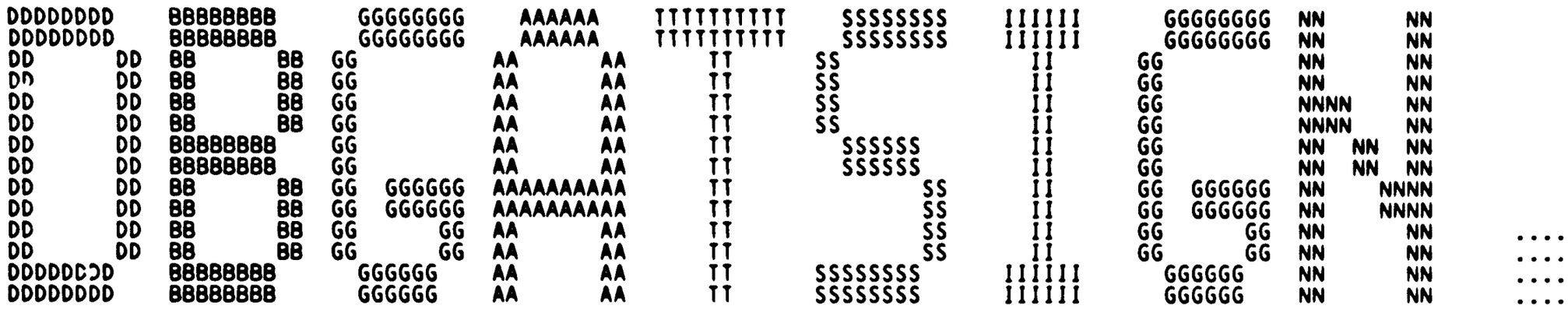
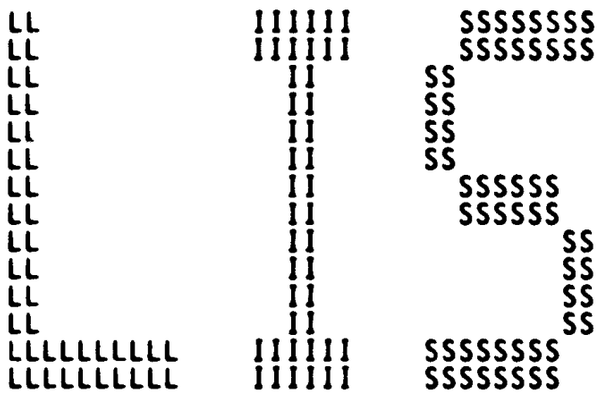
```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB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

FILEID**DBGATSIGN





```
;     1          0001  0 MODULE DBGATSIGN (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !
;     6          0006  1 !**********************************************************************
;     7          0007  1 !*                                                                    *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                           *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.            *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                             *
;    11          0011  1 !*                                                                    *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    16          0016  1 !*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  *
;    17          0017  1 !*  TRANSFERRED.                                                      *
;    18          0018  1 !*                                                                    *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                      *
;    22          0022  1 !*                                                                    *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.           *
;    25          0025  1 !*                                                                    *
;    26          0026  1 !*                                                                    *
;    27          0027  1 !**********************************************************************
;    28          0028  1 !
;    29          0029  1 !
;    30          0030  1 ! MODULE FUNCTION
;    31          0031  1 !
;    32          0032  1 !       This module contains the parse and execution networks to support the
;    33          0033  1 !       commands having to do with command files (@, DECLARE, EXIT, EXITLOOP).
;    34          0034  1 !       SPAWN and ATTACH are also in this module (for lack of a better
;    35          0035  1 !       place to put them).
;    36          0036  1 !
;    37          0037  1 ! AUTHOR:       David Plummer, CREATION DATE: 3/28/80
;    38          0038  1 !
;    39          0039  1 ! MODIFIED BY:
;    40          0040  1 !
;    41          0041  1 !       R. Title        - Added SPAWN, ATTACH, DECLARE, EXITLOOP, and merged
;    42          0042  1 !                         all of this into this one module.
;    43          0043  1 !
;    44          0044  1
;    45          0045  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    46          0179  1
;    47          0180  1 LIBRARY 'LIB$:DBGGEN.L32';
;    48          0181  1
;    49          0182  1 FORWARD ROUTINE
;    50          0183  1     DBG$NPARSE_AT_SIGN,                 ! Parse indirect command file invocation
;    51          0184  1     DBG$NEXECUTE_AT_SIGN,               ! Execute indirect command file invocation
;    52          0185  1     DBG$NPARSE_ATTACH,                  ! ATTACH command parse network
;    53          0186  1     DBG$NEXECUTE_ATTACH,                ! ATTACH command execute network
;    54          0187  1     DBG$NPARSE_SPAWN,                   ! SPAWN command parse network
;    55          0188  1     DBG$NEXECUTE_SPAWN,                 ! SPAWN command execution network
;    56          0189  1     DBG$CONSTRUCT_PARAM_NAME,           ! Builds name of form %Pi
;    57          0190  1     DBG$PARSE_SPAWN,                    ! Interface from old languages
```

