROLL_VERB;
;  453          0584  4              DBG$SCR_PARSE_SCROLL_CMD(.INPUT_DESC, .VERB_NODE);
;  454          0585  4              TRUE
;  455          0586  3              END;
;  456          0587  3
;  457          0588  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_SEARCH, 3)]:
;  458          0589  4              BEGIN
;  459          0590  4              VERB_NODE [DBG$B_VERB_LITERAL] = SEARCH_VERB;
;  460          0591  4              DBG$GB_VERB = DBG$K_SEARCH_VERB;
;  461          0592  4              DBG$NPARSE_SEARCH (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  462          0593  3              END;
;  463          0594  3
;  464          0595  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_SELECT, 3)]:
;  465          0596  4              BEGIN
;  466          0597  4              VERB_NODE [DBG$B_VERB_LITERAL] = SELECT_VERB;
;  467          0598  4              DBG$GB_VERB = DBG$K_SELECT_VERB;
;  468          0599  4              DBG$SCR_PARSE_SELECT_CMD(.INPUT_DESC, .VERB_NODE);
;  469          0600  4              TRUE
;  470          0601  3              END;
;  471          0602  3
;  472          0603  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_SET, 2)]:
;  473          0604  4              BEGIN
;  474          0605  4              VERB_NODE [DBG$B_VERB_LITERAL] = SET_VERB;
;  475          0606  4              DBG$GB_VERB = DBG$K_SET_VERB;
;  476          0607  4              DBG$NPARSE_SET (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  477          0608  3              END;
;  478          0609  3
;  479          0610  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_SHOW, 2)]:
;  480          0611  4              BEGIN
;  481          0612  4              VERB_NODE [DBG$B_VERB_LITERAL] = SHOW_VERB;
;  482          0613  4              DBG$GB_VERB = DBG$K_SHOW_VERB;
;  483          0614  4              DBG$NPARSE_SHOW (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  484          0615  3              END;
;  485          0616  3
;  486          0617  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_SPAWN, 2)]:
;  487          0618  4              BEGIN
;  488          0619  4              VERB_NODE [DBG$B_VERB_LITERAL] = SPAWN_VERB;
;  489          0620  4              DBG$GB_VERB = DBG$K_SPAWN_VERB;
;  490          0621  4              DBG$NPARSE_SPAWN (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  491          0622  3              END;
;  492          0623  3
;  493          0624  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_STEP, 1)]:
;  494          0625  4              BEGIN
;  495          0626  4              VERB_NODE [DBG$B_VERB_LITERAL] = STEP_VERB;
;  496          0627  4              DBG$GB_VERB = DBG$K_STEP_VERB;
;  497          0628  4              IF .DBG$GL_SCREEN_NOGO THEN SIGNAL(DBG$_NOSTEPGO);
;  498          0629  4              DBG$NPARSE_STEP (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  499          0630  3              END;
;  500          0631  3
;  501          0632  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_SYMBOLIZE, 2)]:
;  502          0633  4              BEGIN
;  503          0634  4              VERB_NODE [DBG$B_VERB_LITERAL] = SYMBOLIZE_VERB;
;  504          0635  4              DBG$GB_VERB = DBG$K_SYMBOLIZE_VERB;
;  505          0636  4              DBG$NPARSE_SYMBOLIZE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
```

```
;  506          0637  3              END;
;  507          0638  3
;  508          0639  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_TYPE, 1)]:
;  509          0640  4              BEGIN
;  510          0641  4              VERB_NODE [DBG$B_VERB_LITERAL] = TYPE_VERB;
;  511          0642  4              DBG$GB_VERB = DBG$K_TYPE_VERB;
;  512          0643  4              DBG$NPARSE_TYPE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  513          0644  3              END;
;  514          0645  3
;  515          0646  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_UNDEFINE, 1)]:
;  516          0647  4              BEGIN
;  517          0648  4              VERB_NODE [DBG$B_VERB_LITERAL] = UNDEFINE_VERB;
;  518          0649  4              DBG$GB_VERB = DBG$K_UNDEFINE_VERB;
;  519          0650  4              DBG$NPARSE_UNDEFINE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  520          0651  3              END;
;  521          0652  3
;  522          0653  3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_WHILE, 1)]:
;  523          0654  4              BEGIN
;  524          0655  4              VERB_NODE [DBG$B_VERB_LITERAL] = WHILE_VERB;
;  525          0656  4              DBG$GB_VERB = DBG$K_WHILE_VERB;
;  526          0657  4              DBG$NPARSE_WHILE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;  527          0658  3              END;
;  528          0659  3
;  529          0660  3
;  530          0661  3          ! Any other command name constitutes a syntax error.
;  531          0662  3          !
;  532          0663  3          [OTHERWISE]:
;  533          0664  4              BEGIN
;  534          0665  4              .MESSAGE_VECT = DBG$NSYNTAX_ERROR (DBG$NNEXT_WORD (.INPUT_DESC));
;  535          0666  4              FALSE
;  536          0667  3              END;
;  537          0668  3
;  538          0669  3          TES)
;  539          0670  2      THEN
;  540          0671  2          RETURN
;  541          0672  2
;  542          0673  2
;  543          0674  2          ! Check for exhausted input
;  544          0675  2          !
;  545          0676  3          (IF .INPUT_DESC [DSC$W_LENGTH] EQL 0 OR
;  546          0677  3              DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
;  547          0678  3          THEN
;  548          0679  3              STS$K_SUCCESS
;  549          0680  3
;  550          0681  3          ELSE
;  551          0682  4              BEGIN
;  552          0683  4              .MESSAGE_VECT = DBG$NSYNTAX_ERROR (DBG$NNEXT_WORD (.INPUT_DESC));
;  553          0684  4              STS$K_SEVERE
;  554          0685  4              END
;  555          0686  3          )
;  556          0687  2      ELSE
;  557          0688  2          RETURN STS$K_SEVERE;
;  558          0689  2
;  559          0690  1      END;

                                                        .TITLE  DBGNPARSE
```

```
                                                        .IDENT  \V04-000\

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

45 54 41 43 4F 4C 4C 41 08 00000 P.AAA:  .ASCII  <8>\ALLOCATE\
                     40 01 00009 P.AAB:  .BYTE   1, 64
         48 43 41 54 54 41 06 00008 P.AAC:  .ASCII  <6>\ATTACH\
                  4C 4C 41 43 04 00012 P.AAD:  .ASCII  <4>\CALL\
         4C 45 43 4E 41 43 06 00017 P.AAE:  .ASCII  <6>\CANCEL\
      45 52 41 4C 43 45 44 07 0001E P.AAF:  .ASCII  <7>\DECLARE\
         45 4E 49 46 45 44 06 00026 P.AAG:  .ASCII  <6>\DEFINE\
         45 54 45 4C 45 44 06 0002D P.AAH:  .ASCII  <6>\DELETE\
      54 49 53 4F 50 45 44 07 00034 P.AAI:  .ASCII  <7>\DEPOSIT\
      59 41 4C 50 53 49 44 07 0003C P.AAJ:  .ASCII  <7>\DISPLAY\
               50 4D 55 44 04 00044 P.AAK:  .ASCII  <4>\DUMP\
               54 49 44 45 04 00049 P.AAL:  .ASCII  <4>\EDIT\
45 54 41 55 4C 41 56 45 08 0004E P.AAM:  .ASCII  <8>\EVALUATE\
   45 4E 49 4D 41 58 45 07 00057 P.AAN:  .ASCII  <7>\EXAMINE\
               54 49 58 45 04 0005F P.AAO:  .ASCII  <4>\EXIT\
50 4F 4F 4C 54 49 58 45 08 00064 P.AAP:  .ASCII  <8>\EXITLOOP\
                  52 4F 46 03 0006D P.AAQ:  .ASCII  <3>\FOR\
                     4F 47 02 00071 P.AAR:  .ASCII  <2>\GO\
               50 4C 45 48 04 00074 P.AAS:  .ASCII  <4>\HELP\
                     46 49 02 00079 P.AAT:  .ASCII  <2>\IF\
         54 41 45 50 45 52 06 0007C P.AAU:  .ASCII  <6>\REPEAT\
               45 56 41 53 04 00083 P.AAV:  .ASCII  <4>\SAVE\
         4C 4C 4F 52 43 53 06 00088 P.AAW:  .ASCII  <6>\SCROLL\
         48 43 52 41 45 53 06 0008F P.AAX:  .ASCII  <6>\SEARCH\
         54 43 45 4C 45 53 06 00096 P.AAY:  .ASCII  <6>\SELECT\
                  54 45 53 03 0009D P.AAZ:  .ASCII  <3>\SET\
               57 4F 48 53 04 000A1 P.ABA:  .ASCII  <4>\SHOW\
            4E 57 41 50 53 05 000A6 P.ABB:  .ASCII  <5>\SPAWN\
               50 45 54 53 04 000AC P.ABC:  .ASCII  <4>\STEP\
45 5A 49 4C 4F 42 4D 59 53 09 000B1 P.ABD:  .ASCII  <9>\SYMBOLIZE\
               45 50 59 54 04 000BB P.ABE:  .ASCII  <4>\TYPE\
   45 4E 49 46 45 44 4E 55 08 000C0 P.ABF:  .ASCII  <8>\UNDEFINE\
            45 4C 49 48 57 05 000C9 P.ABG:  .ASCII  <5>\WHILE\
                     0D 01 000CF P.ABH:  .BYTE   1, 13

                                                        .PSECT  DBG$OWN,NOEXE,  PIC,2

                                        00000 WORD_BUF:
                                                        .BLKB   81

                                        DBG$CS_ALLOCATE=    P.AAA
                                        DBG$CS_AT_SIGN=     P.AAB
                                        DBG$CS_ATTACH=      P.AAC
                                        DBG$CS_CALL=        P.AAD
                                        DBG$CS_CANCEL=      P.AAE
                                        DBG$CS_DECLARE=     P.AAF
                                        DBG$CS_DEFINE=      P.AAG
                                        DBG$CS_DELETE=      P.AAH
                                        DBG$CS_DEPOSIT=     P.AAI
                                        DBG$CS_DISPLAY=     P.AAJ
                                        DBG$CS_DUMP=        P.AAK
                                        DBG$CS_EDIT=        P.AAL
                                        DBG$CS_EVALUATE=    P.AAM
```

```
DBG$CS_EXAMINE=     P.AAN
DBG$CS_EXIT=        P.AAO
DBG$CS_EXITLOOP=    P.AAP
DBG$CS_FOR=         P.AAQ
DBG$CS_GO=          P.AAR
DBG$CS_HELP=        P.AAS
DBG$CS_IF=          P.AAT
DBG$CS_REPEAT=      P.AAU
DBG$CS_SAVE=        P.AAV
DBG$CS_SCROLL=      P.AAW
DBG$CS_SEARCH=      P.AAX
DBG$CS_SELECT=      P.AAY
DBG$CS_SET=         P.AAZ
DBG$CS_SHOW=        P.ABA
DBG$CS_SPAWN=       P.ABB
DBG$CS_STEP=        P.ABC
DBG$CS_SYMBOLIZE=   P.ABD
DBG$CS_TYPE=        P.ABE
DBG$CS_UNDEFINE=    P.ABF
DBG$CS_WHILE=       P.ABG
DBG$CS_CR=          P.ABH
        .EXTRN  DBG$ADDR_EXP_INT
        .EXTRN  DBG$DEF_SYM_FIND
        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
        .EXTRN  DBG$EXP_INT, DBG$NPARSE_ALLOCATE
        .EXTRN  DBG$NPARSE_AT_SIGN
        .EXTRN  DBG$NPARSE_ATTACH
        .EXTRN  DBG$NPARSE_CALL
        .EXTRN  DBG$NPARSE_CANCEL
        .EXTRN  DBG$NPARSE_DECLARE
        .EXTRN  DBG$NPARSE_DEFINE
        .EXTRN  DBG$NPARSE_DELETE
        .EXTRN  DBG$NPARSE_DEPOSIT
        .EXTRN  DBG$NPARSE_DUMP
        .EXTRN  DBG$NPARSE_EDIT
        .EXTRN  DBG$NPARSE_EVALUATE
        .EXTRN  DBG$NPARSE_EXAMINE
        .EXTRN  DBG$NPARSE_EXIT
        .EXTRN  DBG$NPARSE_EXITLOOP
        .EXTRN  DBG$NPARSE_FOR, DBG$NPARSE_GO
        .EXTRN  DBG$NPARSE_HELP
        .EXTRN  DBG$NPARSE_IF, DBG$NPARSE_REPEAT
        .EXTRN  DBG$NPARSE_SEARCH
        .EXTRN  DBG$NPARSE_SET, DBG$NPARSE_SHOW
        .EXTRN  DBG$NPARSE_SPAWN
        .EXTRN  DBG$NPARSE_STEP
        .EXTRN  DBG$NPARSE_SYMBOLIZE
        .EXTRN  DBG$NPARSE_TYPE
        .EXTRN  DBG$NPARSE_UNDEFINE
        .EXTRN  DBG$NPARSE_WHILE
        .EXTRN  DBG$NMAKE_ARG_VECT
        .EXTRN  DBG$NOUT_INFO, DBG$NREAD_NAME
        .EXTRN  DBG$NSET_LAST_TYPLEN
        .EXTRN  DBG$NSYNTAX_ERROR
        .EXTRN  DBG$REL_MEMORY, DBG$SCR_PARSE_DISPLAY_CMD
        .EXTRN  DBG$SCR_PARSE_SAVE_CMD
        .EXTRN  DBG$SCR_PARSE_SCROLL_CMD
```

```
                                        .EXTRN  DBG$SCR_PARSE_SELECT_CMD
                                        .EXTRN  DBG$STA_SETCONTEXT
                                        .EXTRN  DBG$GL_DEVELOPER
                                        .EXTRN  DBG$GL_EDIT_ENABLED
                                        .EXTRN  DBG$GB_RADIX, DBG$GB_EXC_BRE_FLAG
                                        .EXTRN  DBG$GB_LANGUAGE
                                        .EXTRN  DBG$GL_SCREEN_NOGO
                                        .EXTRN  DBG$GB_VERB

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

                                01FC 00000      .ENTRY  DBG$NPARSE_CMD, Save R2,R3,R4,R5,R6,R7,R8    ; 0295
                    58 00000000G  00  9E 00002  MOVAB   DBG$GL_SCREEN_NOGO, R8                       ;
                    57 00000000G  00  9E 00009  MOVAB   LIB$SIGNAL, R7                              ;
                    56      0000V CF  9E 00010  MOVAB   DBG$NMATCH, R6                              ;
                    55 00000000'  EF  9E 00015  MOVAB   DBG$CS_ALLOCATE, R5                         ;
                    54 00000000G  00  9E 0001C  MOVAB   DBG$GB_VERB, R4                             ;
                                  03  DD 00023  PUSHL   #3                                         ; 0383
 00000000G  00                    01  FB 00025  CALLS   #1, DBG$GET_TEMPMEM                        ;
                    53            50  D0 0002C  MOVL    R0, VERB_NODE                              ;
          08  BC                  53  D0 0002F  MOVL    VERB_NODE, @VERB_NODE_PTR                  ; 0384
                                  64  94 00033  CLRB    DBG$GB_VERB                                ; 0385
                                  7E  D4 00035  CLRL    -(SP)                                      ; 0390
 00000000G  00                    01  FB 00037  CALLS   #1, DBG$STA_SETCONTEXT                     ;
                                  02  DD 0003E  PUSHL   #2                                         ; 0399
                                  55  DD 00040  PUSHL   R5                                         ;
                    52        04  AC  D0 00042  MOVL    INPUT_DESC, R2                             ;
                                  52  DD 00046  PUSHL   R2                                         ;
                    66            03  FB 00048  CALLS   #3, DBG$NMATCH                            ;
                    01            50  D1 0004B  CMPL    R0, #1                                     ;
                                  14  12 0004E  BNEQ    1$                                         ;
                    63            16  90 00050  MOVB    #22, (VERB_NODE)                           ; 0401
                    64            16  90 00053  MOVB    #22, DBG$GB_VERB                           ; 0402
                             0C   AC  DD 00056  PUSHL   MESSAGE_VECT                               ; 0403
                             0C   BB 00059  PUSHR   #^M<R2,R3>                                     ;
 00000000G  00                    03  FB 0005B  CALLS   #3, DBG$NPARSE_ALLOCATE                    ;
                                  44  11 00062  BRB     3$                                         ;
                                  01  DD 00064  1$:  PUSHL   #1                                    ; 0406
                             09   A5  9F 00066  PUSHAB  DBG$CS_AT_SIGN                             ;
                                  52  DD 00069  PUSHL   R2                                         ;
                    66            03  FB 0006B  CALLS   #3, DBG$NMATCH                            ;
                    01            50  D1 0006E  CMPL    R0, #1                                     ;
                                  14  12 00071  BNEQ    2$                                         ;
                    63            01  90 00073  MOVB    #1, (VERB_NODE)                            ; 0408
                    64            01  90 00076  MOVB    #1, DBG$GB_VERB                            ; 0409
                             0C   AC  DD 00079  PUSHL   MESSAGE_VECT                               ; 0410
                             0C   BB 0007C  PUSHR   #^M<R2,R3>                                     ;
 00000000G  00                    03  FB 0007E  CALLS   #3, DBG$NPARSE_AT_SIGN                     ;
                                  60  11 00085  BRB     7$                                         ;
                                  03  DD 00087  2$:  PUSHL   #3                                    ; 0413
                             0B   A5  9F 00089  PUSHAB  DBG$CS_ATTACH                              ;
                                  52  DD 0008C  PUSHL   R2                                         ;
                    66            03  FB 0008E  CALLS   #3, DBG$NMATCH                            ;
                    01            50  D1 00091  CMPL    R0, #1                                     ;
                                  14  12 00094  BNEQ    4$                                         ;
                    63            1A  90 00096  MOVB    #26, (VERB_NODE)                           ; 0415
                    64            1A  90 00099  MOVB    #26, DBG$GB_VERB                           ; 0416
```

```
                        0C   AC  DD 0009C          PUSHL   MESSAGE_VECT                          ; 0417
                             0C  BB 0009F          PUSHR   #^M<R2,R3>                           ;
00000000G  00                03  FB 000A1          CALLS   #3, DBG$NPARSE_ATTACH                ;
                             60  11 000A8 3$:      BRB     9$                                   ;
                             03  DD 000AA 4$:      PUSHL   #3                                   ; 0429
                        12   A5  9F 000AC          PUSHAB  DBG$CS_CALL                          ;
                             52  DD 000AF          PUSHL   R2                                   ;
                     66      03  FB 000B1          CALLS   #3, DBG$NMATCH                       ;
                     01      50  D1 000B4          CMPL    R0, #1                               ;
                             30  12 000B7          BNEQ    8$                                   ;
                     63      02  90 000B9          MOVB    #2, (VERB_NODE)                      ; 0431
                     64      02  90 000BC          MOVB    #2, DBG$GB_VERB                      ; 0432
                     09 00000000G 00  E9 000BF     BLBC    DBG$GB_EXC_BRE_FLAG, 5$              ; 0433
                        00028E38 8F  DD 000C6      PUSHL   #167480                              ;
                     67      01  FB 000CC          CALLS   #1, LIB$SIGNAL                       ;
                     09      68  E9 000CF 5$:      BLBC    DBG$GL_SCREEN_NOGO, 6$               ; 0434
                        0002836A 8F  DD 000D2      PUSHL   #164714                              ;
                     67      01  FB 000D8          CALLS   #1, LIB$SIGNAL                       ;
                        0C   AC  DD 000DB 6$:      PUSHL   MESSAGE_VECT                         ; 0435
                             0C  BB 000DE          PUSHR   #^M<R2,R3>                           ;
00000000G  00                03  FB 000E0          CALLS   #3, DBG$NPARSE_CALL                  ;
                             67  11 000E7 7$:      BRB     12$                                  ;
                             03  DD 000E9 8$:      PUSHL   #3                                   ; 0438
                        17   A5  9F 000EB          PUSHAB  DBG$CS_CANCEL                        ;
                             52  DD 000EE          PUSHL   R2                                   ;
                     66      03  FB 000F0          CALLS   #3, DBG$NMATCH                       ;
                     01      50  D1 000F3          CMPL    R0, #1                               ;
                             14  12 000F6          BNEQ    10$                                  ;
                     63      03  90 000F8          MOVB    #3, (VERB_NODE)                      ; 0440
                     64      03  90 000FB          MOVB    #3, DBG$GB_VERB                      ; 0441
                        0C   AC  DD 000FE          PUSHL   MESSAGE_VECT                         ; 0442
                             0C  BB 00101          PUSHR   #^M<R2,R3>                           ;
00000000G  00                03  FB 00103          CALLS   #3, DBG$NPARSE_CANCEL                ;
                             67  11 0010A 9$:      BRB     14$                                  ;
                             03  DD 0010C 10$:     PUSHL   #3                                   ; 0445
                        1E   A5  9F 0010E          PUSHAB  DBG$CS_DECLARE                       ;
                             52  DD 00111          PUSHL   R2                                   ;
                     66      03  FB 00113          CALLS   #3, DBG$NMATCH                       ;
                     01      50  D1 00116          CMPL    R0, #1                               ;
                             14  12 00119          BNEQ    11$                                  ;
                     63      14  90 0011B          MOVB    #20, (VERB_NODE)                     ; 0447
                     64      14  90 0011E          MOVB    #20, DBG$GB_VERB                     ; 0448
                        0C   AC  DD 00121          PUSHL   MESSAGE_VECT                         ; 0449
                             0C  BB 00124          PUSHR   #^M<R2,R3>                           ;
00000000G  00                03  FB 00126          CALLS   #3, DBG$NPARSE_DECLARE               ;
                             67  11 0012D          BRB     16$                                  ;
                             03  DD 0012F 11$:     PUSHL   #3                                   ; 0452
                        26   A5  9F 00131          PUSHAB  DBG$CS_DEFINE                        ;
                             52  DD 00134          PUSHL   R2                                   ;
                     66      03  FB 00136          CALLS   #3, DBG$NMATCH                       ;
                     01      50  D1 00139          CMPL    R0, #1                               ;
                             14  12 0013C          BNEQ    13$                                  ;
                     63      04  90 0013E          MOVB    #4, (VERB_NODE)                      ; 0454
                     64      04  90 00141          MOVB    #4, DBG$GB_VERB                      ; 0455
                        0C   AC  DD 00144          PUSHL   MESSAGE_VECT                         ; 0456
                             0C  BB 00147          PUSHR   #^M<R2,R3>                           ;
00000000G  00                03  FB 00149          CALLS   #3, DBG$NPARSE_DEFINE                ;
```