```
;    58          0191  1     GET_PARAM,                  ! Parses a parameter
;    59          0192  1     DBG$NEXECUTE_EXIT,          ! Execution network for EXIT command
;    60          0193  1     DBG$NEXECUTE_EXITLOOP,      ! Execution network for EXITLOOP
;    61          0194  1     DBG$NPARSE_EXIT,            ! ATN parse network for EXIT command
;    62          0195  1     DBG$NPARSE_EXITLOOP,        ! ATN parse network for EXITLOOP
;    63          0196  1     DBG$NPARSE_DECLARE,         ! DECLARE parse network
;    64          0197  1     DBG$NEXECUTE_DECLARE;       ! DECLARE execution network
```

```
;    66          0198  1 EXTERNAL ROUTINE
;    67          0199  1     DBG$DEF_PR_ENTRY,                ! Procedure entry routine for a procedures
;    68          0200  1     DBG$DEF_SYM_ADD,                 ! Adds a DEFINE symbol
;    69          0201  1     DBG$DEF_SYM_FIND,                ! Locate a DEFINE symbol
;    70          0202  1     DBG$GET_MEMORY,                  ! Allocates permanent memory
;    71          0203  1     DBG$GET_TEMPMEM,                 ! Allocates and lists dynamic storage
;    72          0204  1     DBG$NCIS_ADD,                    ! Routine to add a node to the cis
;    73          0205  1     DBG$NCIS_OPENICF,                ! Routine to connect an icf
;    74          0206  1     DBG$NCIS_REMOVE,                 ! Removes links from the command input stream
;    75          0207  1     DBG$NCOPY_DESC,                  ! Copy a descriptor
;    76          0208  1     DBG$NEWLINE: NOVALUE,            ! Output routine
;    77          0209  1     DBG$NGET_SYMID,                  ! Obtain symid list
;    78          0210  1     DBG$NMAKE_ARG_VECT,              ! Construct error message vector
;    79          0211  1     DBG$NMATCH,                      ! Match the next token
;    80          0212  1     DBG$NNEXT_WORD,                  ! Read next work of input
;    81          0213  1     DBG$NPARSE_ADDRESS,              ! Parse an address expression
;    82          0214  1     DBG$NPARSE_EXPRESSION,           ! Parse a language expression
;    83          0215  1     DBG$NREAD_NAME,                  ! Accept a DEFINE name
;    84          0216  1     DBG$NSAVE_BREAK_BUFFER,          ! Parse a DEBUG command sequence
;    85          0217  1     DBG$NSAVE_DECIMAL_INTEGER,       ! Parse an integer
;    86          0218  1     DBG$NSAVE_FILESP,                ! Saves a file specification string
;    87          0219  1     DBG$NSAVE_STRING,                ! Pick up string
;    88          0220  1     DBG$NSYNTAX_ERROR,               ! Construct a syntax error message
;    89          0221  1     DBG$PRINT: NOVALUE,              ! Output routine
;    90          0222  1     DBG$SCR_SCREEN_NORMAL: NOVALUE,  ! Set screen to normal in screen mode
;    91          0223  1     DBG$STA_LOCK_SYMID: NOVALUE,     ! Lock symid list
;    92          0224  1     LIB$ATTACH,                      ! Do the actual ATTACH to a process
;    93          0225  1     LIB$FREE_EF,                     ! Free up and event-flag
;    94          0226  1     LIB$GET_EF,                      ! Allocate an event-flag
;    95          0227  1     LIB$SPAWN,                       ! Do the actual SPAWN of a subprocess
;    96          0228  1     SMG$SET_KEYPAD_MODE;             ! Set terminal to numeric/application mode
;    97          0229  1
;    98          0230  1 EXTERNAL
;    99          0231  1     DBG$GB_KEYPAD_INPUT: BYTE,       ! Set if mode is KEYPAD
;   100          0232  1     DBG$GL_KEYBOARD_ID,              ! Used by SMG$SET_KEYPAD_MODE
;   101          0233  1     DBG$GB_LANGUAGE: BYTE,           ! The current language setting
;   102          0234  1     DBG$GB_RADIX    : VECTOR [3, BYTE],  ! Radix settings
;   103          0235  1     DBG$GB_DEF_OUT  : VECTOR [,BYTE],    !    "     " output configuration structure
;   104          0236  1     DBG$GL_LOGFAB   : BLOCK [,BYTE],     !    "     " FAB for log file
;   105          0237  1     DBG$GL_CISHEAD  : REF CIS$LINK,      !    "     " head of command input stream
;   106          0238  1     DBG$GL_EXIT_STATUS,              ! Last known user error status
;   107          0239  1     DBG$GV_CONTROL: DBG$CONTROL_FLAGS,   ! DEBUG control bits
;   108          0240  1     DBG$GL_SMG_EXIT_HANDLER;         ! Address of exit handler for SMG routines
;   109          0241  1
;   110          0242  1 LITERAL
;   111          0243  1     NOWAIT_FLAG = 0;                 ! Flag position for /NOWAIT
;   112          0244  1
;   113          0245  1 GLOBAL LITERAL
;   114          0246  1     DBG$K_MAX_PR_NESTING = 10;
;   115          0247  1
;   116          0248  1 GLOBAL
;   117          0249  1     DBG$GL_PR_NEST_LEVEL: INITIAL(0),    ! Level of nesting of a procedures
;   118          0250  1     DBG$GL_PARAM_COUNT:              ! Parameter count for each procedure
;   119          0251  1                 VECTOR[1+DBG$K_MAX_PR_NESTING]
;   120          0252  1                 INITIAL(REP 1+DBG$K_MAX_PR_NESTING OF (0));
;   121          0253  1
;   122          0254  1
```