```
                                44  11 00150 12$:    BRB     16$                                     ;
                                03  DD 00152 13$:    PUSHL   #3                                      ; 0459
                             2D A5  9F 00154         PUSHAB  DBG$CS_DELETE                           ;
                                52  DD 00157         PUSHL   R2                                      ;
                             66 03  FB 00159         CALLS   #3, DBG$NMATCH                          ;
                             01 50  D1 0015C         CMPL    R0, #1                                  ;
                                14  12 0015F         BNEQ    15$                                     ;
                             63 20  90 00161         MOVB    #32, (VERB_NODE)                        ; 0461
                             64 20  90 00164         MOVB    #32, DBG$GB_VERB                        ; 0462
                             0C AC  DD 00167         PUSHL   MESSAGE_VECT                            ; 0463
                                0C  BB 0016A         PUSHR   #^M<R2,R3>                              ;
                00000000G    00 03  FB 0016C         CALLS   #3, DBG$NPARSE_DELETE                   ;
                                69  11 00173 14$:    BRB     19$                                     ;
                                01  DD 00175 15$:    PUSHL   #1                                      ; 0466
                             34 A5  9F 00177         PUSHAB  DBG$CS_DEPOSIT                          ;
                                52  DD 0017A         PUSHL   R2                                      ;
                             66 03  FB 0017C         CALLS   #3, DBG$NMATCH                          ;
                             01 50  D1 0017F         CMPL    R0, #1                                  ;
                                14  12 00182         BNEQ    17$                                     ;
                             63 05  90 00184         MOVB    #5, (VERB_NODE)                         ; 0468
                             64 05  90 00187         MOVB    #5, DBG$GB_VERB                         ; 0469
                             0C AC  DD 0018A         PUSHL   MESSAGE_VECT                            ; 0470
                                0C  BB 0018D         PUSHR   #^M<R2,R3>                              ;
                00000000G    00 03  FB 0018F         CALLS   #3, DBG$NPARSE_DEPOSIT                  ;
                                7B  11 00196 16$:    BRB     24$                                     ;
                                03  DD 00198 17$:    PUSHL   #3                                      ; 0473
                             3C A5  9F 0019A         PUSHAB  DBG$CS_DISPLAY                          ;
                                52  DD 0019D         PUSHL   R2                                      ;
                             66 03  FB 0019F         CALLS   #3, DBG$NMATCH                          ;
                             01 50  D1 001A2         CMPL    R0, #1                                  ;
                                16  12 001A5         BNEQ    18$                                     ;
                             63 1C  90 001A7         MOVB    #28, (VERB_NODE)                        ; 0475
                             64 1C  90 001AA         MOVB    #28, DBG$GB_VERB                        ; 0476
                                53  DD 001AD         PUSHL   VERB_NODE                               ; 0477
                                7E  D4 001AF         CLRL    -(SP)                                   ;
                                52  DD 001B1         PUSHL   R2                                      ;
                00000000G    00 03  FB 001B3         CALLS   #3, DBG$SCR_PARSE_DISPLAY_CMD           ;
                              0354  31 001BA         BRW     64$                                     ;
                                02  DD 001BD 18$:    PUSHL   #2                                      ; 0481
                             44 A5  9F 001BF         PUSHAB  DBG$CS_DUMP                             ;
                                52  DD 001C2         PUSHL   R2                                      ;
                             66 03  FB 001C4         CALLS   #3, DBG$NMATCH                          ;
                             01 50  D1 001C7         CMPL    R0, #1                                  ;
                                14  12 001CA         BNEQ    20$                                     ;
                             63 1B  90 001CC         MOVB    #27, (VERB_NODE)                        ; 0483
                             64 1B  90 001CF         MOVB    #27, DBG$GB_VERB                        ; 0484
                             0C AC  DD 001D2         PUSHL   MESSAGE_VECT                            ; 0485
                                0C  BB 001D5         PUSHR   #^M<R2,R3>                              ;
                00000000G    00 03  FB 001D7         CALLS   #3, DBG$NPARSE_DUMP                     ;
                                79  11 001DE 19$:    BRB     27$                                     ;
                07 00000000G 00 E8  001E0 20$:    BLBS    DBG$GL_EDIT_ENABLED, 21$                ; 0496
                0C 00000000G 00 E9  001E7         BLBC    DBG$GL_DEVELOPER, 22$                   ;
                                02  DD 001EE 21$:    PUSHL   #2                                      ; 0498
                             49 A5  9F 001F0         PUSHAB  DBG$CS_EDIT                             ;
                                52  DD 001F3         PUSHL   R2                                      ;
                             66 03  FB 001F5         CALLS   #3, DBG$NMATCH                          ;
                                02  11 001F8         BRB     23$                                     ;
```

```
                        50  D4 001FA 22$:   CLRL    R0                              ; 0496
                    01  50  D1 001FC 23$:   CMPL    R0, #1                          ;
                        14  12 001FF        BNEQ    25$                             ;
                    63  21  90 00201        MOVB    #33, (VERB_NODE)                ; 0502
                    64  21  90 00204        MOVB    #33, DBG$GB_VERB                ; 0503
                0C  AC  DD 00207        PUSHL   MESSAGE_VECT                    ; 0504
                    0C  BB 0020A        PUSHR   #^M<R2,R3>                      ;
00000000G       00  03  FB 0020C        CALLS   #3, DBG$NPARSE_EDIT             ;
                    67  11 00213 24$:    BRB     29$                             ;
                    02  DD 00215 25$:    PUSHL   #2                              ; 0508
                4E  A5  9F 00217        PUSHAB  DBG$CS_EVALUATE                 ;
                    52  DD 0021A        PUSHL   R2                              ;
                    66  03  FB 0021C        CALLS   #3, DBG$NMATCH                  ;
                    01  50  D1 0021F        CMPL    R0, #1                          ;
                        14  12 00222        BNEQ    26$                             ;
                    63  06  90 00224        MOVB    #6, (VERB_NODE)                 ; 0510
                    64  06  90 00227        MOVB    #6, DBG$GB_VERB                 ; 0511
                0C  AC  DD 0022A        PUSHL   MESSAGE_VECT                    ; 0512
                    0C  BB 0022D        PUSHR   #^M<R2,R3>                      ;
00000000G       00  03  FB 0022F        CALLS   #3, DBG$NPARSE_EVALUATE         ;
                    67  11 00236        BRB     31$                             ;
                    01  DD 00238 26$:    PUSHL   #1                              ; 0515
                57  A5  9F 0023A        PUSHAB  DBG$CS_EXAMINE                  ;
                    52  DD 0023D        PUSHL   R2                              ;
                    66  03  FB 0023F        CALLS   #3, DBG$NMATCH                  ;
                    01  50  D1 00242        CMPL    R0, #1                          ;
                        14  12 00245        BNEQ    28$                             ;
                    63  07  90 00247        MOVB    #7, (VERB_NODE)                 ; 0517
                    64  07  90 0024A        MOVB    #7, DBG$GB_VERB                 ; 0518
                0C  AC  DD 0024D        PUSHL   MESSAGE_VECT                    ; 0519
                    0C  BB 00250        PUSHR   #^M<R2,R3>                      ;
00000000G       00  03  FB 00252        CALLS   #3, DBG$NPARSE_EXAMINE          ;
                    67  11 00259 27$:    BRB     33$                             ;
                    03  DD 0025B 28$:    PUSHL   #3                              ; 0522
                5F  A5  9F 0025D        PUSHAB  DBG$CS_EXIT                     ;
                    52  DD 00260        PUSHL   R2                              ;
                    66  03  FB 00262        CALLS   #3, DBG$NMATCH                  ;
                    01  50  D1 00265        CMPL    R0, #1                          ;
                        14  12 00268        BNEQ    30$                             ;
                    63  08  90 0026A        MOVB    #8, (VERB_NODE)                 ; 0524
                    64  08  90 0026D        MOVB    #8, DBG$GB_VERB                 ; 0525
                0C  AC  DD 00270        PUSHL   MESSAGE_VECT                    ; 0526
                    0C  BB 00273        PUSHR   #^M<R2,R3>                      ;
00000000G       00  03  FB 00275        CALLS   #3, DBG$NPARSE_EXIT             ;
                    73  11 0027C 29$:    BRB     36$                             ;
                    05  DD 0027E 30$:    PUSHL   #5                              ; 0529
                64  A5  9F 00280        PUSHAB  DBG$CS_EXITLOOP                 ;
                    52  DD 00283        PUSHL   R2                              ;
                    66  03  FB 00285        CALLS   #3, DBG$NMATCH                  ;
                    01  50  D1 00288        CMPL    R0, #1                          ;
                        14  12 0028B        BNEQ    32$                             ;
                    63  13  90 0028D        MOVB    #19, (VERB_NODE)                ; 0531
                    64  13  90 00290        MOVB    #19, DBG$GB_VERB                ; 0532
                0C  AC  DD 00293        PUSHL   MESSAGE_VECT                    ; 0533
                    0C  BB 00296        PUSHR   #^M<R2,R3>                      ;
00000000G       00  03  FB 00298        CALLS   #3, DBG$NPARSE_EXITLOOP         ;
                    73  11 0029F 31$:    BRB     38$                             ;
```

```
                                01 DD 002A1 32$:    PUSHL   #1                                      ;0536
                             6D A5 9F 002A3         PUSHAB  DBG$CS_FOR
                                52 DD 002A6         PUSHL   R2
                             66 03 FB 002A8         CALLS   #3, DBG$NMATCH
                             01 50 D1 002AB         CMPL    R0, #1
                                14 12 002AE         BNEQ    34$
                             63 19 90 002B0         MOVB    #25, (VERB_NODE)                        ;0538
                             64 19 90 002B3         MOVB    #25, DBG$GB_VERB                        ;0539
                             0C AC DD 002B6         PUSHL   MESSAGE_VECT                            ;0540
                                0C BB 002B9         PUSHR   #^M<R2,R3>
                00000000G    00 03 FB 002BB         CALLS   #3, DBG$NPARSE_FOR
                                73 11 002C2 33$:    BRB     40$
                                01 DD 002C4 34$:    PUSHL   #1                                      ;0543
                             71 A5 9F 002C6         PUSHAB  DBG$CS_GO
                                52 DD 002C9         PUSHL   R2
                             66 03 FB 002CB         CALLS   #3, DBG$NMATCH
                             01 50 D1 002CE         CMPL    R0, #1
                                20 12 002D1         BNEQ    37$
                             63 09 90 002D3         MOVB    #9, (VERB_NODE)                         ;0545
                             64 09 90 002D6         MOVB    #9, DBG$GB_VERB                         ;0546
                             09 68 E9 002D9         BLBC    DBG$GL_SCREEN_NOGO, 35$                 ;0547
                      0002836A 8F DD 002DC         PUSHL   #164714
                             67 01 FB 002E2         CALLS   #1, LIB$SIGNAL
                             0C AC DD 002E5 35$:    PUSHL   MESSAGE_VECT                            ;0548
                                0C BB 002E8         PUSHR   #^M<R2,R3>
                00000000G    00 03 FB 002EA         CALLS   #3, DBG$NPARSE_GO
                                67 11 002F1 36$:    BRB     42$
                                01 DD 002F3 37$:    PUSHL   #1                                      ;0551
                             74 A5 9F 002F5         PUSHAB  DBG$CS_HELP
                                52 DD 002F8         PUSHL   R2
                             66 03 FB 002FA         CALLS   #3, DBG$NMATCH
                             01 50 D1 002FD         CMPL    R0, #1
                                14 12 00300         BNEQ    39$
                             63 0D 90 00302         MOVB    #13, (VERB_NODE)                        ;0553
                             64 0D 90 00305         MOVB    #13, DBG$GB_VERB                        ;0554
                             0C AC DD 00308         PUSHL   MESSAGE_VECT                            ;0555
                                0C BB 0030B         PUSHR   #^M<R2,R3>
                00000000G    00 03 FB 0030D         CALLS   #3, DBG$NPARSE_HELP
                                44 11 00314 38$:    BRB     42$
                                01 DD 00316 39$:    PUSHL   #1                                      ;0558
                             79 A5 9F 00318         PUSHAB  DBG$CS_IF
                                52 DD 0031B         PUSHL   R2
                             66 03 FB 0031D         CALLS   #3, DBG$NMATCH
                             01 50 D1 00320         CMPL    R0, #1
                                14 12 00323         BNEQ    41$
                             63 10 90 00325         MOVB    #16, (VERB_NODE)                        ;0560
                             64 10 90 00328         MOVB    #16, DBG$GB_VERB                        ;0561
                             0C AC DD 0032B         PUSHL   MESSAGE_VECT                            ;0562
                                0C BB 0032E         PUSHR   #^M<R2,R3>
                00000000G    00 03 FB 00330         CALLS   #3, DBG$NPARSE_IF
                                21 11 00337 40$:    BRB     42$
                                02 DD 00339 41$:    PUSHL   #2                                      ;0565
                             7C A5 9F 0033B         PUSHAB  DBG$CS_REPEAT
                                52 DD 0033E         PUSHL   R2
                             66 03 FB 00340         CALLS   #3, DBG$NMATCH
                             01 50 D1 00343         CMPL    R0, #1
                                14 12 00346         BNEQ    43$
```

```
                63              12  90 00348          MOVB    #18, (VERB_NODE)                        ; 0567
                64              12  90 0034B          MOVB    #18, DBG$GB_VERB                        ; 0568
                          0C    AC  DD 0034E          PUSHL   MESSAGE_VECT                            ; 0569
                                0C  BB 00351          PUSHR   #^M<R2,R3>
00000000G       00              03  FB 00353          CALLS   #3, DBG$NPARSE_REPEAT
                                64  11 00357 42$:     BRB     46$
                                03  DD 00359 43$:     PUSHL   #3                                      ; 0572
                          0083  C5  9F 0035B          PUSHAB  DBG$CS_SAVE
                                52  DD 00362          PUSHL   R2
                66              03  FB 00364          CALLS   #3, DBG$NMATCH
                01              50  D1 00367          CMPL    R0, #1
                                11  12 0036A          BNEQ    44$
                63              1F  90 0036C          MOVB    #31, (VERB_NODE)                        ; 0574
                64              1F  90 0036F          MOVB    #31, DBG$GB_VERB                        ; 0575
                                0C  BB 00372          PUSHR   #^M<R2,R3>                              ; 0576
00000000G       00              02  FB 00374          CALLS   #2, DBG$SCR_PARSE_SAVE_CMD
                                64  11 00378          BRB     48$
                                03  DD 0037A 44$:     PUSHL   #3                                      ; 0580
                          0088  C5  9F 0037C          PUSHAB  DBG$CS_SCROLL
                                52  DD 00383          PUSHL   R2
                66              03  FB 00385          CALLS   #3, DBG$NMATCH
                01              50  D1 00388          CMPL    R0, #1
                                11  12 0038B          BNEQ    45$
                63              1D  90 0038D          MOVB    #29, (VERB_NODE)                        ; 0582
                64              1D  90 00390          MOVB    #29, DBG$GB_VERB                        ; 0583
                                0C  BB 00393          PUSHR   #^M<R2,R3>                              ; 0584
00000000G       00              02  FB 00395          CALLS   #2, DBG$SCR_PARSE_SCROLL_CMD
                                43  11 00399          BRB     48$
                                03  DD 0039B 45$:     PUSHL   #3                                      ; 0588
                          008F  C5  9F 0039D          PUSHAB  DBG$CS_SEARCH
                                52  DD 003A4          PUSHL   R2
                66              03  FB 003A6          CALLS   #3, DBG$NMATCH
                01              50  D1 003A9          CMPL    R0, #1
                                14  12 003AC          BNEQ    47$
                63              0F  90 003AE          MOVB    #15, (VERB_NODE)                        ; 0590
                64              0F  90 003B1          MOVB    #15, DBG$GB_VERB                        ; 0591
                          0C    AC  DD 003B4          PUSHL   MESSAGE_VECT                            ; 0592
                                0C  BB 003B7          PUSHR   #^M<R2,R3>
00000000G       00              03  FB 003B9          CALLS   #3, DBG$NPARSE_SEARCH
                                68  11 003BD 46$:     BRB     51$
                                03  DD 003BF 47$:     PUSHL   #3                                      ; 0595
                          0096  C5  9F 003C1          PUSHAB  DBG$CS_SELECT
                                52  DD 003C5          PUSHL   R2
                66              03  FB 003C7          CALLS   #3, DBG$NMATCH
                01              50  D1 003CA          CMPL    R0, #1
                                12  12 003CD          BNEQ    49$
                63              1E  90 003CF          MOVB    #30, (VERB_NODE)                        ; 0597
                64              1E  90 003D2          MOVB    #30, DBG$GB_VERB                        ; 0598
                                0C  BB 003D5          PUSHR   #^M<R2,R3>                              ; 0599
00000000G       00              02  FB 003D7          CALLS   #2, DBG$SCR_PARSE_SELECT_CMD
                          012D  31  03E1 48$:         BRW     64$
                                02  DD 003E4 49$:     PUSHL   #2                                      ; 0603
                          009D  C5  9F 003E6          PUSHAB  DBG$CS_SET
                                52  DD 003EA          PUSHL   R2
                66              03  FB 003EC          CALLS   #3, DBG$NMATCH
                01              50  D1 003EF          CMPL    R0, #1
                                14  12 003F2          BNEQ    50$
```

```
                63              0A  90 003F4          MOVB    #10, (VERB_NODE)                ; 0605
                64              0A  90 003F7          MOVB    #10, DBG$GB_VERB                ; 0606
                          0C    AC  DD 003FA          PUSHL   MESSAGE_VECT                    ; 0607
                                0C  BB 003FD          PUSHR   #^M<R2,R3>
00000000G       00              03  FB 003FF          CALLS   #3, DBG$NPARSE_SET
                                76  11 00406          BRB     55$
                                02  DD 00408 50$:     PUSHL   #2                              ; 0610
                          00A1  C5  9F 0040A          PUSHAB  DBG$CS_SHOW
                                52  DD 0040E          PUSHL   R2
                66              03  FB 00410          CALLS   #3, DBG$NMATCH
                01              50  D1 00413          CMPL    R0, #1
                                14  12 00416          BNEQ    52$
                63              0B  90 00418          MOVB    #11, (VERB_NODE)                ; 0612
                64              0B  90 0041B          MOVB    #11, DBG$GB_VERB                ; 0613
                          0C    AC  DD 0041E          PUSHL   MESSAGE_VECT                    ; 0614
                                0C  BB 00421          PUSHR   #^M<R2,R3>
00000000G       00              03  FB 00423          CALLS   #3, DBG$NPARSE_SHOW
                                76  11 0042A 51$:     BRB     57$
                                02  DD 0042C 52$:     PUSHL   #2                              ; 0617
                          00A6  C5  9F 0042E          PUSHAB  DBG$CS_SPAWN
                                52  DD 00432          PUSHL   R2
                66              03  FB 00434          CALLS   #3, DBG$NMATCH
                01              50  D1 00437          CMPL    R0, #1
                                14  12 0043A          BNEQ    53$
                63              15  90 0043C          MOVB    #21, (VERB_NODE)                ; 0619
                64              15  90 0043F          MOVB    #21, DBG$GB_VERB                ; 0620
                          0C    AC  DD 00442          PUSHL   MESSAGE_VECT                    ; 0621
                                0C  BB 00445          PUSHR   #^M<R2,R3>
00000000G       00              03  FB 00447          CALLS   #3, DBG$NPARSE_SPAWN
                                76  11 0044E          BRB     59$
                                01  DD 00450 53$:     PUSHL   #1                              ; 0624
                          00AC  C5  9F 00452          PUSHAB  DBG$CS_STEP
                                52  DD 00456          PUSHL   R2
                66              03  FB 00458          CALLS   #3, DBG$NMATCH
                01              50  D1 0045B          CMPL    R0, #1
                                20  12 0045E          BNEQ    56$
                63              0C  90 00460          MOVB    #12, (VERB_NODE)                ; 0626
                64              0C  90 00463          MOVB    #12, DBG$GB_VERB                ; 0627
                09              68  E9 00466          BLBC    DBG$GL_SCREEN_NOGO, 54$         ; 0628
                      0002836A  8F  DD 00469          PUSHL   #164714
                67              01  FB 0046F          CALLS   #1, LIB$SIGNAL
                          0C    AC  DD 00472 54$:     PUSHL   MESSAGE_VECT                    ; 0629
                                0C  BB 00475          PUSHR   #^M<R2,R3>
00000000G       00              03  FB 00477          CALLS   #3, DBG$NPARSE_STEP
                                6A  11 0047E 55$:     BRB     61$
                                02  DD 00480 56$:     PUSHL   #2                              ; 0632
                          00B1  C5  9F 00482          PUSHAB  DBG$CS_SYMBOLIZE
                                52  DD 00486          PUSHL   R2
                66              03  FB 00488          CALLS   #3, DBG$NMATCH
                01              50  D1 0048B          CMPL    R0, #1
                                14  12 0048E          BNEQ    58$
                63              17  90 00490          MOVB    #23, (VERB_NODE)                ; 0634
                64              17  90 00493          MOVB    #23, DBG$GB_VERB                ; 0635
                          0C    AC  DD 00496          PUSHL   MESSAGE_VECT                    ; 0636
                                0C  BB 00499          PUSHR   #^M<R2,R3>
00000000G       00              03  FB 0049B          CALLS   #3, DBG$NPARSE_SYMBOLIZE
                                6A  11 004A2 57$:     BRB     63$
```

```
                        01  DD 004A4 58$:    PUSHL   #1                                      ; 0639
                    00BB    C5  9F 004A6      PUSHAB  DBG$CS_TYPE
                            52  DD 004AA      PUSHL   R2
                        66      03  FB 004AC      CALLS   #3, DBG$NMATCH
                        01      50  D1 004AF      CMPL    R0, #1
                                14  12 004B2      BNEQ    60$
                        63      0E  90 004B4      MOVB    #14, (VERB_NODE)                    ; 0641
                        64      0E  90 004B7      MOVB    #14, DBG$GB_VERB                    ; 0642
                          0C    AC  DD 004BA      PUSHL   MESSAGE_VECT                        ; 0643
                                0C  BB 004BD      PUSHR   #^M<R2,R3>
                00000000G   00      03  FB 004BF      CALLS   #3, DBG$NPARSE_TYPE
                                46  11 004C6 59$:    BRB     63$
                                01  DD 004C8 60$:    PUSHL   #1                              ; 0646
                    00C0        C5  9F 004CA      PUSHAB  DBG$CS_UNDEFINE
                                52  DD 004CE      PUSHL   R2
                        66      03  FB 004D0      CALLS   #3, DBG$NMATCH
                        01      50  D1 004D3      CMPL    R0, #1
                                14  12 004D6      BNEQ    62$
                        63      18  90 004D8      MOVB    #24, (VERB_NODE)                    ; 0648
                        64      18  90 004DB      MOVB    #24, DBG$GB_VERB                    ; 0649
                          0C    AC  DD 004DE      PUSHL   MESSAGE_VECT                        ; 0650
                                0C  BB 004E1      PUSHR   #^M<R2,R3>
                00000000G   00      03  FB 004E3      CALLS   #3, DBG$NPARSE_UNDEFINE
                                22  11 004EA 61$:    BRB     63$
                                01  DD 004EC 62$:    PUSHL   #1                              ; 0653
                    00C9        C5  9F 004EE      PUSHAB  DBG$CS_WHILE
                                52  DD 004F2      PUSHL   R2
                        66      03  FB 004F4      CALLS   #3, DBG$NMATCH
                        01      50  D1 004F7      CMPL    R0, #1
                                2B  12 004FA      BNEQ    66$
                        63      11  90 004FC      MOVB    #17, (VERB_NODE)                    ; 0655
                        64      11  90 004FF      MOVB    #17, DBG$GB_VERB                    ; 0656
                          0C    AC  DD 00502      PUSHL   MESSAGE_VECT                        ; 0657
                                0C  BB 00505      PUSHR   #^M<R2,R3>
                00000000G   00      03  FB 00507      CALLS   #3, DBG$NPARSE_WHILE
                                2A      50  E9 0050E 63$:    BLBC    R0, 67$
                                        62  B5 00511 64$:    TSTW    (R2)                    ; 0676
                                        0E  13 00513      BEQL    65$
                                        01  DD 00515      PUSHL   #1                      ; 0677
                                    00CF    C5  9F 00517      PUSHAB  DBG$CS_CR
                                        52  DD 0051B      PUSHL   R2
                                66      03  FB 0051D      CALLS   #3, DBG$NMATCH
                                04      50  E9 00520      BLBC    R0, 66$
                                50      01  D0 00523 65$:    MOVL    #1, R0                  ; 0676
                                            04 00526      RET
                                        52  DD 00527 66$:    PUSHL   R2                      ; 0683
                            0000V   CF      01  FB 00529      CALLS   #1, DBG$NNEXT_WORD
                                        50  DD 0052E      PUSHL   R0
                00000000G   00      01  FB 00530      CALLS   #1, DBG$NSYNTAX_ERROR
                              0C    BC      50  D0 00537      MOVL    R0, @MESSAGE_VECT
                                50      04  D0 0053B 67$:    MOVL    #4, R0                  ; 0688
                                            04 0053E      RET                             ; 0690

; Routine Size:  1343 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  561          0691  1 GLOBAL ROUTINE DBG$NMATCH (STRING_DESC, COUNTED_STRING, UNIQUE_CHARS) =
;  562          0692  1 !
;  563          0693  1 ! FUNCTION
;  564          0694  1 !       This routine is used extensively during command parsing. What it does is
;  565          0695  1 !       to compare the first word of the command input string against the supplied
;  566          0696  1 !       counted string to see if they match. If they do, the matched word is removed
;  567          0697  1 !       from the head of the command input string and success is returned as the value
;  568          0698  1 !       of the routine. Note that the input word may be shorter than the counted
;  569          0699  1 !       string to allow for abbreviations of commands.
;  570          0700  1 !
;  571          0701  1 ! FORMAL PARAMETERS:
;  572          0702  1 !
;  573          0703  1 !       STRING_DESC     - VAX standard string descriptor of the input command.
;  574          0704  1 !
;  575          0705  1 !       COUNTED_STRING  - The counted string against which the input word is
;  576          0706  1 !                         to be compared.
;  577          0707  1 !
;  578          0708  1 !       JNIQUE_CHARS    - The number of characters which the input word must
;  579          0709  1 !                         match against the counted string for purposes of
;  580          0710  1 !                         disambiguating the input word.
;  581          0711  1 !
;  582          0712  1 ! IMPLICIT INPUTS:
;  583          0713  1 !
;  584          0714  1 !       NONE
;  585          0715  1 !
;  586          0716  1 ! IMPLICIT OUTPUTS:
;  587          0717  1 !
;  588          0718  1 !       NONE
;  589          0719  1 !
;  590          0720  1 ! ROUTINE VALUE:
;  591          0721  1 !
;  592          0722  1 !       Unsigned integer longword.
;  593          0723  1 !
;  594          0724  1 ! COMPLETION CODES:
;  595          0725  1 !
;  596          0726  1 !       STS$K_SEVERE (4)  - The input word did not match the counted string.
;  597          0727  1 !
;  598          0728  1 !       STS$K_SUCCESS (1) - The input word did match.
;  599          0729  1 !
;  600          0730  1 ! SIDE EFFECTS:
;  601          0731  1 !
;  602          0732  1 !       On a successful match, the input string descriptor is updated to just beyond
;  603          0733  1 !       the word of input matched against the counted string.
;  604          0734  1 !
;  605          0735  1
;  606          0736  2     BEGIN
;  607          0737  2
;  608          0738  2     MAP
;  609          0739  2         STRING_DESC : REF DBG$STG_DESC,
;  610          0740  2         COUNTED_STRING: REF VECTOR[,BYTE];
;  611          0741  2
;  612          0742  2     LOCAL
;  613          0743  2         WORD_STRING: REF VECTOR[,BYTE], ! Holds the next input word
;  614          0744  2         SAVE_PTR,                       ! Saves the address of the input buffer
;  615          0745  2         SAVE_LEN,                       ! Saves the length of the input buffer
;  616          0746  2         MATCH_FLAG;                     ! Signals a match
;  617          0747  2
```

```
;  618          0748  2
;  619          0749  2
;  620          0750  2      ! Save the original descriptor buffer address and length
;  621          0751  2      !
;  622          0752  2      SAVE_PTR = .STRING_DESC [DSC$A_POINTER];
;  623          0753  2      SAVE_LEN = .STRING_DESC [DSC$W_LENGTH];
;  624          0754  2
;  625          0755  2
;  626          0756  2      ! Get the next input word
;  627          0757  2      !
;  628          0758  2      WORD_STRING = DBG$NNEXT_WORD (.STRING_DESC);
;  629          0759  2
;  630          0760  2
;  631          0761  2      ! Perform the match, check for special cases
;  632          0762  2      !
;  633          0763  2      MATCH_FLAG = TRUE;
;  634          0764  2      SELECTONE TRUE OF
;  635          0765  2          SET
;  636          0766  2
;  637          0767  2
;  638          0768  2          ! Match
;  639          0769  2          !
;  640          0770  2          [.COUNTED_STRING [0] EQL 0]:
;  641          0771  2              0;
;  642          0772  2
;  643          0773  2
;  644          0774  2          ! No match
;  645          0775  2          !
;  646          0776  2          [.WORD_STRING [0] GTR .COUNTED_STRING [0]]:
;  647          0777  2              MATCH_FLAG = FALSE;
;  648          0778  2
;  649          0779  2
;  650          0780  2          ! No match
;  651          0781  2          !
;  652          0782  2          [.WORD_STRING [0] EQL 0]:
;  653          0783  2              MATCH_FLAG = FALSE;
;  654          0784  2
;  655          0785  2
;  656          0786  2          ! No match
;  657          0787  2          !
;  658          0788  2          [.WORD_STRING [0] LSS .UNIQUE_CHARS]:
;  659          0789  2              MATCH_FLAG = FALSE;
;  660          0790  2
;  661          0791  2
;  662          0792  2          ! Normal match
;  663          0793  2          !
;  664          0794  2          [OTHERWISE]:
;  665          0795  3              BEGIN
;  666          0796  3
;  667          0797  3              LOCAL
;  668          0798  3                  I;                          ! Loop counter
;  669          0799  3
;  670          0800  3              I = 1;
;  671          0801  3              WHILE .I LEQ .WORD_STRING [0] DO
;  672          0802  4                  BEGIN
;  673          0803  4                  IF .WORD_STRING [.I] NEQ .COUNTED_STRING [.I]
;  674          0804  4                  THEN
```

```
;  675     0805  4                    MATCH_FLAG = FALSE;
;  676     0806  4
;  677     0807  4                I = .I + 1;
;  678     0808  3                END;
;  679     0809  3
;  680     0810  2            END;
;  681     0811  2
;  682     0812  2        TES;
;  683     0813  2
;  684     0814  2
;  685     0815  2    ! Return successfully if we found a match.  Otherwise, restore the old
;  686     0816  2    ! descriptor and return failure to find a match.
;  687     0817  2    !
;  688     0818  2    IF .MATCH_FLAG THEN RETURN STS$K_SUCCESS;
;  689     0819  2    STRING_DESC [DSC$A_POINTER] = .SAVE_PTR;
;  690     0820  2    STRING_DESC [DSC$W_LENGTH] = .SAVE_LEN;
;  691     0821  2    RETURN STS$K_SUCCESS;
;  692     0822  2
;  693     0823  1    END;


                                                003C 00000         .ENTRY  DBG$NMATCH, Save R2,R3,R4,R5                    ; 0691
                                          52    04   AC  D0 00002  MOVL    STRING_DESC, R2                                 ; 0752
                                          55    04   A2  D0 00006  MOVL    4(R2), SAVE_PTR                                 ;
                                          54         62  3C 0000A  MOVZWL  (R2), SAVE_LEN                                 ; 0753
                                                     52  DD 0000D  PUSHL   R2                                            ; 0758
                                    0000V CF         01  FB 0000F  CALLS   #1, DBG$NNEXT_WORD                            ;
                                          53         01  D0 00014  MOVL    #1, MATCH_FLAG                                  ; 0763
                                                08   BC  95 00017  TSTB    @COUNTED_STRING                                 ; 0770
                                                     2E  13 0001A  BEQL    5$                                              ;
                                               08    BC  60 91 0001C  CMPB    (WORD_STRING), @COUNTED_STRING                  ; 0776
                                                     0C  1A 00020  BGTRU   1$                                              ;
                                                     60  95 00022  TSTB    (WORD_STRING)                                   ; 0782
                                                     08  13 00024  BEQL    1$                                              ;
                                    0C    AC         60      08  00  ED 00026  CMPZV   #0, #8, (WORD_STRING), UNIQUE_CHARS  ; 0788
                                                     04  18 0002C  BGEQ    2$                                              ;
                                                     53  D4 0002E  1$:  CLRL    MATCH_FLAG                                  ; 0789
                                                     18  11 00030  BRB     5$                                              ;
                                          51         01  D0 00032  2$:  MOVL    #1, I                                       ; 0800
                                          51         60      08  00  ED 00035  3$:  CMPZV   #0, #8, (WORD_STRING), I       ; 0801
                                                     0E  19 0003A  BLSS    5$                                              ;
                                               08 BC41          6140  91 0003C  CMPB    (I)[WORD_STRING], @COUNTED_STRING[I]  ; 0803
                                                     02  13 00042  BEQL    4$                                              ;
                                                     53  D4 00044  CLRL    MATCH_FLAG                                  ; 0805
                                                     51  D6 00046  4$:  INCL    I                                           ; 0807
                                                     EB  11 00048  BRB     3$                                              ; 0801
                                          04         53  E9 0004A  5$:  BLBC    MATCH_FLAG, 6$                              ; 0818
                                          50         01  D0 0004D  MOVL    #1, R0                                          ;
                                                         04 00050  RET                                                     ;
                                               04   A2        55  D0 00051  6$:  MOVL    SAVE_PTR, 4(R2)                 ; 0819
                                                    62        54  B0 00055  MOVW    SAVE_LEN, (R2)                  ; 0820
                                                    50        04  D0 00058  MOVL    #4, R0                          ; 0821
                                                                  04 0005B  RET                                             ; 0823
```

; Routine Size: 92 bytes,   Routine Base: DBG$CODE + 053F

```
;  695          0824  1 GLOBAL ROUTINE DBG$NNEXT_WORD (STRING_DESC) =
;  696          0825  1 !
;  697          0826  1 ! FUNCTION
;  698          0827  1 !       Routine DBG$NNEXT_WORD isolates the next word of the input command. A word
;  699          0828  1 !       is defined to be any number of alphabetic characters delimited by blanks
;  700          0829  1 !       or a single non-alphabetic character. The word isolated is represented as
;  701          0830  1 !       a counted string.
;  702          0831  1 !
;  703          0832  1 !       The input string is assumed to be terminated by a <cr>.
;  704          0833  1 !
;  705          0834  1 !       Maximum word size is 80 characters.
;  706          0835  1 !
;  707          0836  1 ! FORMAL PARAMETERS:
;  708          0837  1 !
;  709          0838  1 !       STRING_DESC     - VAX standard string descriptor of the input command.
;  710          0839  1 !
;  711          0840  1 ! IMPLICIT INPUTS:
;  712          0841  1 !
;  713          0842  1 !       WORD_BUF        - OWNed byte vector to contain the word of input.
;  714          0843  1 !
;  715          0844  1 ! IMPLICIT OUTPUTS:
;  716          0845  1 !
;  717          0846  1 !       NONE
;  718          0847  1 !
;  719          0848  1 ! ROUTINE VALUE:
;  720          0849  1 !
;  721          0850  1 !       The address of WORD_BUF, the counted string representing the next word
;  722          0851  1 !       of input.
;  723          0852  1 !
;  724          0853  1 ! COMPLETION CODES:
;  725          0854  1 !
;  726          0855  1 !       NONE
;  727          0856  1 !
;  728          0857  1 ! SIDE EFFECTS:
;  729          0858  1 !
;  730          0859  1 !       WORD_BUF is always filled with the next word of input. WORD_BUF[0]
;  731          0860  1 !       contains the count of the characters in the word.  On exhausted input,
;  732          0861  1 !       WORD_BUF[0] equals 0.  The command string descriptor is updated past
;  733          0862  1 !       the word of input.
;  734          0863  1 !
;  735          0864  1
;  736          0865  2     BEGIN
;  737          0866  2
;  738          0867  2     MAP
;  739          0868  2         STRING_DESC : REF DBG$STG_DESC; ! Input string descriptor
;  740          0869  2
;  741          0870  2     LOCAL
;  742          0871  2         CHAR : BYTE,                    ! Holds next character of input
;  743          0872  2         POINTER,                        ! Temp pointer
;  744          0873  2         COUNT;                          ! Count of characters
;  745          0874  2
;  746          0875  2
;  747          0876  2
;  748          0877  2     ! Check for exhausted input
;  749          0878  2     !
;  750          0879  2     IF .STRING_DESC [DSC$W_LENGTH] LEQ 0
;  751          0880  2     THEN
```

```
;  752          0881  3          BEGIN
;  753          0882  3          STRING_DESC [DSC$W_LENGTH] = 0;
;  754          0883  3          WORD_BUF [0] = 0;                    ! No word
;  755          0884  3          STRING_DESC [DSC$A_POINTER] = 0;
;  756          0885  3          RETURN WORD_BUF [0];
;  757          0886  2          END;
;  758          0887  2
;  759          0888  2
;  760          0889  2      ! Ignore leading white space
;  761          0890  2      !
;  762          0891  2      WHILE CH$RCHAR (.STRING_DESC [DSC$A_POINTER]) EQL ' ' DO
;  763          0892  3          BEGIN
;  764          0893  3          STRING_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_DESC [DSC$A_POINTER], 1);
;  765          0894  3          STRING_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH] - 1;
;  766          0895  2          END;
;  767          0896  2
;  768          0897  2
;  769          0898  2      ! Count the number of characters in the next word.  Note that we always
;  770          0899  2      ! return at least one character.
;  771          0900  2      !
;  772          0901  2      POINTER = .STRING_DESC [DSC$A_POINTER];
;  773          0902  2      CHAR = CH$RCHAR (.STRING_DESC [DSC$A_POINTER]);
;  774          0903  3      IF (.CHAR GEQ 'A' AND .CHAR LEQ 'Z')
;  775          0904  2      THEN
;  776          0905  3          BEGIN
;  777          0906  3
;  778          0907  3
;  779          0908  3          ! We take more than one char
;  780          0909  3          !
;  781          0910  3          CHAR = CH$A_RCHAR (STRING_DESC [DSC$A_POINTER]);
;  782          0911  3          STRING_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH] - 1;
;  783          0912  3          WHILE ((.CHAR GEQ 'A' AND .CHAR LEQ 'Z') OR .CHAR EQL ' ') AND
;  784          0913  3                CH$DIFF (.STRING_DESC [DSC$A_POINTER], .POINTER) [EQU WORD_SIZE
;  785          0914  3          DO
;  786          0915  4              BEGIN
;  787          0916  4              CHAR = CH$A_RCHAR (STRING_DESC [DSC$A_POINTER]);
;  788          0917  4              STRING_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH] - 1;
;  789          0918  3              END;
;  790          0919  3
;  791          0920  3          END
;  792          0921  3
;  793          0922  3
;  794          0923  3      ! Take one character only.
;  795          0924  3      !
;  796          0925  2      ELSE
;  797          0926  3          BEGIN
;  798          0927  3          CHAR = CH$A_RCHAR (STRING_DESC [DSC$A_POINTER]);
;  799          0928  3          STRING_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH] - 1;
;  800          0929  2          END;
;  801          0930  2
;  802          0931  2
;  803          0932  2      ! Calculate the number of characters in the new word
;  804          0933  2      !
;  805          0934  2      COUNT = CH$DIFF (.STRING_DESC [DSC$A_POINTER], .POINTER);
;  806          0935  2
;  807          0936  2
;  808          0937  2      ! Now copy over th appropriate number of chars.
```

```
;  809          0938  2     !
;  810          0939  2     WORD_BUF [0] = .COUNT;
;  811          0940  2     CH$MOVE (.COUNT, .POINTER, WORD_BUF [1]);
;  812          0941  2
;  813          0942  2
;  814          0943  2     ! Check for exhausted input
;  815          0944  2     !
;  816          0945  2     IF .STRING_DESC [DSC$W_LENGTH] LEQ 0
;  817          0946  2     THEN
;  818          0947  3         BEGIN
;  819          0948  3         STRING_DESC [DSC$W_LENGTH] = 0;
;  820          0949  3         STRING_DESC [DSC$A_POINTER] = 0;
;  821          0950  2         END;
;  822          0951  2
;  823          0952  2     RETURN WORD_BUF[0];
;  824          0953  2
;  825          0954  1     END;


                                01FC 00000          .ENTRY  DBG$NNEXT_WORD, Save R2,R3,R4,R5,R6,R7,R8   ; 0824
                58 00000000'    EF   9E 00002       MOVAB   WORD_BUF, R8                                ;
                57        04    AC   D0 00009       MOVL    STRING_DESC, R7                             ; 0879
                                67   B5 0000D       TSTW    (R7)                                        ;
                                09   12 0000F       BNEQ    1$                                          ;
                                67   B4 00011       CLRW    (R7)                                        ; 0882
                                68   94 00013       CLRB    WORD_BUF                                    ; 0883
                          04    A7   D4 00015       CLRL    4(R7)                                       ; 0884
                                68   11 00018       BRB     9$                                          ; 0885
                56        04    A7   9E 0001A 1$:   MOVAB   4(R7), R6                                   ; 0891
                20        00    B6   91 0001E 2$:   CMPB    @0(R6), #32                                 ;
                                06   12 00022       BNEQ    3$                                          ;
                                66   D6 00024       INCL    (R6)                                        ; 0893
                                67   B7 00026       DECW    (R7)                                        ; 0894
                                F4   11 00028       BRB     2$                                          ; 0891
                52              66   D0 0002A 3$:   MOVL    (R6), POINTER                               ; 0901
                50        00    B6   90 0002D       MOVB    @0(R6), CHAR                                ; 0902
                      41  8F        50   91 00031       CMPB    CHAR, #65                               ; 0903
                                2F   1F 00035       BLSSU   7$                                          ;
                      5A  8F        50   91 00037       CMPB    CHAR, #90                               ;
                                29   1A 0003B       BGTRU   7$                                          ;
                                66   D6 0003D 4$:   INCL    (R6)                                        ; 0910
                50        00    B6   90 0003F       MOVB    @0(R6), CHAR                                ;
                                67   B7 00043       DECW    (R7)                                        ; 0911
                      41  8F        50   91 00045       CMPB    CHAR, #65                               ; 0912
                                06   1F 00049       BLSSU   5$                                          ;
                      5A  8F        50   91 0004B       CMPB    CHAR, #90                               ;
                                06   1B 0004F       BLEQU   6$                                          ;
                      5F  8F        50   91 00051 5$:   CMPB    CHAR, #95                               ;
                                17   12 00055       BNEQ    8$                                          ;
          51                    66   52   C3 00057 6$:   SUBL3   POINTER, (R6), R1                      ; 0913
                00000050    8F        51   D1 0005B       CMPL    R1, #80                                 ;
                                0A   1A 00062       BGTRU   8$                                          ;
                                D7   11 00064       BRB     4$                                          ; 0916
                                66   D6 00066 7$:   INCL    (R6)                                        ; 0927
```

```
                    50      00    B6  90 00068          MOVB    @0(R6), CHAR                    ;
                                  67  B7 0006C          DECW    (R7)                            ; 0928
                50          66    52  C3 0006E 8$:     SUBL3   POINTER, (R6), COUNT            ; 0934
                            68    50  90 00072          MOVB    COUNT, WORD_BUF                 ; 0939
          01    A8          62    50  28 00075          MOVC3   COUNT, (POINTER), WORD_BUF+1    ; 0940
                                  67  B5 0007A          TSTW    (R7)                            ; 0945
                                  04  12 0007C          BNEQ    9$                              ;
                                  67  B4 0007E          CLRW    (R7)                            ; 0948
                                  66  D4 00080          CLRL    (R6)                            ; 0949
                    50            68  9E 00082 9$:     MOVAB   WORD_BUF, R0                    ; 0952
                                      04 00085          RET                                     ; 0954

; Routine Size:  134 bytes,    Routine Base:  DBG$CODE + 059B
```

```
;  827     0955  1 GLOBAL ROUTINE DBG$NPARSE_EXPRESSION(INPUT_DESC, RADIX,
;  828     0956  1                                      VALUE_DESC_PTR, TERM_INDEX, MESSAGE_VECT) =
;  829     0957  1 !
;  830     0958  1 ! FUNCTION
;  831     0959  1 !       This routine interfaces to the Expression Interpreter.
;  832     0960  1 !       It first checks whether the expression is a DEFINEd name, in
;  833     0961  1 !       which case the expansion is done.
;  834     0962  1 !
;  835     0963  1 ! FORMAL PARAMETERS:
;  836     0964  1 !
;  837     0965  1 !       INPUT_DESC      - The address of a VAX Standard ASCII string descriptor
;  838     0966  1 !                         which describes the user input.
;  839     0967  1 !
;  840     0968  1 !       RADIX           - A longword containing an integer encoding of the
;  841     0969  1 !                         radix to be used for the interpretation of numeric
;  842     0970  1 !                         literals.
;  843     0971  1 !
;  844     0972  1 !       VALUE_DESC_PTR  - The address of a longword to contain the address
;  845     0973  1 !                         of a language specific value descriptor.
;  846     0974  1 !
;  847     0975  1 !       TERM_INDEX      - The "terminator index" value which indicates which
;  848     0976  1 !                         lexical tokens are valid expression terminators in
;  849     0977  1 !                         the current context.  For example, in the IF command,
;  850     0978  1 !                         the keyword "THEN" is the valid expression termina-
;  851     0979  1 !                         tor.  These index values have names of the form
;  852     0980  1 !                         TOKEN$K_TERM_xxx.
;  853     0981  1 !
;  854     0982  1 !       MESSAGE_VECT    - The address of a longword to contain the address
;  855     0983  1 !                         of a message argument vector for errors.
;  856     0984  1 !
;  857     0985  1 !       6th Optional Parameter - If this is present, and the value is
;  858     0986  1 !                         DBG$K_DEPOSIT_VERB then pass this into DBG$EXP_INT,
;  859     0987  1 !                         and into DBG$EXPRESSION_PARSER, so that in
;  860     0988  1 !                         DBG$EXPRESSION_PARSER, when the expression is not
;  861     0989  1 !                         address expression and in deposit command,
;  862     0990  1 !                         DBG$EVAL_LANG_OPERATOR will not be called with
;  863     0991  1 !                         DBG$GL_IDENTITY_TOKEN.
;  864     0992  1 !
;  865     0993  1 ! IMPLICIT INPUTS:
;  866     0994  1 !
;  867     0995  1 !       NONE
;  868     0996  1 !
;  869     0997  1 ! IMPLICIT OUTPUTS:
;  870     0998  1 !
;  871     0999  1 !       NONE
;  872     1000  1 !
;  873     1001  1 ! ROUTINE VALUE:
;  874     1002  1 !
;  875     1003  1 !       Same as the Expression Interpreter.
;  876     1004  1 !
;  877     1005  1 ! COMPLETION CODES:
;  878     1006  1 !
;  879     1007  1 !       Same as the Expression Interpreter.
;  880     1008  1 !
;  881     1009  1 ! SIDE EFFECTS:
;  882     1010  1 !
;  883     1011  1 !       Same as the Expression Interpreter.
```

```
;  884          1012  1 !
;  885          1013  1
;  886          1014  2     BEGIN
;  887          1015  2
;  888          1016  2     BUILTIN ACTUALCOUNT,ACTUALPARAMETER;
;  889          1017  2
;  890          1018  2
;  891          1019  2     ! Translate default radix.
;  892          1020  2     !
;  893          1021  2     IF .RADIX EQL DBG$K_DEFAULT
;  894          1022  2     THEN
;  895          1023  2         RADIX = .DBG$GB_RADIX[DBG$B_RADIX_INPUT];
;  896          1024  2
;  897          1025  2     IF ACTUALCOUNT() GTR 5
;  898          1026  2     THEN
;  899          1027  3         BEGIN
;  900          1028  3         IF ACTUALPARAMETER(6) NEQ DBG$K_DEPOSIT_VERB
;  901          1029  3         THEN
;  902          1030  3             $DBG_ERROR('DBGNPARSE\DBG$NPARSE_EXPRESSION');
;  903          1031  3
;  904          1032  3         RETURN DBG$EXP_INT (.INPUT_DESC, .RADIX, .VALUE_DESC_PTR, .TERM_INDEX,
;  905          1033  3                 DBG$K_DEPOSIT_VERB);
;  906          1034  3         END
;  907          1035  3
;  908          1036  2     ELSE
;  909          1037  2         RETURN DBG$EXP_INT (.INPUT_DESC, .RADIX, .VALUE_DESC_PTR, .TERM_INDEX);
;  910          1038  1     END;


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

24 47 42 44 5C 45 53 52 41 50 4E 47 42 44 1F  000D1 P.ABI:    .ASCII  <31>\DBGNPARSE\<92>\DBG$NPARSE_EXPRESS\     ;
   53 53 45 52 50 58 45 5F 45 53 52 41 50 4E  000E0                                                           ;
                                              000EE          .ASCII  \ION\                                      ;


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

                                                 0004 00000     .ENTRY  DBG$NPARSE_EXPRESSION, Save R2          ; 0955
                                  52 00000000G   00  9E 00002  MOVAB   DBG$EXP_INT, R2                         ;
                                  01        08   AC  D1 00009  CMPL    RADIX, #1                             ; 1021
                                               08  12 0000D  BNEQ    1$                                    ;
                              08  AC 00000000G   00  9A 0000F  MOVZBL  DBG$GB_RADIX, RADIX                   ; 1023
                                  05            6C  91 00017 1$: CMPB    (AP), #5                              ; 1025
                                               29  1B 0001A  BLEQU   3$                                     ;
                                  05        18   AC  D1 0001C  CMPL    24(AP), #5                            ; 1028
                                               15  13 00020  BEQL    2$                                    ;
                                     00000000'  EF  9F 00022  PUSHAB  P.ABI                                 ; 1030
                                               01  DD 00028  PUSHL   #1                                    ;
                                     00028362   8F  DD 0002A  PUSHL   #164706                               ;
                          00000000G  00         03  FB 00030  CALLS   #3, LIB$SIGNAL                       ;
                                               05  DD 00037 2$: PUSHL   #5                                  ; 1032
                                  7E        0C   AC  7D 00039  MOVQ    VALUE_DESC_PTR, -(SP)                 ;
                                  7E        04   AC  7D 0003D  MOVQ    INPUT_DESC, -(SP)                     ;
                                  62             05  FB 00041  CALLS   #5, DBG$EXP_INT                       ;
```

```
                                04 00044          RET                                     ; 1037
                7E      0C  AC  7D 00045 3$:      MOVQ    VALUE_DESC_PTR, -(SP)           ;
                7E      04  AC  7D 00049          MOVQ    INPUT_DESC, -(SP)               ;
                62          04  FB 0004D          CALLS   #4, DBG$EXP_INT                 ;
                                04 00050          RET                                     ; 1038

; Routine Size:  81 bytes,    Routine Base:  DBG$CODE + 0621
```

```
;  912     1039  1 GLOBAL ROUTINE DBG$NPARSE_ADDRESS(INPUT_DESC, ADDR_EXP_PTR,
;  913     1040  1                                   RADIX, TERM_INDEX, MESSAGE_VECT) =
;  914     1041  1 !
;  915     1042  1 ! FUNCTION
;  916     1043  1 !       This routine interfaces to the Address Expression Interpreter to obtain
;  917     1044  1 !       an address expression descriptor. Address expression descriptors are
;  918     1045  1 !       used to describe the object of commands which use address expressions.
;  919     1046  1 !       The descriptor may contain a pointer to a primary descriptor, a pointer
;  920     1047  1 !       to a permanent symbol descriptor, an untyped L-value, or an L-value of
;  921     1048  1 !       type instruction.
;  922     1049  1 !
;  923     1050  1 ! INPUTS
;  924     1051  1 !       INPUT_DESC      - A longword containing the address of a standard
;  925     1052  1 !                         string descriptor describing the input command.
;  926     1053  1 !
;  927     1054  1 !       ADDR_EXP_PTR    - The address of a longword to contain the address
;  928     1055  1 !                         of an address expression descriptor.
;  929     1056  1 !
;  930     1057  1 !       RADIX           - A longword containing the integer code of the
;  931     1058  1 !                         radix to be used in interpreting numeric literals.
;  932     1059  1 !
;  933     1060  1 !       TERM_INDEX      - The "terminator index" value which indicates which
;  934     1061  1 !                         lexical tokens are valid expression terminators in
;  935     1062  1 !                         the current context.  For example, in the EXAMINE
;  936     1063  1 !                         command, comma and colon are valid terminators (as
;  937     1064  1 !                         is carriage-return).  These index values have names
;  938     1065  1 !                         of the form TOKEN$K_TERM_xxx.
;  939     1066  1 !
;  940     1067  1 !       MESSAGE_VECT    - The address of a longword to contain the address
;  941     1068  1 !                         of a message argument vector for errors.
;  942     1069  1 !
;  943     1070  1 ! OUTPUTS
;  944     1071  1 !       INPUT_DESC      - The input string descriptor is updated to point to
;  945     1072  1 !                         the first character after the address expression
;  946     1073  1 !                         that was just parsed.
;  947     1074  1 !
;  948     1075  1 !       ADDR_EXP_PTR    - An Address Expression Descriptor is constructed and
;  949     1076  1 !                         its address is returned to ADDR_EXP_PTR.
;  950     1077  1 !
;  951     1078  1 !       MESSAGE_VECT    - If an error is encountered, a message argument vector
;  952     1079  1 !                         may be constructed (unless the error is singalled)
;  953     1080  1 !                         and its address returned to MESSAGE_VECT.
;  954     1081  1 !
;  955     1082  1 !       An unsigned integer longword completion code is returned as the routine
;  956     1083  1 !                         value.  This is whatever completion code the Address
;  957     1084  1 !                         Expression Interpreter returns.
;  958     1085  1 !
;  959     1086  2     BEGIN
;  960     1087  2
;  961     1088  2     MAP
;  962     1089  2         INPUT_DESC: REF DBG$STG_DESC;
;  963     1090  2
;  964     1091  2     BIND
;  965     1092  2         DBG$CS_CR = UPLIT BYTE (1, DBG$K_CAR_RETURN);
;  966     1093  2
;  967     1094  2     LOCAL
;  968     1095  2         LENGTH,                         ! Current location length
```

```
;  969          1096  2          STATUS,                         ! Return status
;  970          1097  2          TYPE;                           ! Current location type
;  971          1098  2
;  972          1099  2
;  973          1100  2
;  974          1101  2      ! Initialize type and length
;  975          1102  2      !
;  976          1103  2      TYPE = DBG$K_NOTYPE;
;  977          1104  2      LENGTH = 0;
;  978          1105  2
;  979          1106  2
;  980          1107  2      ! Translate default radix.
;  981          1108  2      !
;  982          1109  2      IF .RADIX EQL DBG$K_DEFAULT
;  983          1110  2      THEN
;  984          1111  2          RADIX = .DBG$GB_RADIX[DBG$B_RADIX_INPUT];
;  985          1112  2
;  986          1113  2
;  987          1114  2      ! Now call the Address Expression Interpreter appropriate for the current
;  988          1115  2      ! language.
;  989          1116  2      !
;  990          1117  2      STATUS = DBG$ADDR_EXP_INT(.INPUT_DESC, .ADDR_EXP_PTR,
;  991          1118  2                                TYPE, LENGTH, .RADIX, .TERM_INDEX);
;  992          1119  2
;  993          1120  2      ! Set the potential type and length for current location.
;  994          1121  2      !
;  995          1122  2      DBG$NSET_LAST_TYPLEN (.TYPE, .LENGTH);
;  996          1123  2      RETURN .STATUS;
;  997          1124  2
;  998          1125  1      END;


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

                                    0D  01  000F1 P.ABJ:  .BYTE   1, 13                                   ;

                                                DBG$CS_CR=          P.ABJ


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

                                      0004 00000        .ENTRY  DBG$NPARSE_ADDRESS, Save R2                 ; 1039
                        7E      80    8F   9A 00002        MOVZBL  #128, TYPE                                  ; 1103
                                      7E   D4 00006        CLRL    LENGTH                                      ; 1104
                        01      0C    AC   D1 00008        CMPL    RADIX, #1                                   ; 1109
                                      08   12 0000C        BNEQ    1$                                          ;
                    0C  AC  00000000G 00   9A 0000E        MOVZBL  DBG$GB_RADIX, RADIX                         ; 1111
                        7E      0C    AC   7D 00016 1$:     MOVQ    RADIX, -(SP)                                ; 1118
                                0C    AE   9F 0001A        PUSHAB  LENGTH                                      ; 1117
                                10    AE   9F 0001D        PUSHAB  TYPE                                        ;
                        7E      04    AC   7D 00020        MOVQ    INPUT_DESC, -(SP)                           ;
                00000000G       00          FB 00024        CALLS   #6, DBG$ADDR_EXP_INT                        ;
                        52              50   D0 0002B        MOVL    R0, STATUS                                  ;
                                      6E   DD 0002E        PUSHL   LENGTH                                      ; 1122
                                0C    AE   DD 00030        PUSHL   TYPE                                        ;
                00000000G       00          FB 00033        CALLS   #2, DBG$NSET_LAST_TYPLEN                    ;
```

```
                                50              52  D0 0003A        MOVL    STATUS, R0                      ; 1123
                                                    04 0003D        RET                                     ; 1125

; Routine Size:  62 bytes,    Routine Base:  DBG$CODE + 0672
```

```
;1000        1126  1 GLOBAL ROUTINE DBG$NSAVE_DECIMAL_INTEGER(INPUT_DESC, RESULT) =
;1001        1127  1 !
;1002        1128  1 ! FUNCTION
;1003        1129  1 !       Extracts a numeric string from the command input string and converts it
;1004        1130  1 !       to a binary longword integer.  Decimal radix is assumed for the input.
;1005        1131  1 !
;1006        1132  1 ! INPUTS
;1007        1133  1 !       INPUT_DESC - The address of a standard ASCII string descriptor
;1008        1134  1 !                    pointing to the input command line to be scanned.
;1009        1135  1 !
;1010        1136  1 !       RESULT  - The address of a longword to receive the result, meaning
;1011        1137  1 !                 the scanned integer value in internal format.
;1012        1138  1 !
;1013        1139  1 ! OUTPUTS
;1014        1140  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;1015        1141  1 !                    first character after the scanned integer constant.
;1016        1142  1 !
;1017        1143  1 !       RESULT  - The scanned integer value (in internal format) is returned
;1018        1144  1 !                 the RESULT location.
;1019        1145  1 !
;1020        1146  1 !       The value STS$K_SUCCESS is always returned as the routine value.
;1021        1147  1 !
;1022        1148  1
;1023        1149  2     BEGIN
;1024        1150  2
;1025        1151  2     BUILTIN
;1026        1152  2         EMUL;                           ! Multiply and add two longwords
;1027        1153  2                                         !       to produce quadword result.
;1028        1154  2
;1029        1155  2     MAP
;1030        1156  2         INPUT_DESC : REF DBG$STG_DESC;  ! Input string descriptor
;1031        1157  2
;1032        1158  2     BIND
;1033        1159  2         DBG$CS_CR = UPLIT BYTE(1, DBG$K_CAR_RETURN);
;1034        1160  2
;1035        1161  2     LOCAL
;1036        1162  2         VALUE: VECTOR[2],               ! Result quadword
;1037        1163  2         STRING_PTR,                     ! Pointer to input string
;1038        1164  2         TRUNC_FLAG,                     ! Indicates truncation
;1039        1165  2         CHAR;                           ! Holds characters
;1040        1166  2
;1041        1167  2
;1042        1168  2
;1043        1169  2     ! Check for null input
;1044        1170  2     !
;1045        1171  2     IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
;1046        1172  2     THEN
;1047        1173  2         SIGNAL(DBG$_NEEDMORE);
;1048        1174  2
;1049        1175  2
;1050        1176  2     ! Delete leading white space
;1051        1177  2     !
;1052        1178  2     STRING_PTR = .INPUT_DESC [DSC$A_POINTER];
;1053        1179  2     WHILE .INPUT_DESC [DSC$W_LENGTH] GTR 0 DO
;1054        1180  3         BEGIN
;1055        1181  3         CHAR = CH$RCHAR_A (STRING_PTR);
;1056        1182  3         INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
```

```
;  1057          1183  3          IF (.CHAR NEQ DBG$K_BLANK) AND (.CHAR NEQ DBG$K_TAB) THEN EXITLOOP;
;  1058          1184  2          END;
;  1059          1185  2
;  1060          1186  2
;  1061          1187  2      ! Check for a numeric character. If we don't have one - syntax error.
;  1062          1188  2      !
;  1063          1189  3      IF (.CHAR LSS '0') OR (.CHAR GTR '9')
;  1064          1190  2      THEN
;  1065          1191  3          BEGIN
;  1066          1192  3          INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
;  1067          1193  3          DBG$SYNTAX_ERROR(.INPUT_DESC);
;  1068          1194  2          END;
;  1069          1195  2
;  1070          1196  2
;  1071          1197  2      ! Now continue to accept numeric chars and convert them.  Loop until we
;  1072          1198  2      ! find a non-numeric character.  In the loop, append each new digit to
;  1073          1199  2      ! the number being accumulated, check for overflow, and loop for the next
;  1074          1200  2      ! character.
;  1075          1201  2      !
;  1076          1202  2      VALUE [0] = 0;
;  1077          1203  2      VALUE [1] = 0;
;  1078          1204  2      TRUNC_FLAG = FALSE;
;  1079          1205  2      WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '9') DO
;  1080          1206  3          BEGIN
;  1081          1207  3          CHAR = .CHAR - '0';
;  1082          1208  3          EMUL (VALUE [0], %REF (10), CHAR, VALUE);
;  1083          1209  3          IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
;  1084          1210  3          INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
;  1085          1211  3          CHAR = CH$RCHAR_A (STRING_PTR);
;  1086          1212  2          END;
;  1087          1213  2
;  1088          1214  2
;  1089          1215  2      ! If the next character is an alphabetic character, signal a syntax error.
;  1090          1216  2      ! This can happen if the user tries to enter a hex number, for example.
;  1091          1217  2      !
;  1092          1218  3      IF (.CHAR GEQ 'A') AND (.CHAR LEQ 'Z')
;  1093          1219  2      THEN
;  1094          1220  2          SIGNAL(DBG$_INVDIGDEC, 2, 1, CHAR);
;  1095          1221  2
;  1096          1222  2
;  1097          1223  2      ! Update the input pointer and length. Remember that char right now contains
;  1098          1224  2      ! the first non-numeric character of the remaining string. Therefore, string_ptr
;  1099          1225  2      ! points one place to the right too far.
;  1100          1226  2      !
;  1101          1227  2      INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] + 1;
;  1102          1228  2      INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
;  1103          1229  2
;  1104          1230  2
;  1105          1231  2      ! Set up the result, check for truncation, and check for a negative number.
;  1106          1232  2      !
;  1107          1233  2      IF .TRUNC_FLAG THEN SIGNAL(DBG$_NUMTRUNC);
;  1108          1234  2      IF .VALUE [0] LSS 0 THEN VALUE [0] = -.VALUE [0];
;  1109          1235  2      .RESULT = .VALUE [0];
;  1110          1236  2      RETURN STS$K_SUCCESS;
;  1111          1237  2
;  1112          1238  1      END;
```

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

                        0D  01  000F3 P.ABK:    .BYTE   1, 13                                   ;
                                DBG$CS_CR=              P.ABK

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

                              003C 00000        .ENTRY  DBG$NSAVE_DECIMAL_INTEGER, Save R2,R3,R4,R5 ; 1126
                    55 00000000G  00  9E 00002  MOVAB   LIB$SIGNAL, R5                          ;
                    5E            0C  C2 00009  SUBL2   #12, SP                               ;
                                  01  DD 0000C  PUSHL   #1                                     ; 1171
                       00000000'  EF  9F 0000E  PUSHAB  DBG$CS_CR                              ;
                    52        04  AC  D0 00014  MOVL    INPUT_DESC, R2                        ;
                                  52  DD 00018  PUSHL   R2                                     ;
                FE70    CF        03  FB 0001A  CALLS   #3, DBG$NMATCH                        ;
                    09            50  E9 0001F  BLBC    R0, 1$                                ;
                       000280D0   8F  DD 00022  PUSHL   #164048                               ; 1173
                    65            01  FB 00028  CALLS   #1, LIB$SIGNAL                        ;
                    54        04  A2  D0 0002B 1$:     MOVL    4(R2), STRING_PTR             ; 1178
                                  62  B5 0002F 2$:     TSTW    (R2)                          ; 1179
                                  0F  13 00031  BEQL    3$                                    ;
                    6E            84  9A 00033  MOVZBL  (STRING_PTR)+, CHAR                   ; 1181
                                  62  B7 00036  DECW    (R2)                                  ; 1182
                    20            6E  D1 00038  CMPL    CHAR, #32                             ; 1183
                                  F2  13 0003B  BEQL    2$                                    ;
                    09            6E  D1 0003D  CMPL    CHAR, #9                              ;
                                  ED  13 00040  BEQL    2$                                    ;
                    30            6E  D1 00042 3$:      CMPL    CHAR, #48                     ; 1189
                                  05  19 00045  BLSS    4$                                    ;
                    39            6E  D1 00047  CMPL    CHAR, #57                             ;
                                  0C  15 0004A  BLEQ    5$                                    ;
                          04      A2      FF    A4  9E 0004C 4$:      MOVAB   -1(R4), 4(R2)   ; 1192
                                                52  DD 00051  PUSHL   R2                      ; 1193
                       0000V      CF            01  FB 00053  CALLS   #1, DBG$SYNTAX_ERROR    ;
                                                04  AE  7C 00058 5$:      CLRQ    VALUE       ; 1202
                                                53  D4 0005B  CLRL    TRUNC_FLAG            ; 1204
                    30                          6E  D1 0005D 6$:      CMPL    CHAR, #48       ; 1205
                                                1E  19 00060  BLSS    8$                    ;
                    39                          6E  D1 00062  CMPL    CHAR, #57             ;
                                                19  14 00065  BGTR    8$                    ;
                    6E                          30  C2 00067  SUBL2   #48, CHAR             ; 1207
    04  AE              6E                      04  AE  7A 0006A  EMUL    VALUE, #10, CHAR, VALUE ; 1208
                                                08  AE  D5 00071  TSTL    VALUE+4           ; 1209
                                                03  13 00074  BEQL    7$                    ;
                    53                          01  D0 00076  MOVL    #1, TRUNC_FLAG        ;
                                                62  B7 00079 7$:      DECW    (R2)          ; 1210
                    6E                          84  9A 0007B  MOVZBL  (STRING_PTR)+, CHAR   ; 1211
                                                DD  11 0007E  BRB     6$                    ; 1205
                          00000041      8F                  6E  D1 00080 8$:      CMPL    CHAR, #65 ; 1218
                                                18  19 00087  BLSS    9$                    ;
                          0000005A      8F                  6E  D1 00089  CMPL    CHAR, #90     ;
                                                0F  14 00090  BGTR    9$                    ;
                                                5E  DD 00092  PUSHL   SP                    ; 1220
```

```
                                    01  DD 00094          PUSHL   #1                                          ;
                                    02  DD 00096          PUSHL   #2                                          ;
                          00028AAA  8F  DD 00098          PUSHL   #166570                                     ;
                         65         04  FB 0009E          CALLS   #4, LIB$SIGNAL                              ;
                                    62  B6 000A1 9$:      INCW    (R2)                                        ; 1227
                   04    A2    FF   A4  9E 000A3          MOVAB   -1(R4), 4(R2)                               ; 1228
                         09         53  E9 000A8          BLBC    TRUNC_FLAG, 10$                             ; 1233
                          00028043  8F  DD 000AB          PUSHL   #163907                                     ;
                         65         01  FB 000B1          CALLS   #1, LIB$SIGNAL                              ;
                               04   AE  D5 000B4 10$:      TSTL    VALUE                                       ; 1234
                                    05  18 000B7          BGEQ    11$                                         ;
                   04    AE    04   AE  CE 000B9          MNEGL   VALUE, VALUE                                ;
                   08    BC    04   AE  D0 000BE 11$:      MOVL    VALUE, @RESULT                              ; 1235
                         50         01  D0 000C3          MOVL    #1, R0                                      ; 1236
                                    04  04 000C6          RET                                                 ; 1238

; Routine Size:  199 bytes,    Routine Base:  DBG$CODE + 06A0
```

```
;1114     1239  1 GLOBAL ROUTINE DBG$NSAVE_INTEGER(INPUT_DESC, RESULT) =
;1115     1240  1 !
;1116     1241  1 ! FUNCTION
;1117     1242  1 !       Extracts a numeric string from the command input string and converts it
;1118     1243  1 !       to a binary longword integer.  The default radix is assumed for the
;1119     1244  1 !       input.
;1120     1245  1 !
;1121     1246  1 ! INPUTS
;1122     1247  1 !       INPUT_DESC - The address of a standard ASCII string descriptor
;1123     1248  1 !                    pointing to the input command line to be scanned.
;1124     1249  1 !
;1125     1250  1 !       RESULT  - The address of a longword to receive the result, meaning
;1126     1251  1 !                 the scanned integer value in internal format.
;1127     1252  1 !
;1128     1253  1 ! OUTPUTS
;1129     1254  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;1130     1255  1 !                    first character after the scanned integer constant.
;1131     1256  1 !
;1132     1257  1 !       RESULT  - The scanned integer value (in internal format) is returned
;1133     1258  1 !                 the RESULT location.
;1134     1259  1 !
;1135     1260  1 !       The value STS$K_SUCCESS is always returned as the routine value.
;1136     1261  1 !
;1137     1262  1
;1138     1263  2     BEGIN
;1139     1264  2
;1140     1265  2     BUILTIN
;1141     1266  2         EMUL;                           ! Multiply and add two longwords
;1142     1267  2                                         !       to produce quadword result.
;1143     1268  2
;1144     1269  2     MAP
;1145     1270  2         INPUT_DESC : REF DBG$STG_DESC;  ! Input string descriptor
;1146     1271  2
;1147     1272  2     BIND
;1148     1273  2         DBG$CS_CR = UPLIT BYTE(1, DBG$K_CAR_RETURN);
;1149     1274  2
;1150     1275  2     LOCAL
;1151     1276  2         VALUE: VECTOR[2],               ! Result quadword
;1152     1277  2         STRING_PTR,                     ! Pointer to input string
;1153     1278  2         TRUNC_FLAG,                     ! Indicates truncation
;1154     1279  2         CHAR;                           ! Holds characters
;1155     1280  2
;1156     1281  2
;1157     1282  2
;1158     1283  2     ! Check for null input
;1159     1284  2     !
;1160     1285  2     IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
;1161     1286  2     THEN
;1162     1287  2         SIGNAL(DBG$_NEEDMORE);
;1163     1288  2
;1164     1289  2
;1165     1290  2     ! Delete leading white space
;1166     1291  2     !
;1167     1292  2     STRING_PTR = .INPUT_DESC [DSC$A_POINTER];
;1168     1293  2     WHILE .INPUT_DESC [DSC$W_LENGTH] GTR 0 DO
;1169     1294  3         BEGIN
;1170     1295  3         CHAR = CH$RCHAR_A (STRING_PTR);
```

```
;  1171          1296  3          INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
;  1172          1297  3          IF (.CHAR NEQ DBG$K_BLANK) AND (.CHAR NEQ DBG$K_TAB) THEN EXITLOOP;
;  1173          1298  2          END;
;  1174          1299  2
;  1175          1300  2      ! Case on the current radix setting.  We will believe this number to
;  1176          1301  2      ! be in that radix.
;  1177          1302  2      !
;  1178          1303  2      SELECTONE .DBG$GB_RADIX[DBG$B_RADIX_INPUT] OF
;  1179          1304  2          SET
;  1180          1305  2
;  1181          1306  2          [DBG$K_BINARY] :
;  1182          1307  3              BEGIN
;  1183          1308  3
;  1184          1309  3              ! Check for a binary character. If we don't have one - syntax error.
;  1185          1310  3              !
;  1186          1311  4              IF (.CHAR LSS '0') OR (.CHAR GTR '1')
;  1187          1312  3              THEN
;  1188          1313  4                  BEGIN
;  1189          1314  4                  INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
;  1190          1315  4                  DBG$SYNTAX_ERROR(.INPUT_DESC);
;  1191          1316  3              END;
;  1192          1317  3
4 ;  1193          1318  3              ! Now continue to accept binary chars and convert them.  Loop
;  1194          1319  3              ! until we find a non-numeric character.  In the loop, append
9 ;  1195          1320  3              ! each new digit to the number being accumulated, check for
;  1196          1321  3              ! overflow, and loop for the next character.
;  1197          1322  3              !
2 ;  1198          1323  3              VALUE [0] = 0;
3 ;  1199          1324  3              VALUE [1] = 0;
4 ;  1200          1325  3              TRUNC_FLAG = FALSE;
5 ;  1201          1326  3              WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '1') DO
1 ;  1202          1327  4              BEGIN
;  1203          1328  4                  CHAR = .CHAR - '0';
;  1204          1329  4                  EMUL (VALUE [0], %REF (2), CHAR, VALUE);
3 ;  1205          1330  4                  IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
4 ;  1206          1331  4                  INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
1 ;  1207          1332  4                  CHAR = CH$RCHAR_A (STRING_PTR);
1 ;  1208          1333  3              END;
2 ;  1209          1334  3
3 ;  1210          1335  3              ! If the next character is an alphabetic character, signal a
;  1211          1336  3              ! syntax error.  This can happen if the user tries to enter a
;  1212          1337  3              ! hex number, for example.
;  1213          1338  3              !
0 ;  1214          1339  3              IF (.CHAR GEQ 'A') AND (.CHAR LEQ 'Z') OR
;  1215          1340  4                 (.CHAR GEQ '2') AND (.CHAR LEQ '9')
1 ;  1216          1341  3              THEN
2 ;  1217          1342  3                  SIGNAL(DBG$_INVDIGBIN, 2, 1, CHAR);
;  1218          1343  3
;  1219          1344  2              END;
;  1220          1345  2
;  1221          1346  2          [DBG$K_OCTAL] :
;  1222          1347  3              BEGIN
8 ;  1223          1348  3
;  1224          1349  3              ! Check for a octal character. If we don't have one - syntax error.
;  1225          1350  3              !
9 ;  1226          1351  4              IF (.CHAR LSS '0') OR (.CHAR GTR '7')
7 ;  1227          1352  3              THEN
```

```
;  1228          1353  4              BEGIN
;  1229          1354  4              INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
;  1230          1355  4              DBG$SYNTAX_ERROR(.INPUT_DESC);
;  1231          1356  3          END;
;  1232          1357  3
;  1233          1358  3          ! Now continue to accept numeric chars and convert them.  Loop
;  1234          1359  3          ! until we find a non-numeric character.  In the loop, append
;  1235          1360  3          ! each new digit to the number being accumulated, check for
;  1236          1361  3          ! overflow, and loop for the next character.
;  1237          1362  3          !
;  1238          1363  3          VALUE [0] = 0;
;  1239          1364  3          VALUE [1] = 0;
;  1240          1365  3          TRUNC_FLAG = FALSE;
;  1241          1366  3          WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '7') DO
;  1242          1367  4          BEGIN
;  1243          1368  4              CHAR = .CHAR - '0';
;  1244          1369  4              EMUL (VALUE [0], %REF (8), CHAR, VALUE);
;  1245          1370  4              IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
;  1246          1371  4              INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
;  1247          1372  4              CHAR = CH$RCHAR_A (STRING_PTR);
;  1248          1373  3          END;
;  1249          1374  3
;  1250          1375  3          ! If the next character is an alphabetic character, signal a
;  1251          1376  3          ! syntax error.  This can happen if the user tries to enter a
;  1252          1377  3          ! hex number, for example.
;  1253          1378  3          !
;  1254          1379  3          IF (.CHAR GEQ 'A') AND (.CHAR LEQ 'Z') OR
;  1255          1380  4             (.CHAR GEQ '8') AND (.CHAR LEQ '9')
;  1256          1381  3          THEN
;  1257          1382  3              SIGNAL(DBG$_INVDIGOCT, 2, 1, CHAR);
;  1258          1383  3
;  1259          1384  2          END;
;  1260          1385  2
;  1261          1386  2      [DBG$K_DECIMAL] :
;  1262          1387  3          BEGIN
;  1263          1388  3
;  1264          1389  3          ! Check for a decimal character. If we don't have one - syntax error.
;  1265          1390  3          !
;  1266          1391  4          IF (.CHAR LSS '0') OR (.CHAR GTR '9')
;  1267          1392  3          THEN
;  1268          1393  4              BEGIN
;  1269          1394  4              INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
;  1270          1395  4              DBG$SYNTAX_ERROR(.INPUT_DESC);
;  1271          1396  3          END;
;  1272          1397  3
;  1273          1398  3          ! Now continue to accept numeric chars and convert them.  Loop
;  1274          1399  3          ! until we find a non-numeric character.  In the loop, append
;  1275          1400  3          ! each new digit to the number being accumulated, check for
;  1276          1401  3          ! overflow, and loop for the next character.
;  1277          1402  3          !
;  1278          1403  3          VALUE [0] = 0;
;  1279          1404  3          VALUE [1] = 0;
;  1280          1405  3          TRUNC_FLAG = FALSE;
;  1281          1406  3          WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '9') DO
;  1282          1407  4          BEGIN
;  1283          1408  4              CHAR = .CHAR - '0';
;  1284          1409  4              EMUL (VALUE [0], %REF (10), CHAR, VALUE);
```

```
;  1285          1410  4                IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
;  1286          1411  4                INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
;  1287          1412  4                CHAR = CH$RCHAR_A (STRING_PTR);
;  1288          1413  3            END;
;  1289          1414  3
;  1290          1415  3            ! If the next character is an alphabetic character, signal a
;  1291          1416  3            ! syntax error.  This can happen if the user tries to enter a
;  1292          1417  3            ! hex number, for example.
;  1293          1418  3            !
;  1294          1419  4            IF (.CHAR GEQ 'A') AND (.CHAR LEQ 'Z')
;  1295          1420  3            THEN
;  1296          1421  3                SIGNAL(DBG$_INVDIGDEC, 2, 1, CHAR);
;  1297          1422  3
;  1298          1423  2            END;
;  1299          1424  2
;  1300          1425  2        [DBG$K_HEX] :
;  1301          1426  3            BEGIN
;  1302          1427  3
;  1303          1428  3            ! Check for a hexidecimal character. If we don't have one - syntax error.
;  1304          1429  3            !
;  1305          1430  3            IF ((.CHAR LSS '0') OR (.CHAR GTR '9')) AND
;  1306          1431  4               ((.CHAR LSS 'A') OR (.CHAR GTR 'F'))
;  1307          1432  3            THEN
;  1308          1433  4                BEGIN
;  1309          1434  4                INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
;  1310          1435  4                DBG$SYNTAX_ERROR(.INPUT_DESC);
;  1311          1436  3            END;
;  1312          1437  3
;  1313          1438  3            ! Now continue to accept hexidecimal chars and convert them.
;  1314          1439  3            ! Loop until we find a non-numeric character.  In the loop,
;  1315          1440  3            ! append each new digit to the number being accumulated, check
;  1316          1441  3            ! for overflow, and loop for the next character.
;  1317          1442  3            !
;  1318          1443  3            VALUE [0] = 0;
;  1319          1444  3            VALUE [1] = 0;
;  1320          1445  3            TRUNC_FLAG = FALSE;
;  1321          1446  3            WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '9') OR
;  1322          1447  3                  (.CHAR GEQ 'A') AND (.CHAR LEQ 'F') DO
;  1323          1448  4            BEGIN
;  1324          1449  5                IF (.CHAR GEQ '0') AND (.CHAR LEQ '9')
;  1325          1450  4                THEN
;  1326          1451  4                    CHAR = .CHAR - '0'
;  1327          1452  4                ELSE
;  1328          1453  4                    CHAR = .CHAR - 55;
;  1329          1454  4                EMUL (VALUE [0], %REF (16), CHAR, VALUE);
;  1330          1455  4                IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
;  1331          1456  4                INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
;  1332          1457  4                CHAR = CH$RCHAR_A (STRING_PTR);
;  1333          1458  3            END;
;  1334          1459  3
;  1335          1460  3            ! If the next character is an alphabetic character, signal a
;  1336          1461  3            ! syntax error.
;  1337          1462  3            !
;  1338          1463  4            IF (.CHAR GEQ 'G') AND (.CHAR LEQ 'Z')
;  1339          1464  3            THEN
;  1340          1465  3                SIGNAL(DBG$_INVDIGHEX, 2, 1, CHAR);
;  1341          1466  3
```

```
;  1342          1467  2              END;
;  1343          1468  2
;  1344          1469  2          [OTHERWISE] :
;  1345          1470  3              BEGIN
;  1346          1471  3              $DBG_ERROR('DBGNPARSE\DBG$NSAVE_INTEGER, DBG$GB_RADIX is invalid');
;  1347          1472  2              END;
;  1348          1473  2
;  1349          1474  2          TES;
;  1350          1475  2
;  1351          1476  2      ! Update the input pointer and length. Remember that char right now contains
;  1352          1477  2      ! the first non-numeric character of the remaining string. Therefore, string_ptr
;  1353          1478  2      ! points one place to the right too far.
;  1354          1479  2      !
;  1355          1480  2      INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] + 1;
;  1356          1481  2      INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
;  1357          1482  2
;  1358          1483  2
;  1359          1484  2      ! Set up the result, check for truncation, and check for a negative number.
;  1360          1485  2      !
;  1361          1486  2      IF .TRUNC_FLAG THEN SIGNAL(DBG$_NUMTRUNC);
;  1362          1487  2      IF .VALUE [0] LSS 0 THEN VALUE [0] = -.VALUE [0];
;  1363          1488  2      .RESULT = .VALUE [0];
;  1364          1489  2      RETURN STS$K_SUCCESS;
;  1365          1490  2
;  1366          1491  1      END;


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

                                                 0D  01  000F5 P.ABL:  .BYTE   1, 13
24  47  42  44  5C  45  53  52  41  50  4E  47  42  44  34  000F7 P.ABM:  .ASCII  \4DBGNPARSE\<92>\DBG$NSAVE_INTEGER, DBG$\
20  2C  52  45  47  45  54  4E  49  5F  45  56  41  53  4E  00106
                                        24  47  42  44  00115
76  6E  69  20  73  69  20  58  49  44  41  52  5F  42  47  00119         .ASCII  \GB_RADIX is invalid\
                                        64  69  6C  61  00128

                                                DBG$CS_CR=          P.ABL

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

                                             00FC 00000         .ENTRY  DBG$NSAVE_INTEGER, Save R2,R3,R4,R5,R6,R7   ;  1239
                                57      0000V  CF   9E 00002    MOVAB   DBG$SYNTAX_ERROR, R7
                                56 00000000G   00   9E 00007    MOVAB   LIB$SIGNAL, R6
                                5E             0C   C2 0000E    SUBL2   #12, SP
                                               01   DD 00011    PUSHL   #1                                          ;  1285
                                   00000000'   EF   9F 00013    PUSHAB  DBG$CS_CR
                                52        04   AC   D0 00019    MOVL    INPUT_DESC, R2
                                               52   DD 0001D    PUSHL   R2
                             FDA4  CF          03   FB 0001F    CALLS   #3, DBG$NMATCH
                                   09          50   E9 00024    BLBC    R0, 1$
                                   000280D0    8F   DD 00027    PUSHL   #164048                                     ;  1287
                                66             01   FB 0002D    CALLS   #1, LIB$SIGNAL
                                53        04   A2   9E 00030 1$:  MOVAB   4(R2), R3                               ;  1292
                                54             63   D0 00034    MOVL    (R3), STRING_PTR
                                               62   B5 00037 2$:  TSTW    (R2)                                     ;  1293
```

```
                                 0F  13 00039          BEQL    3$                                          ;
                              6E 84  9A 0003B          MOVZBL  (STRING_PTR)+, CHAR                         ; 1295
                                 62  B7 0003E          DECW    (R2)                                        ; 1296
                              20 6E  D1 00040          CMPL    CHAR, #32                                   ; 1297
                                 F2  13 00043          BEQL    2$                                          ;
                              09 6E  D1 00045          CMPL    CHAR, #9                                    ;
                                 ED  13 00048          BEQL    2$                                          ;
                           50 00000000G  00  9A 0004A 3$: MOVZBL  DBG$GB_RADIX, R0                         ; 1303
                              02 50  91 00051          CMPB    R0, #2                                      ; 1306
                                 65  12 00054          BNEQ    11$                                         ;
                              30 6E  D1 00056          CMPL    CHAR, #48                                   ; 1311
                                 05  19 00059          BLSS    4$                                          ;
                              31 6E  D1 0005B          CMPL    CHAR, #49                                   ;
                                 09  15 0005E          BLEQ    5$                                          ;
                           63    FF  A4  9E 00060 4$:  MOVAB   -1(R4), (R3)                               ; 1314
                                 52  DD 00064          PUSHL   R2                                          ; 1315
                              67 01  FB 00066          CALLS   #1, DBG$SYNTAX_ERROR                        ;
                                 04  AE  7C 00069 5$:  CLRQ    VALUE                                       ; 1323
                                 55  D4 0006C          CLRL    TRUNC_FLAG                                  ; 1325
                              30 6E  D1 0006E 6$:  CMPL    CHAR, #48                                   ; 1326
                                 1E  19 00071          BLSS    8$                                          ;
                              31 6E  D1 00073          CMPL    CHAR, #49                                   ;
                                 19  14 00076          BGTR    8$                                          ;
                              6E 30  C2 00078          SUBL2   #48, CHAR                                   ; 1328
04  AE                    6E  02 04  AE  7A 0007B      EMUL    VALUE, #2, CHAR, VALUE                      ; 1329
                                 08  AE  D5 00082      TSTL    VALUE+4                                     ; 1330
                                 03  13 00085          BEQL    7$                                          ;
                              55 01  D0 00087          MOVL    #1, TRUNC_FLAG                              ;
                                 62  B7 0008A 7$:  DECW    (R2)                                        ; 1331
                              6E 84  9A 0008C          MOVZBL  (STRING_PTR)+, CHAR                         ; 1332
                                 DD  11 0008F          BRB     6$                                          ; 1326
00000041                      8F 6E  D1 00091 8$:  CMPL    CHAR, #65                                   ; 1339
                                 09  19 00098          BLSS    9$                                          ;
0000005A                      8F 6E  D1 0009A          CMPL    CHAR, #90                                   ;
                                 0A  15 000A1          BLEQ    10$                                         ;
                              32 6E  D1 000A3 9$:  CMPL    CHAR, #50                                   ; 1340
                                 68  19 000A6          BLSS    18$                                         ;
                              39 6E  D1 000A8          CMPL    CHAR, #57                                   ;
                                 68  14 000AB          BGTR    19$                                         ;
                                 5E  DD 000AD 10$:  PUSHL   SP                                          ; 1342
                                 01  DD 000AF          PUSHL   #1                                          ;
                                 02  DD 000B1          PUSHL   #2                                          ;
                           00028AB2  8F  DD 000B3          PUSHL   #166578                                     ;
                                 68  11 000B9          BRB     21$                                         ;
                              08 50  91 000BB 11$:  CMPB    R0, #8                                      ; 1346
                                 65  12 000BE          BNEQ    22$                                         ;
                              30 6E  D1 000C0          CMPL    CHAR, #48                                   ; 1351
                                 05  19 000C3          BLSS    12$                                         ;
                              37 6E  D1 000C5          CMPL    CHAR, #55                                   ;
                                 09  15 000C8          BLEQ    13$                                         ;
                           63    FF  A4  9E 000CA 12$:  MOVAB   -1(R4), (R3)                               ; 1354
                                 52  DD 000CE          PUSHL   R2                                          ; 1355
                              67 01  FB 000D0          CALLS   #1, DBG$SYNTAX_ERROR                        ;
                                 04  AE  7C 000D3 13$:  CLRQ    VALUE                                       ; 1363
                                 55  D4 000D6          CLRL    TRUNC_FLAG                                  ; 1365
                              30 6E  D1 000D8 14$:  CMPL    CHAR, #48                                   ; 1366
                                 1E  19 000DB          BLSS    16$                                         ;
```

```
                                                37              6E  D1 000DD          CMPL    CHAR, #55                                       ;
                                                                19  14 000E0          BGTR    16$                                             ;
                                                6E              30  C2 000E2          SUBL2   #48, CHAR                                       ; 1368
    04  AE              6E                      08      04      AE  7A 000E5          EMUL    VALUE, #8, CHAR, VALUE                          ; 1369
                                                        08      AE  D5 000EC          TSTL    VALUE+4                                         ; 1370
                                                                03  13 000EF          BEQL    15$                                             ;
                                                55              01  D0 000F1          MOVL    #1, TRUNC_FLAG                                  ;
                                                                62  B7 000F4 15$:     DECW    (R2)                                            ; 1371
                                                6E              84  9A 000F6          MOVZBL  (STRING_PTR)+, CHAR                             ; 1372
                                                                DD  11 000F9          BRB     14$                                             ; 1366
                                    00000041    8F              6E  D1 000FB 16$:     CMPL    CHAR, #65                                       ; 1379
                                                                09  19 00102          BLSS    17$                                             ;
                                    0000005A    8F              6E  D1 00104          CMPL    CHAR, #90                                       ;
                                                                0A  15 0010B          BLEQ    20$                                             ;
                                                38              6E  D1 0010D 17$:     CMPL    CHAR, #56                                       ; 1380
                                                                5A  19 00110 18$:     BLSS    28$                                             ;
                                                39              6E  D1 00112          CMPL    CHAR, #57                                       ;
                                                                61  14 00115 19$:     BGTR    31$                                             ;
                                                                5E  DD 00117 20$:     PUSHL   SP                                              ; 1382
                                                                01  DD 00119          PUSHL   #1                                              ;
                                                                02  DD 0011B          PUSHL   #2                                              ;
                                            00028AC2            8F  DD 0011D          PUSHL   #166594                                         ;
                                                                61  11 00123 21$:     BRB     32$                                             ;
                                                0A              50  91 00125 22$:     CMPB    R0, #10                                         ; 1386
                                                                5F  12 00128          BNEQ    33$                                             ;
                                                30              6E  D1 0012A          CMPL    CHAR, #48                                       ; 1391
                                                                05  19 0012D          BLSS    23$                                             ;
                                                39              6E  D1 0012F          CMPL    CHAR, #57                                       ;
                                                                09  15 00132          BLEQ    24$                                             ;
                                                63      FF      A4  9E 00134 23$:     MOVAB   -1(R4), (R3)                                    ; 1394
                                                                52  DD 00138          PUSHL   R2                                              ; 1395
                                                67              01  FB 0013A          CALLS   #1, DBG$SYNTAX_ERROR                            ;
                                                        04      AE  7C 0013D 24$:     CLRQ    VALUE                                           ; 1403
                                                                55  D4 00140          CLRL    TRUNC_FLAG                                      ; 1405
                                                30              6E  D1 00142 25$:     CMPL    CHAR, #48                                       ; 1406
                                                                1E  19 00145          BLSS    27$                                             ;
                                                39              6E  D1 00147          CMPL    CHAR, #57                                       ;
                                                                19  14 0014A          BGTR    27$                                             ;
                                                6E              30  C2 0014C          SUBL2   #48, CHAR                                       ; 1408
    04  AE              6E                      0A      04      AE  7A 0014F          EMUL    VALUE, #10, CHAR, VALUE                         ; 1409
                                                        08      AE  D5 00156          TSTL    VALUE+4                                         ; 1410
                                                                03  13 00159          BEQL    26$                                             ;
                                                55              01  D0 0015B          MOVL    #1, TRUNC_FLAG                                  ;
                                                                62  B7 0015E 26$:     DECW    (R2)                                            ; 1411
                                                6E              84  9A 00160          MOVZBL  (STRING_PTR)+, CHAR                             ; 1412
                                                                DD  11 00163          BRB     25$                                             ; 1406
                                    00000041    8F              6E  D1 00165 27$:     CMPL    CHAR, #65                                       ; 1419
                                                                03  18 0016C 28$:     BGEQ    30$                                             ;
                                                            00C7    31 0016E 29$:     BRW     47$                                             ;
                                    0000005A    8F              6E  D1 00171 30$:     CMPL    CHAR, #90                                       ;
                                                                F4  14 00178 31$:     BGTR    29$                                             ;
                                                                5E  DD 0017A          PUSHL   SP                                              ; 1421
                                                                01  DD 0017C          PUSHL   #1                                              ;
                                                                02  DD 0017E          PUSHL   #2                                              ;
                                            00028AAA            8F  DD 00180          PUSHL   #166570                                         ;
                                                            0099    31 00186 32$:     BRW     45$                                             ;
                                                10              50  91 00189 33$:     CMPB    R0, #16                                         ; 1425
```

```
                                    03  13 0018C          BEQL    34$                                                     ;
                                  0096  31 0018E          BRW     46$                                                     ;
                            30     6E  D1 00191 34$:      CMPL    CHAR, #48                                               ; 1430
                                  05  19 00194          BLSS    35$                                                     ;
                            39     6E  D1 00196          CMPL    CHAR, #57                                               ;
                                  1B  15 00199          BLEQ    37$                                                     ;
                  00000041  8F     6E  D1 0019B 35$:      CMPL    CHAR, #65                                               ; 1431
                                  09  19 001A2          BLSS    36$                                                     ;
                  00000046  8F     6E  D1 001A4          CMPL    CHAR, #70                                               ;
                                  09  15 001AB          BLEQ    37$                                                     ;
                            63  FF  A4  9E 001AD 36$:      MOVAB   -1(R4), (R3)                                            ; 1434
                                  52  DD 001B1          PUSHL   R2                                                      ; 1435
                            67     01  FB 001B3          CALLS   #1, DBG$SYNTAX_ERROR                                    ;
                                04  AE  7C 001B6 37$:      CLRQ    VALUE                                                   ; 1443
                                  55  D4 001B9          CLRL    TRUNC_FLAG                                              ; 1445
                            50     6E  D0 001BB 38$:      MOVL    CHAR, R0                                                ; 1446
                                  51  D4 001BE          CLRL    R1                                                      ;
                            30     50  D1 001C0          CMPL    R0, #48                                                 ;
                                  07  19 001C3          BLSS    39$                                                     ;
                                  51  D6 001C5          INCL    R1                                                      ;
                            39     50  D1 001C7          CMPL    R0, #57                                                 ;
                                  12  15 001CA          BLEQ    40$                                                     ;
                  00000041  8F     50  D1 001CC 39$:      CMPL    R0, #65                                                 ; 1447
                                  2F  19 001D3          BLSS    44$                                                     ;
                  00000046  8F     50  D1 001D5          CMPL    R0, #70                                                 ;
                                  26  14 001DC          BGTR    44$                                                     ;
                            0A     51  E9 001DE 40$:      BLBC    R1, 41$                                                 ; 1449
                            39     50  D1 001E1          CMPL    R0, #57                                                 ;
                                  05  14 001E4          BGTR    41$                                                     ;
                            6E     30  C2 001E6          SUBL2   #48, CHAR                                               ; 1451
                                  03  11 001E9          BRB     42$                                                     ;
                            6E     37  C2 001EB 41$:      SUBL2   #55, CHAR                                               ; 1453
04  AE            6E          10      04  AE  7A 001EE 42$:      EMUL    VALUE, #16, CHAR, VALUE                         ; 1454
                                    08  AE  D5 001F5          TSTL    VALUE+4                                             ; 1455
                                  03  13 001F8          BEQL    43$                                                     ;
                            55     01  D0 001FA          MOVL    #1, TRUNC_FLAG                                          ;
                                  62  B7 001FD 43$:      DECW    (R2)                                                    ; 1456
                            6E     84  9A 001FF          MOVZBL  (STRING_PTR)+, CHAR                                     ; 1457
                                  B7  11 00202          BRB     38$                                                     ; 1446
                  00000047  8F     6E  D1 00204 44$:      CMPL    CHAR, #71                                               ; 1463
                                  2B  19 0020B          BLSS    47$                                                     ;
                  0000005A  8F     6E  D1 0020D          CMPL    CHAR, #90                                               ;
                                  22  14 00214          BGTR    47$                                                     ;
                                  5E  DD 00216          PUSHL   SP                                                      ; 1465
                                  01  DD 00218          PUSHL   #1                                                      ;
                                  02  DD 0021A          PUSHL   #2                                                      ;
                      00028ABA    8F  DD 0021C          PUSHL   #166586                                                 ;
                            66     04  FB 00222 45$:      CALLS   #4, LIB$SIGNAL                                          ;
                                  11  11 00225          BRB     47$                                                     ; 1303
                      00000000'   EF  9F 00227 46$:      PUSHAB  P.ABM                                                   ; 1471
                                  01  DD 0022D          PUSHL   #1                                                      ;
                      00028362    8F  DD 0022F          PUSHL   #164706                                                 ;
                            66     03  FB 00235          CALLS   #3, LIB$SIGNAL                                          ;
                                  62  B6 00238 47$:      INCW    (R2)                                                    ; 1480
                            63     FF  A4  9E 0023A          MOVAB   -1(R4), (R3)                                            ; 1481
                            09     55  E9 0023E          BLBC    TRUNC_FLAG, 48$                                         ; 1486
                      00028043    8F  DD 00241          PUSHL   #163907                                                 ;
```

```
                                66          01  FB 00247          CALLS   #1, LIB$SIGNAL                        ;
                                        04  AE  D5 0024A 48$:      TSTL    VALUE                               ; 1487
                                            05  18 0024D          BGEQ    49$                                 ;
                            04  AE      04  AE  CE 0024F          MNEGL   VALUE, VALUE                        ;
                            08  BC      04  AE  D0 00254 49$:     MOVL    VALUE, @RESULT                      ; 1488
                                50          01  D0 00259          MOVL    #1, R0                              ; 1489
                                            04  0025C            RET                                         ; 1491

; Routine Size:  605 bytes,    Routine Base:  DBG$CODE + 0777
```

```
;1368     1492  1 GLOBAL ROUTINE DBG$NSAVE_STRING(INPUT_DESC, BUFF_PTR, MESSAGE_VECT) =
;1369     1493  1 !
;1370     1494  1 ! FUNCTION
;1371     1495  1 !       This routine accepts the next string from the input buffer
;1372     1496  1 !       and stores it away in a dynamic buffer. Leading white space is ignored.
;1373     1497  1 !       Trailing white space, comma, or <cr> terminate the string.
;1374     1498  1 !       The saved string is stored as a counted string.
;1375     1499  1 !
;1376     1500  1 ! FORMAL PARAMETERS:
;1377     1501  1 !
;1378     1502  1 !       INPUT_DESC      - The address of a VAX standard ASCII string
;1379     1503  1 !                         descriptor which describes input user command.
;1380     1504  1 !
;1381     1505  1 !       BUFF_PTR        - The address of a longword to contain the beginning
;1382     1506  1 !                         address of the stored counted string.
;1383     1507  1 !
;1384     1508  1 !       MESSAGE_VECT    - The address of a longword to contain the address
;1385     1509  1 !                         of a message argument vector for errors.
;1386     1510  1 !
;1387     1511  1 ! IMPLICIT INPUTS:
;1388     1512  1 !
;1389     1513  1 !       NONE
;1390     1514  1 !
;1391     1515  1 ! IMPLICIT OUTPUTS:
;1392     1516  1 !
;1393     1517  1 !       On success, the stored counted ASCII string.
;1394     1518  1 !
;1395     1519  1 !       On failure, a message argument vector.
;1396     1520  1 !
;1397     1521  1 ! ROUTINE VALUE:
;1398     1522  1 !
;1399     1523  1 !       An unsigned integer longword completicn code
;1400     1524  1 !
;1401     1525  1 ! COMPLETION CODES:
;1402     1526  1 !
;1403     1527  1 !       STS$K_SUCCESS (1)       - Success. String isolated and stored.
;1404     1528  1 !
;1405     1529  1 !       STS$K_SEVERE  (4)       - Failure. No string stored. Message argument
;1406     1530  1 !                                 vector returned.
;1407     1531  1 !
;1408     1532  1 !                                 Note that this routine returns failure if a
;1409     1533  1 !                                 string cannot be found. A NEEDMORE
;1410     1534  1 !                                 message is constructed in such cases.
;1411     1535  1 !
;1412     1536  1 ! SIDE EFFECTS:
;1413     1537  1 !
;1414     1538  1 !       The input buffer is updated to reflect one character beyond the last
;1415     1539  1 !       character accepted.
;1416     1540  1 !
;1417     1541  1
;1418     1542  2     BEGIN
;1419     1543  2
;1420     1544  2     MAP
;1421     1545  2         INPUT_DESC : REF DBG$STG_DESC;  ! Input string descriptor
;1422     1546  2
;1423     1547  2     BIND
;1424     1548  2         DBG$CS_CR = UPLIT BYTE(1, DBG$K_CAR_RETURN);
```

```
;1425     1549  2
;1426     1550  2      LOCAL
;1427     1551  2          STRING: REF VECTOR[,BYTE],      ! Will hold the string
;1428     1552  2          COUNT,                          ! Count of characters in string
;1429     1553  2          POINTER;                        ! Temporary pointer
;1430     1554  2
;1431     1555  2
;1432     1556  2
;1433     1557  2      ! Check for null input
;1434     1558  2      !
;1435     1559  2      IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
;1436     1560  2      THEN
;1437     1561  3          BEGIN
;1438     1562  3          .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE);
;1439     1563  3          RETURN STS$K_SEVERE;
;1440     1564  2          END;
;1441     1565  2
;1442     1566  2
;1443     1567  2      ! Delete leading white space
;1444     1568  2      !
;1445     1569  2      WHILE CH$RCHAR (.INPUT_DESC [DSC$A_POINTER]) EQL ' ' DO
;1446     1570  3          BEGIN
;1447     1571  3          INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.INPUT_DESC [DSC$A_POINTER], 1);
;1448     1572  3          INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
;1449     1573  2          END;
;1450     1574  2
;1451     1575  2
;1452     1576  2      ! Save the start of the string
;1453     1577  2      !
;1454     1578  2      POINTER = .INPUT_DESC [DSC$A_POINTER];
;1455     1579  2
;1456     1580  2
;1457     1581  2      ! Now look for a comma, a blank, or <cr>
;1458     1582  2      !
;1459     1583  2      WHILE CH$RCHAR (.INPUT_DESC [DSC$A_POINTER]) NEQ ' ' AND CH$RCHAR (.INPUT_DESC [DSC$A_POINTER]) NEQ ','
;1460     1584  2          AND CH$RCHAR (.INPUT_DESC [DSC$A_POINTER]) NEQ DBG$K_CAR_RETURN DO
;1461     1585  3          BEGIN
;1462     1586  3          INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.INPUT_DESC [DSC$A_POINTER], 1);
;1463     1587  3          INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
;1464     1588  2          END;
;1465     1589  2
;1466     1590  2
;1467     1591  2      ! Figure out how many characters to save
;1468     1592  2      !
;1469     1593  2      COUNT = CH$DIFF (.INPUT_DESC [DSC$A_POINTER], .POINTER);
;1470     1594  2
;1471     1595  2
;1472     1596  2      ! Allocate the proper size buffer
;1473     1597  2      !
;1474     1598  2      STRING = DBG$GET_TEMPMEM ((.COUNT/%UPVAL) + 1);
;1475     1599  2
;1476     1600  2
;1477     1601  2      ! Copy over the count and the characters.
;1478     1602  2      !
;1479     1603  2      STRING [0] = .COUNT;
;1480     1604  2      CH$MOVE (.COUNT, .POINTER, STRING [1]);
;1481     1605  2
```

```
;  1482          1606  2
;  1483          1607  2     ! Set up the output parameter and return
;  1484          1608  2     !
;  1485          1609  2     .BUFF_PTR = STRING [0];
;  1486          1610  2     RETURN STS$K_SUCCESS;
;  1487          1611  2
;  1488          1612  1     END;


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

                                              0D  01  0012C P.ABN:  .BYTE   1, 13                                   ;

                                                        DBG$CS_CR=          P.ABN


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

                                                 007C 00000         .ENTRY  DBG$NSAVE_STRING, Save R2,R3,R4,R5,R6      ; 1492
                                                 01   DD 00002      PUSHL   #1                                         ; 1559
                                      00000000'  EF   9F 00004      PUSHAB  DBG$CS_CR
                                           52    04   AC  D0 00008  MOVL    INPUT_DESC, R2
                                                 52   DD 0000E      PUSHL   R2
                                          FB56   CF   03  FB 00010  CALLS   #3, DBG$NMATCH
                                                 15   50  E9 00015  BLBC    R0, 1$
                                                 000280D0 8F  DD 00018  PUSHL   #164048                                 ; 1562
                                  00000000G      00   01  FB 0001E  CALLS   #1, DBG$NMAKE_ARG_VECT
                                          0C     BC   50  D0 00025  MOVL    R0, @MESSAGE_VECT
                                                 50   04  D0 00029  MOVL    #4, R0                                     ; 1563
                                                      04  0002C     RET
                                                 50   04   A2  9E 0002D 1$:  MOVAB   4(R2), R0                        ; 1569
                                                 20   00   B0  91 00031 2$:  CMPB    @0(R0), #32
                                                           06  12 00035  BNEQ    3$
                                                           60  D6 00037  INCL    (R0)                                   ; 1571
                                                           62  B7 00039  DECW    (R2)                                   ; 1572
                                                           F4  11 0003B  BRB     2$                                     ; 1569
                                                 53        60  D0 0003D 3$:  MOVL    (R0), POINTER                      ; 1578
                                                 20   00   B0  91 00040 4$:  CMPB    @0(R0), #32                        ; 1583
                                                           12  13 00044  BEQL    5$
                                                 2C   00   B0  91 00046  CMPB    @0(R0), #44
                                                           0C  13 0004A  BEQL    5$
                                                 0D   00   B0  91 0004C  CMPB    @0(R0), #13                        ; 1584
                                                           06  13 00050  BEQL    5$
                                                           60  D6 00052  INCL    (R0)                                   ; 1586
                                                           62  B7 00054  DECW    (R2)                                   ; 1587
                                                           E8  11 00056  BRB     4$                                     ; 1583
                                           52              60  53  C3 00058 5$:  SUBL3   POINTER, (R0), COUNT          ; 1593
                                           50              52  04  C7 0005C  DIVL3   #4, COUNT, R0                     ; 1598
                                                      01   A0  9F 00060  PUSHAB  1(R0)
                                               00000000G  00   01  FB 00063  CALLS   #1, DBG$GET_TEMPMEM
                                                          56   50  D0 0006A  MOVL    R0, STRING
                                                          66   52  90 0006D  MOVB    COUNT, (STRING)                    ; 1603
                                      01   A6             63   52  28 00070  MOVC3   COUNT, (POINTER), 1(STRING)       ; 1604
                                                    08    BC   56  D0 00075  MOVL    STRING, @BUFF_PTR                  ; 1609
                                                          50   01  D0 00079  MOVL    #1, R0                             ; 1610
                                                               04  0007C     RET                                        ; 1612
```

; Routine Size: 125 bytes,   Routine Base: DBG$CODE + 09D4

```
;1490     1613  1 GLOBAL ROUTINE DBG$NGET_DIR_LIST (LEX_STG_DESC, RESULT_PTR, MESSAGE_VECT) =
;1491     1614  1 !
;1492     1615  1 ! FUNCTION
;1493     1616  1 !
;1494     1617  1 !       Parse a directory list. Used by the command
;1495     1618  1 !       SET SOURCE dir-list
;1496     1619  1 !
;1497     1620  1 ! INPUTS
;1498     1621  1 !
;1499     1622  1 !       LEX_STG_DESC    - A string descriptor for the unparsed
;1500     1623  1 !                         directory list.
;1501     1624  1 !       RESULT_PTR      - The result of the parse will be placed at the
;1502     1625  1 !                         address given by result_ptr.
;1503     1626  1 !       MESSAGE_VECT    - Address of a longword to contain the address of
;1504     1627  1 !                         the message argument vector.
;1505     1628  1 !
;1506     1629  1 ! OUTPUTS
;1507     1630  1 !
;1508     1631  1 !       This routine builds a linked list of directory names and
;1509     1632  1 !       leaves a pointer to this list in result_ptr.
;1510     1633  1 !
;1511     1634  1 ! ROUTINE VALUE
;1512     1635  1 !
;1513     1636  1 !       A completion code which is one of either:
;1514     1637  1 !       STS$K_SEVERE (4)        Unsuccessful parse
;1515     1638  1 !       STS$K_SUCCESS (1)       Successful parse
;1516     1639  1 !
;1517     1640  1 ! ALGORITHM
;1518     1641  1 !
;1519     1642  1 !       The directory list will be of the form:
;1520     1643  1 !       [dir1]file1,[dir2]file2,...[dirn]filen
;1521     1644  1 !       Do until end of list is reached:
;1522     1645  1 !           Pick up a single directory name by scanning until
;1523     1646  1 !           a separating comma is reached (commas inside of
;1524     1647  1 !           square brackets are ignored)
;1525     1648  1 !           Allocate space for a node containing the directory name.
;1526     1649  1 !           Copy the directory name into this node, and
;1527     1650  1 !           link the node into the list.
;1528     1651  1 !
;1529     1652  1
;1530     1653  2     BEGIN
;1531     1654  2
;1532     1655  2     MAP
;1533     1656  2         LEX_STG_DESC : REF BLOCK [, BYTE]; ! Input string descriptor
;1534     1657  2
;1535     1658  2     LOCAL
;1536     1659  2         TOT_LENGTH,                     ! Number of chars to be read from
;1537     1660  2                                         !      PARSE_STG_DESC
;1538     1661  2         TOT_CHARS_READ,                 ! Number of chars read from
;1539     1662  2                                         !      PARSE_STG_DESC so far
;1540     1663  2         CHARS_READ,                     ! Number of chars in directory name
;1541     1664  2                                         !      under construction
;1542     1665  2         IN_BRACKETS,                    ! TRUE if we are inside of square
;1543     1666  2                                         !      brackets
;1544     1667  2         CHAR,                           ! Current character
;1545     1668  2         PREV_CHAR_PTR,                  ! ???
;1546     1669  2         FIRST_DIR_FLAG,                 ! TRUE if this is first directory in
```

```
;  1547          1670  2                                    !       list
;  1548          1671  2          PREV_DIRNAME: REF SDSL$ENTRY,    ! Previous directory name
;  1549          1672  2          CURR_DIRNAME: REF SDSL$ENTRY,    ! Current directory name
;  1550          1673  2          FIRST_DIRNAME,                   ! Pointer to first directory name
;  1551          1674  2                                           !       (head of linked list)
;  1552          1675  2          INPUT_PTR,                       ! ???
;  1553          1676  2          OUTPUT_PTR,                      ! ???
;  1554          1677  2          NEW_INPUT_PTR,                   ! ???
;  1555          1678  2          PREV_CHAR,                       ! ???
;  1556          1679  2          QUOTE_FLAG,                      ! TRUE if the directory currently being
;  1557          1680  2                                           !       processed is inside of quotes
;  1558          1681  2          DBLQUOTE_FLAG;                   ! TRUE if the directory currently being
;  1559          1682  2                                           !       processed is inside double quotes
;  1560          1683  2
;  1561          1684  2
;  1562          1685  2
;  1563          1686  2      ! Initialize the various flags and counters we need.
;  1564          1687  2      !
;  1565          1688  2      IN_BRACKETS = FALSE;
;  1566          1689  2      TOT_LENGTH = .LEX_STG_DESC [DSC$W_LENGTH];
;  1567          1690  2      TOT_CHARS_READ = 0;
;  1568          1691  2      QUOTE_FLAG = FALSE;
;  1569          1692  2      DBLQUOTE_FLAG = FALSE;
;  1570          1693  2      CHARS_READ = 0;
;  1571          1694  2
;  1572          1695  2
;  1573          1696  2      ! FIRST_DIR_FLAG is true initially because we are processing the
;  1574          1697  2      ! first directory in the list.
;  1575          1698  2      !
;  1576          1699  2      FIRST_DIR_FLAG = TRUE;
;  1577          1700  2      PREV_DIRNAME = 0;
;  1578          1701  2      WHILE .TOT_CHARS_READ LSS .TOT_LENGTH DO
;  1579          1702  3          BEGIN
;  1580          1703  3          INPUT_PTR = CH$PTR (.LEX_STG_DESC [DSC$A_POINTER]);
;  1581          1704  3          CHAR = CH$RCHAR_A (INPUT_PTR);
;  1582          1705  3
;  1583          1706  3
;  1584          1707  3          ! Skip leading blanks
;  1585          1708  3          !
;  1586          1709  3          WHILE .CHAR EQL DBG$K_BLANK AND .TOT_CHARS_READ LSS .TOT_LENGTH DO
;  1587          1710  4              BEGIN
;  1588          1711  4              CHAR = CH$RCHAR_A (INPUT_PTR);
;  1589          1712  4              TOT_CHARS_READ = .TOT_CHARS_READ + 1;
;  1590          1713  4              LEX_STG_DESC [DSC$A_POINTER] =
;  1591          1714  4                                  CH$PLUS(.LEX_STG_DESC [DSC$A_POINTER], 1);
;  1592          1715  4              LEX_STG_DESC [DSC$W_LENGTH] = .LEX_STG_DESC [DSC$W_LENGTH] - 1;
;  1593          1716  3              END;
;  1594          1717  3
;  1595          1718  3          IF .CHAR EQL DBG$K_QUOTE OR .CHAR EQL DBG$K_DBLQUOTE
;  1596          1719  3          THEN
;  1597          1720  4              BEGIN
;  1598          1721  4              IF .CHAR EQL DBG$K_QUOTE
;  1599          1722  4              THEN
;  1600          1723  4                  QUOTE_FLAG = TRUE
;  1601          1724  4
;  1602          1725  4              ELSE
;  1603          1726  4                  DBLQUOTE_FLAG = TRUE;
```

```
;1604     1727  4
;1605     1728  4
;1606     1729  4          ! Advance the pointer past the quote.
;1607     1730  4          !
;1608     1731  4          LEX_STG_DESC [DSC$A_POINTER] =
;1609     1732  4                              CH$PLUS(.LEX_STG_DESC [DSC$A_POINTER], 1);
;1610     1733  4          LEX_STG_DESC [DSC$W_LENGTH] = .LEX_STG_DESC [DSC$W_LENGTH] - 1;
;1611     1734  4          TOT_CHARS_READ = .TOT_CHARS_READ + 1;
;1612     1735  4          CHAR = CH$RCHAR_A (INPUT_PTR);
;1613     1736  3          END;
;1614     1737  3
;1615     1738  3
;1616     1739  3        ! Pick up next directory name.
;1617     1740  3        !
;1618     1741  3        WHILE .TOT_CHARS_READ LSS .TOT_LENGTH AND
;1619     1742  3              .CHAR NEQ DBG$K_CAR_RETURN AND
;1620     1743  4              (.CHAR NEQU %C';' OR .IN_BRACKETS)
;1621     1744  3        DO
;1622     1745  4            BEGIN
;1623     1746  4          IF .CHAR EQLU %C'[' OR .CHAR EQLU %C'<' THEN IN_BRACKETS = TRUE;
;1624     1747  4          IF .CHAR EQLU %C']' OR .CHAR EQLU %C'>' THEN IN_BRACKETS = FALSE;
;1625     1748  4          IF .CHAR NEQ DBG$K_BLANK
;1626     1749  4          THEN
;1627     1750  4              PREV_CHAR_PTR = CH$PLUS (.INPUT_PTR, -1);
;1628     1751  4
;1629     1752  4          PREV_CHAR = .CHAR;
;1630     1753  4          CHAR = CH$RCHAR_A (INPUT_PTR);
;1631     1754  4
;1632     1755  4
;1633     1756  4          ! If this is a doubled up quote character, undouble them and mark
;1634     1757  4          ! the position.
;1635     1758  4          !
;1636     1759  4          IF .DBLQUOTE_FLAG           AND
;1637     1760  4             .CHAR EQL DBG$K_DBLQUOTE AND
;1638     1761  4             .PREV_CHAR EQL DBG$K_DBLQUOTE
;1639     1762  4          THEN
;1640     1763  4              (.PREV_CHAR_PTR)<0, 8, 0> = 0;
;1641     1764  4
;1642     1765  4          TOT_CHARS_READ = .TOT_CHARS_READ + 1;
;1643     1766  4          CHARS_READ = .CHARS_READ + 1;
;1644     1767  3          END;                        ! End of inner while loop
;1645     1768  3
;1646     1769  3
;1647     1770  3        ! We now have a complete directory name.  Allocate space for it.
;1648     1771  3        !
;1649     1772  3        CURR_DIRNAME = DBG$GET_MEMORY (2 + (1 + .CHARS_READ)/%UPVAL);
;1650     1773  4        IF (.FIRST_DIR_FLAG)
;1651     1774  3        THEN
;1652     1775  4            BEGIN
;1653     1776  4            FIRST_DIR_FLAG = FALSE;
;1654     1777  4            FIRST_DIRNAME = .CURR_DIRNAME;
;1655     1778  3            END;
;1656     1779  3
;1657     1780  3
;1658     1781  3        ! Link in next directory name
;1659     1782  3        !
;1660     1783  4        IF (.PREV_DIRNAME NEQA 0)
```

```
;1661     1784  3          THEN
;1662     1785  3              PREV_DIRNAME [SDSL$L_ENT_FLINK] = .CURR_DIRNAME;
;1663     1786  3
;1664     1787  3
;1665     1788  3          ! Set PREV_DIRNAME for next time around loop.
;1666     1789  3          !
;1667     1790  3          PREV_DIRNAME = .CURR_DIRNAME;
;1668     1791  3
;1669     1792  3
;1670     1793  3          ! Fill in the fields of CURR_DIRNAME.
;1671     1794  3          !
;1672     1795  3          CURR_DIRNAME [SDSL$L_ENT_FLINK] = 0;
;1673     1796  3          CURR_DIRNAME [SDSL$B_ENT_DIRLEN] = 0;
;1674     1797  3          NEW_INPUT_PTR = CH$PTR (.LEX_STG_DESC [DSC$A_POINTER]);
;1675     1798  3          OUTPUT_PTR = CH$PTR (.CURR_DIRNAME + 5);
;1676     1799  3          INCR I FROM 0 TO .CHARS_READ - 1 DO
;1677     1800  4              BEGIN
;1678     1801  4              CHAR = CH$RCHAR_A (NEW_INPUT_PTR);
;1679     1802  4              IF .CHAR NEQ 0
;1680     1803  4              THEN
;1681     1804  5                  BEGIN
;1682     1805  5                  CH$WCHAR_A (.CHAR, OUTPUT_PTR);
;1683     1806  5                  CURR_DIRNAME [SDSL$B_ENT_DIRLEN] =
;1684     1807  5                                  .CURR_DIRNAME [SDSL$B_ENT_DIRLEN] + 1;
;1685     1808  4                  END;
;1686     1809  4
;1687     1810  3              END;
;1688     1811  3
;1689     1812  3
;1690     1813  3          ! Strip off trailing quote.
;1691     1814  3          !
;1692     1815  3          IF .QUOTE_FLAG
;1693     1816  3          THEN
;1694     1817  4              BEGIN
;1695     1818  4              IF CH$RCHAR (.PREV_CHAR_PTR) EQL DBG$K_QUOTE
;1696     1819  4              THEN
;1697     1820  5                  BEGIN
;1698     1821  5                  CURR_DIRNAME [SDSL$B_ENT_DIRLEN] = .CURR_DIRNAME[SDSL$B_ENT_DIRLEN]
;1699     1822  5                                                  - (.INPUT_PTR - .PREV_CHAR_PTR - 1);
;1700     1823  5                  QUOTE_FLAG = FALSE;
;1701     1824  5                  END
;1702     1825  5
;1703     1826  4              END
;1704     1827  4
;1705     1828  4
;1706     1829  4          ! ???
;1707     1830  4          !
;1708     1831  3          ELSE IF .DBLQUOTE_FLAG
;1709     1832  3          THEN
;1710     1833  4              BEGIN
;1711     1834  4              IF CH$RCHAR (.PREV_CHAR_PTR) EQL DBG$K_DBLQUOTE
;1712     1835  4              THEN
;1713     1836  5                  BEGIN
;1714     1837  6                  CURR_DIRNAME [SDSL$B_ENT_DIRLEN] = .CURR_DIRNAME [SDSL$B_ENT_DIRLEN] - (.INPUT_PTR -
;1715     1838  5                  .PREV_CHAR_PTR - 1);
;1716     1839  5                  DBLQUOTE_FLAG = FALSE;
;1717     1840  4                  END;
```

```
;  1718          1841  4
;  1719          1842  3            END;
;  1720          1843  3
;  1721          1844  3
;  1722          1845  3        ! If there are still characters left in the input string, increment
;  1723          1846  3        ! CHARS_READ to take us past the separating comma.
;  1724          1847  3        !
;  1725          1848  3        IF .TOT_CHARS_READ LSS .TOT_LENGTH
;  1726          1849  3        THEN
;  1727          1850  4            BEGIN
;  1728          1851  4            CHARS_READ = .CHARS_READ + 1;
;  1729          1852  4            TOT_CHARS_READ = .TOT_CHARS_READ + 1;
;  1730          1853  3            END;
;  1731          1854  3
;  1732          1855  3        LEX_STG_DESC [DSC$W_LENGTH] = .LEX_STG_DESC [DSC$W_LENGTH] - .CHARS_READ;
;  1733          1856  3        LEX_STG_DESC [DSC$A_POINTER] = CH$PLUS (.LEX_STG_DESC [DSC$A_POINTER], .CHARS_READ);
;  1734          1857  3        CHARS_READ = 0;
;  1735          1858  2        END;                                ! End of outer while loop
;  1736          1859  2
;  1737          1860  2    IF .QUOTE_FLAG OR .DBLQUOTE_FLAG THEN SIGNAL (DBG$_MATQUOMIS);
;  1738          1861  2
;  1739          1862  2
;  1740          1863  2    ! Place pointer to linked list into into the result pointer.
;  1741          1864  2    .RESULT_PTR = .FIRST_DIRNAME;
;  1742          1865  2    RETURN STS$K_SUCCESS
;  1743          1866  2
;  1744          1867  1    END;

                      0FFC 00000          .ENTRY  DBG$NGET_DIR_LIST, Save R2,R3,R4,R5,R6,R7,- ;  1613
                                                  R8,R9,R10,R11                               ;
                   5E   20  C2 00002          SUBL2   #32, SP                                ;
                   52 04  AC  D0 00005          MOVL    LEX_STG_DESC, R2                    ;  1689
                   5A     62  3C 00009          MOVZWL  (R2), TOT_LENGTH                    ;
                          55  D4 0000C          CLRL    TOT_CHARS_READ                      ;  1690
                          59  D4 0000E          CLRL    QUOTE_FLAG                          ;  1691
                          5B  D4 00010          CLRL    DBLQUOTE_FLAG                       ;  1692
                          57  D4 00012          CLRL    CHARS_READ                          ;  1693
                18 AE     01  7D 00014          MOVQ    #1, FIRST_DIR_FLAG                  ;  1699
                      14  AE  D4 00018          CLRL    PREV_DIRNAME                        ;  1700
                   53 04  A2  9E 0001B          MOVAB   4(R2), R3                           ;  1703
                   5A     55  D1 0001F 1$:      CMPL    TOT_CHARS_READ, TOT_LENGTH          ;  1701
                          03  19 00022          BLSS    2$                                  ;
                        013C  31 00024          BRW     25$                                 ;
                   56     63  D0 00027 2$:      MOVL    (R3), INPUT_PTR                     ;  1703
                   54     86  9A 0002A          MOVZBL  (INPUT_PTR)+, CHAR                  ;  1704
                   20     54  D1 0002D 3$:      CMPL    CHAR, #32                           ;  1709
                          10  12 00030          BNEQ    4$                                  ;
                   5A     55  D1 00032          CMPL    TOT_CHARS_READ, TOT_LENGTH          ;
                          0B  18 00035          BGEQ    4$                                  ;
                   54     86  9A 00037          MOVZBL  (INPUT_PTR)+, CHAR                  ;  1711
                          55  D6 0003A          INCL    TOT_CHARS_READ                      ;  1712
                          63  D6 0003C          INCL    (R3)                                ;  1714
                          62  B7 0003E          DECW    (R2)                                ;  1715
```

```
                                 EB  11 00040          BRB     3$                                      ; 1709
                                 51  D4 00042 4$:      CLRL    R1                                      ; 1718
                           27    54  D1 00044          CMPL    CHAR, #39                               ;
                                 04  12 00047          BNEQ    5$                                      ;
                                 51  D6 00049          INCL    R1                                      ;
                                 05  11 0004B          BRB     6$                                      ;
                           22    54  D1 0004D 5$:      CMPL    CHAR, #34                               ;
                                 14  12 00050          BNEQ    9$                                      ;
                           05    51  E9 00052 6$:      BLBC    R1, 7$                                  ; 1721
                           59    01  D0 00055          MOVL    #1, QUOTE_FLAG                          ; 1723
                                 03  11 00058          BRB     8$                                      ;
                           5B    01  D0 0005A 7$:      MOVL    #1, DBLQUOTE_FLAG                       ; 1726
                                 63  D6 0005D 8$:      INCL    (R3)                                    ; 1732
                                 62  B7 0005F          DECW    (R2)                                    ; 1733
                                 55  D6 00061          INCL    TOT_CHARS_READ                          ; 1734
                           54    86  9A 00063          MOVZBL  (INPUT_PTR)+, CHAR                      ; 1735
                           5A    55  D1 00066 9$:      CMPL    TOT_CHARS_READ, TOT_LENGTH              ; 1741
                                 57  18 00069          BGEQ    17$                                     ;
                           0D    54  D1 0006B          CMPL    CHAR, #13                               ; 1742
                                 52  13 0006E          BEQL    17$                                     ;
                           2C    54  D1 00070          CMPL    CHAR, #44                               ; 1743
                                 04  12 00073          BNEQ    10$                                     ;
                           49  1C    AE  E9 00075          BLBC    IN_BRACKETS, 17$                        ;
0000005B                   8F    54  D1 00079 10$:      CMPL    CHAR, #91                               ; 1746
                                 05  13 00080          BEQL    11$                                     ;
                           3C    54  D1 00082          CMPL    CHAR, #60                               ;
                                 04  12 00085          BNEQ    12$                                     ;
                        1C AE    01  D0 00087 11$:      MOVL    #1, IN_BRACKETS                         ;
0000005D                   8F    54  D1 0008B 12$:      CMPL    CHAR, #93                               ; 1747
                                 05  13 00092          BEQL    13$                                     ;
                           3E    54  D1 00094          CMPL    CHAR, #62                               ;
                                 03  12 00097          BNEQ    14$                                     ;
                        1C AE    D4  00099 13$:         CLRL    IN_BRACKETS                             ;
                           20    54  D1 0009C 14$:      CMPL    CHAR, #32                               ; 1748
                                 04  13 0009F          BEQL    15$                                     ;
                           58  FF  A6  9E 000A1          MOVAB   -1(R6), PREV_CHAR_PTR                   ; 1750
                     0C    AE    54  D0 000A5 15$:      MOVL    CHAR, PREV_CHAR                         ; 1752
                           54    86  9A 000A9          MOVZBL  (INPUT_PTR)+, CHAR                      ; 1753
                           0D    5B  E9 000AC          BLBC    DBLQUOTE_FLAG, 16$                      ; 1759
                           22    54  D1 000AF          CMPL    CHAR, #34                               ; 1760
                                 08  12 000B2          BNEQ    16$                                     ;
                           22    0C  AE  D1 000B4          CMPL    PREV_CHAR, #34                          ; 1761
                                 02  12 000B8          BNEQ    16$                                     ;
                                 68  94 000BA          CLRB    (PREV_CHAR_PTR)                         ; 1763
                                 55  D6 000BC 16$:      INCL    TOT_CHARS_READ                          ; 1765
                                 57  D6 000BE          INCL    CHARS_READ                              ; 1766
                                 A4  11 000C0          BRB     9$                                      ; 1741
                           51    01  A7  9E 000C2 17$:      MOVAB   1(R7), R1                               ; 1772
                           51    04  C6  000C6          DIVL2   #4, R1                                  ;
                                 02  A1  9F 000C9          PUSHAB  2(R1)                                   ;
                00000000G  00    01  FB  000CC          CALLS   #1, DBG$GET_MEMORY                      ;
                           07    18  AE  E9 000D3          BLBC    FIRST_DIR_FLAG, 18$                     ; 1773
                                 18  AE  D4 000D7          CLRL    FIRST_DIR_FLAG                          ; 1776
                     04    AE    50  D0 000DA          MOVL    CURR_DIRNAME, FIRST_DIRNAME             ; 1777
                                 14  AE  D5 000DE 18$:      TSTL    PREV_DIRNAME                            ; 1783
                                 04  13 000E1          BEQL    19$                                     ;
                     14    BE    50  D0 000E3          MOVL    CURR_DIRNAME, @PREV_DIRNAME             ; 1785
```

```
                14   AE          50  D0 000E7 19$:   MOVL    CURR_DIRNAME, PREV_DIRNAME                ; 1790
                                 60  D4 000EB        CLRL    (CURR_DIRNAME)                            ; 1795
                          04     A0  94 000ED        CLRB    4(CURR_DIRNAME)                           ; 1796
                08   AE          63  D0 000F0        MOVL    (R3), NEW_INPUT_PTR                       ; 1797
                10   AE      05  A0  9E 000F4        MOVAB   5(R0), OUTPUT_PTR                         ; 1798
                     51          01  CE 000F9        MNEGL   #1, I                                     ; 1799
                                 15  11 000FC        BRB     21$                                       ;
                     54      08  BE  9A 000FE 20$:   MOVZBL  @NEW_INPUT_PTR, CHAR                     ; 1801
                             08  AE  D6 00102        INCL    NEW_INPUT_PTR                             ;
                                 54  D5 00105        TSTL    CHAR                                      ; 1802
                                 0A  13 00107        BEQL    21$                                       ;
                10   BE          54  90 00109        MOVB    CHAR, @OUTPUT_PTR                         ; 1805
                             10  AE  D6 0010D        INCL    OUTPUT_PTR                                ;
                             04  A0  96 00110        INCB    4(CURR_DIRNAME)                           ; 1807
          E7         51          57  F2 00113 21$:   AOBLSS  CHARS_READ, I, 20$                       ; 1799
                     1A          59  E9 00117        BLBC    QUOTE_FLAG, 22$                           ; 1815
                     27          68  91 0011A        CMPB    (PREV_CHAR_PTR), #39                      ; 1818
                                 30  12 0011D        BNEQ    23$                                       ;
          51         56          58  C3 0011F        SUBL3   PREV_CHAR_PTR, INPUT_PTR, R1              ; 1822
                     6E      04  A0  9A 00123        MOVZBL  4(CURR_DIRNAME), (SP)                     ;
          51         6E          51  C3 00127        SUBL3   R1, (SP), R1                              ;
    04    A0         51          01  81 0012B        ADDB3   #1, R1, 4(CURR_DIRNAME)                   ;
                                 59  D4 00130        CLRL    QUOTE_FLAG                                ; 1823
                                 1B  11 00132        BRB     23$                                       ; 1817
                     18          5B  E9 00134 22$:   BLBC    DBLQUOTE_FLAG, 23$                        ; 1831
                     22          68  91 00137        CMPB    (PREV_CHAR_PTR), #34                      ; 1834
                                 13  12 0013A        BNEQ    23$                                       ;
          51         56          58  C3 0013C        SUBL3   PREV_CHAR_PTR, INPUT_PTR, R1              ; 1838
                     6E      04  A0  9A 00140        MOVZBL  4(CURR_DIRNAME), (SP)                     ; 1837
          51         6E          51  C3 00144        SUBL3   R1, (SP), R1                              ;
    04    A0         51          01  81 00148        ADDB3   #1, R1, 4(CURR_DIRNAME)                   ;
                                 5B  D4 0014D        CLRL    DBLQUOTE_FLAG                             ; 1839
                     5A          55  D1 0014F 23$:   CMPL    TOT_CHARS_READ, TOT_LENGTH                ; 1848
                                 04  18 00152        BGEQ    24$                                       ;
                                 57  D6 00154        INCL    CHARS_READ                                ; 1851
                                 55  D6 00156        INCL    TOT_CHARS_READ                            ; 1852
                     62          57  A2 00158 24$:   SUBW2   CHARS_READ, (R2)                         ; 1855
                     63          57  C0 0015B        ADDL2   CHARS_READ, (R3)                          ; 1856
                                 57  D4 0015E        CLRL    CHARS_READ                                ; 1857
                               FEBC  31 00160        BRW     1$                                        ; 1701
                     03          59  E8 00163 25$:   BLBS    QUOTE_FLAG, 26$                           ; 1860
                     0D          5B  E9 00166        BLBC    DBLQUOTE_FLAG, 27$                        ;
                       00028E30  8F  DD 00169 26$:   PUSHL   #167472                                   ;
          00000000G  00          01  FB 0016F        CALLS   #1, LIB$SIGNAL                            ;
                08   BC      04  AE  D0 00176 27$:   MOVL    FIRST_DIRNAME, @RESULT_PTR                ; 1864
                     50          01  D0 0017B        MOVL    #1, R0                                    ; 1865
                                     04 0017E        RET                                               ; 1867

; Routine Size:  383 bytes,    Routine Base:  DBG$CODE + 0A51
```

```
;  1746          1868  1 GLOBAL ROUTINE DBG$EXPAND_DEFINE_NAME(INPUT_DESC, KIND, RESULT_ADDR) =
;  1747          1869  1 !
;  1748          1870  1 ! FUNCTION
;  1749          1871  1 !       This routine checks for the next symbol in the input stream being
;  1750          1872  1 !       a name which has been define with the DEFINE command. If so, and
;  1751          1873  1 !       if the kind matches the kind given in the input parameter, then
;  1752          1874  1 !       the corresponding value is returned in result_addr.
;  1753          1875  1 !
;  1754          1876  1 ! INPUTS
;  1755          1877  1 !       INPUT_DESC -    A string descriptor for the remaining input.
;  1756          1878  1 !
;  1757          1879  1 !       KIND -          The kind of DEFINE symbol we are expecting (one
;  1758          1880  1 !                       of define_address, define_command, define_procedure,
;  1759          1881  1 !                       define_string, or define_value)
;  1760          1882  1 !
;  1761          1883  1 !       RESULT_ADDR -   The address in which to leave the result
;  1762          1884  1 !
;  1763          1885  1 ! OUTPUTS
;  1764          1886  1 !       The routine value is one of:
;  1765          1887  1 !       TRUE -  A matching symbol was found
;  1766          1888  1 !       FALSE - A matching symbol was not found
;  1767          1889  1 !
;  1768          1890  1 !       If TRUE, the output parameter RESULT_ADDR is filled in.
;  1769          1891  1 !       Also, the input descriptor is updated to point past the name
;  1770          1892  1 !       that was read.
;  1771          1893  1 !
;  1772          1894  1
;  1773          1895  2     BEGIN
;  1774          1896  2
;  1775          1897  2     MAP
;  1776          1898  2         INPUT_DESC: REF BLOCK[,BYTE];   ! Input string descriptor
;  1777          1899  2
;  1778          1900  2     LOCAL
;  1779          1901  2         FOUND_FLAG,                     ! TRUE if we found a DEFINEd
;  1780          1902  2                                         !     symbol matching the input
;  1781          1903  2         GLOBAL_FLAG,                    ! Holds an output value for
;  1782          1904  2                                         !     DBG$DEF_SYM_FIND
;  1783          1905  2         MESSAGE_VECT,                   ! Dummy error message vector
;  1784          1906  2         NAME_PTR: REF VECTOR[,BYTE],    ! Points to a name which
;  1785          1907  2                                         !     could potentially be
;  1786          1908  2                                         !     a DEFINEd symbol.
;  1787          1909  2         SAVED_INPUT_LENGTH,             ! Copy of the length from the
;  1788          1910  2                                         !     original input descriptor
;  1789          1911  2         SAVED_INPUT_POINTER,            ! Copy of the pointer from the
;  1790          1912  2                                         !     original input   criptor
;  1791          1913  2         SYMBOL_KIND,                    ! Kind of defined sy
;  1792          1914  2         SYMBOL_VALUE;                   ! Value of DEFINEd symbol
;  1793          1915  2
;  1794          1916  2
;  1795          1917  2
;  1796          1918  2     ! Initialize found_flag.
;  1797          1919  2     !
;  1798          1920  2     FOUND_FLAG = FALSE;
;  1799          1921  2
;  1800          1922  2
;  1801          1923  2     ! First save away the input descriptor.
;  1802          1924  2     !
```

```
;  1803          1925  2      SAVED_INPUT_POINTER = .INPUT_DESC [DSC$A_POINTER];
;  1804          1926  2      SAVED_INPUT_LENGTH = .INPUT_DESC [DSC$W_LENGTH];
;  1805          1927  2
;  1806          1928  2
;  1807          1929  2      ! Now attempt to read a name which potentially could be DEFINEd.
;  1808          1930  2      !
;  1809          1931  2      IF DBG$NREAD_NAME (.INPUT_DESC, NAME_PTR, MESSAGE_VECT)
;  1810          1932  2      THEN
;  1811          1933  3          BEGIN
;  1812          1934  3
;  1813          1935  3
;  1814          1936  3          ! If we read a name, we now attempt to look it up.
;  1815          1937  3          !
;  1816          1938  3          IF DBG$DEF_SYM_FIND (.NAME_PTR, SYMBOL_KIND,
;  1817          1939  3                               SYMBOL_VALUE, GLOBAL_FLAG, MESSAGE_VECT)
;  1818          1940  3          THEN
;  1819          1941  4              BEGIN
;  1820          1942  4
;  1821          1943  4
;  1822          1944  4              ! Found a definition. Check the type of symbol. We expect
;  1823          1945  4              ! the kind to match what is given in "kind" at this point.
;  1824          1946  4              !
;  1825          1947  4              IF .SYMBOL_KIND EQL .KIND
;  1826          1948  4              THEN
;  1827          1949  5                  BEGIN
;  1828          1950  5                  .RESULT_ADDR = .SYMBOL_VALUE;
;  1829          1951  5                  FOUND_FLAG = TRUE;
;  1830          1952  4                  END;
;  1831          1953  4
;  1832          1954  3              END;
;  1833          1955  3
;  1834          1956  3
;  1835          1957  3          ! Deallocate the space for the name.
;  1836          1958  3          !
;  1837          1959  3          DBG$REL_MEMORY(.NAME_PTR);
;  1838          1960  2          END;
;  1839          1961  2
;  1840          1962  2
;  1841          1963  2      ! If we found a symbol, return success.  Otherwise, back up the input
;  1842          1964  2      ! descriptor so that it will be returned unchanged.
;  1843          1965  2      !
;  1844          1966  2      IF .FOUND_FLAG THEN RETURN TRUE;
;  1845          1967  2      INPUT_DESC [DSC$A_POINTER] = .SAVED_INPUT_POINTER;
;  1846          1968  2      INPUT_DESC [DSC$W_LENGTH] = .SAVED_INPUT_LENGTH;
;  1847          1969  2      RETURN FALSE;
;  1848          1970  2
;  1849          1971  1      END;


                                        003C 00000          .ENTRY  DBG$EXPAND_DEFINE_NAME, Save R2,R3,R4,R5    ; 1868
                                5E      14   C2 00002          SUBL2   #20, SP                                     ;
                                        53   D4 00005          CLRL    FOUND_FLAG                                  ; 1920
                                52  04  AC   D0 00007          MOVL    INPUT_DESC, R2                              ; 192?
                                54  04  A2   D0 0000B          MOVL    4(R2), SAVED_INPUT_POINTER                  ;
```

```
                                 55            62  3C 0000F          MOVZWL  (R2), SAVED_INPUT_LENGTH                ; 1926
                                         04    AE  9F 00012          PUSHAB  MESSAGE_VECT                            ; 1931
                                         04    AE  9F 00015          PUSHAB  NAME_PTR                                ;
                                               52  DD 00018          PUSHL   R2                                      ;
                          00000000G      00     03  FB 0001A          CALLS   #3, DBG$NREAD_NAME                      ;
                                         31     50  E9 00021          BLBC    R0, 2$                                  ;
                                         04    AE  9F 00024          PUSHAB  MESSAGE_VECT                            ; 1938
                                         0C    AE  9F 00027          PUSHAB  GLOBAL_FLAG                             ;
                                         14    AE  9F 0002A          PUSHAB  SYMBOL_VALUE                            ;
                                         1C    AE  9F 0002D          PUSHAB  SYMBOL_KIND                             ;
                                         10    AE  DD 00030          PUSHL   NAME_PTR                                ;
                          00000000G      00     05  FB 00033          CALLS   #5, DBG$DEF_SYM_FIND                    ;
                                         0F     50  E9 0003A          BLBC    R0, 1$                                  ;
                                08       AC    10  AE  D1 0003D          CMPL    SYMBOL_KIND, KIND                       ; 1947
                                               08  12 00042          BNEQ    1$                                      ;
                                0C       BC    0C  AE  D0 00044          MOVL    SYMBOL_VALUE, @RESULT_ADDR              ; 1950
                                         53          01  D0 00049          MOVL    #1, FOUND_FLAG                          ; 1951
                                                     6E  DD 0004C 1$:       PUSHL   NAME_PTR                                ; 1959
                          00000000G      00         01  FB 0004E          CALLS   #1, DBG$REL_MEMORY                      ;
                                         04         53  E9 00055 2$:       BLBC    FOUND_FLAG, 3$                          ; 1966
                                         50         01  D0 00058          MOVL    #1, R0                                  ;
                                                        04 0005B          RET                                             ;
                                04       A2        54  D0 0005C 3$:       MOVL    SAVED_INPUT_POINTER, 4(R2)              ; 1967
                                         62        55  B0 00060          MOVW    SAVED_INPUT_LENGTH, (R2)                ; 1968
                                                   50  D4 00063          CLRL    R0                                      ; 1969
                                                       04 00065          RET                                             ; 1971

; Routine Size:  102 bytes,    Routine Base:  DBG$CODE + 0BD0
```

```
;1851        1972  1 GLOBAL ROUTINE DBG$SYNTAX_ERROR(INPUT_DESC): NOVALUE =
;1852        1973  1 !
;1853        1974  1 ! FUNCTION
;1854        1975  1 !       This routine is called when a syntax error has occurred because
;1855        1976  1 !       an expected piece of input was not found.  The routine accepts
;1856        1977  1 !       an input string descriptor as input which is expected to point
;1857        1978  1 !       to the start of the offending piece of command input.  It then
;1858        1979  1 !       signals the "need more input" message if the next character is
;1859        1980  1 !       a carriage-return (unexpected end of input) or the "syntax error"
;1860        1981  1 !       error message otherwise.
;1861        1982  1 !
;1862        1983  1 ! INPUTS
;1863        1984  1 !       INPUT_DESC - A pointer to a string descriptor which points to the
;1864        1985  1 !                    current parse position in the input command line.
;1865        1986  1 !
;1866        1987  1 ! OUTPUTS
;1867        1988  1 !       INPUT_DESC - The input string descriptor may be updated to point
;1868        1989  1 !                    to the first character after the offending syntactic
;1869        1990  1 !                    entity.
;1870        1991  1 !
;1871        1992  1 !       Since this routine always exits by signalling out, no value is
;1872        1993  1 !                    ever returned.  Control is not returned either.
;1873        1994  1 !
;1874        1995  1
;1875        1996  2     BEGIN
;1876        1997  2
;1877        1998  2     BIND
;1878        1999  2         DBG$CS_CR       = UPLIT BYTE(1, 13);    ! Carriage return character
;1879        2000  2
;1880        2001  2     LOCAL
;1881        2002  2         STRDESC: BLOCK[8,BYTE],         ! String descriptor
;1882        2003  2         STRPTR: REF VECTOR[,BYTE];      ! Pointer to error string
;1883        2004  2
;1884        2005  2
;1885        2006  2
;1886        2007  2     ! If the line has already ended, signal the "need more" error message.
;1887        2008  2     ! Otherwise, signal the "syntax error" error message.
;1888        2009  2     !
;1889        2010  2     IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1) THEN SIGNAL(DBG$_NEEDMORE);
;1890        2011  2     STRPTR = DBG$NNEXT_WORD(.INPUT_DESC);
;1891        2012  2     STRDESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;1892        2013  2     STRDESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;1893        2014  2     STRDESC[DSC$W_LENGTH] = .STRPTR[0];
;1894        2015  2     STRDESC[DSC$A_POINTER] = STRPTR[1];
;1895        2016  2     SIGNAL(DBG$_SYNTAX, 1, STRDESC);
;1896        2017  2     RETURN;
;1897        2018  2
;1898        2019  1     END;


                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
                    0D  01  0012E P.ABO:  .BYTE   1, 13                                 ;
                                  DBG$CS_CR=          P.ABO
```

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

                                                    0004 00000          .ENTRY  DBG$SYNTAX_ERROR, Save R2                       ; 1972
                                          52 00000000G  00   9E 00002  MOVAB   LIB$SIGNAL, R2                                 ;
                                          5E            08   C2 00009  SUBL2   #8, SP                                        ;
                                                        01   DD 0000C  PUSHL   #1                                           ; 2010
                                             00000000'  EF   9F 0000E  PUSHAB  DBG$CS_CR                                     ;
                                                   04   AC   DD 00014  PUSHL   INPUT_DESC                                    ;
                                    F8ED          CF   03   FB 00017  CALLS   #3, DBG$NMATCH                                ;
                                                  09   50   E9 0001C  BLBC    R0, 1$                                        ;
                                             000280D0  8F   DD 0001F  PUSHL   #164048                                       ;
                                                  62   01   FB 00025  CALLS   #1, LIB$SIGNAL                               ;
                                                   04   AC   DD 00028  1$:    PUSHL   INPUT_DESC                            ; 2011
                                    F935          CF   01   FB 0002B  CALLS   #1, DBG$NNEXT_WORD                          ;
                                      02          AE  010E  8F   B0 00030  MOVW    #270, STRDESC+2                         ; 2013
                                                  6E   60   9B 00036  MOVZBW  (STRPTR), STRDESC                           ; 2014
                                      04          AE   01   A0   9E 00039  MOVAB   1(R0), STRDESC+4                        ; 2015
                                                        5E   DD 0003E  PUSHL   SP                                         ; 2016
                                                        01   DD 00040  PUSHL   #1                                         ;
                                             00028238   8F   DD 00042  PUSHL   #164408                                      ;
                                                  62   03   FB 00048  CALLS   #3, LIB$SIGNAL                              ;
                                                        04   00004B  RET                                                    ; 2019

; Routine Size:  76 bytes,    Routine Base:  DBG$CODE + 0C56

; 1899          2020  1
; 1900          2021  0 END ELUDOM


                                                                .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY

;      Name                        Bytes                           Attributes

;  DBG$OWN                           81  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$PLIT                         304  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                        3202  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         8         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        63         4          97        00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                            418         0         0          31        00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1       386        11         2          22        00:00.3
```

```
;                                       COMMAND QUALIFIERS

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

; Size:         3202 code + 385 data bytes
; Run Time:        01:03.1
; Elapsed Time:    03:14.5
; Lines/CPU Min:    1922
; Lexemes/CPU-Min: 12329
; Memory Used:  462 pages
; Compilation Complete
```

DBGNERMSG
LIS

DBGNHELP
LIS

DBGNPARSE
LIS

DBGNEXCTE
LIS

DBGNPNP
LIS