```
;  123          0255  1 ! The following macro is just an abbreviation for some error-reporting
;  124          0256  1 ! code that occurs repeatedly
;  125          0257  1 !
;  126        M 0258  1 MACRO report_error =
;  127        M 0259  1     BEGIN
;  128        M 0260  1     .message_vect = (
;  129        M 0261  1         IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  130        M 0262  1         THEN
;  131        M 0263  1             dbg$nmake_arg_vect (dbg$_needmore)
;  132        M 0264  1         ELSE
;  133        M 0265  1             dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;  134        M 0266  1     RETURN sts$k_severe;
;  135          0267  1     END %;
```

```
;  137     0268  1 GLOBAL ROUTINE DBG$NPARSE_AT_SIGN (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;  138     0269  1
;  139     0270  1 !++
;  140     0271  1 ! FUNCTIONAL DESCRIPTION:
;  141     0272  1 !
;  142     0273  1 !       Parses an @ filespec command. The @ has already been recognized in the routine
;  143     0274  1 !       dbg$nparse_cmd network. This routine merely aquires the filespec string
;  144     0275  1 !       and causes the noun node ( noun value ) field to point to the filespec which
;  145     0276  1 !       is stored as a counted string in dynamically allocated storage.
;  146     0277  1 !
;  147     0278  1 !
;  148     0279  1 ! FORMAL PARAMETERS:
;  149     0280  1 !
;  150     0281  1 !       input_desc -                    The present command line input descriptor
;  151     0282  1 !
;  152     0283  1 !       verb_node  -                    The already existing command verb node
;  153     0284  1 !
;  154     0285  1 !       message_vect -                  The address of a longword to contain the
;  155     0286  1 !                                       address of a message argument vector
;  156     0287  1 !
;  157     0288  1 ! IMPLICIT INPUTS:
;  158     0289  1 !
;  159     0290  1 !       NONE
;  160     0291  1 !
;  161     0292  1 ! IMPLICIT OUTPUTS:
;  162     0293  1 !
;  163     0294  1 !       On success, the appropriate command execution tree is constructed.
;  164     0295  1 !
;  165     0296  1 !       On failure, a message argument vector is constructed
;  166     0297  1 !
;  167     0298  1 ! ROUTINE VALUE:
;  168     0299  1 !
;  169     0300  1 !       An unsigned integer longword completion code
;  170     0301  1 !
;  171     0302  1 ! COMPLETION CODES:
;  172     0303  1 !
;  173     0304  1 !       sts$k_severe (4) -              Error in parsing
;  174     0305  1 !
;  175     0306  1 !       sts$k_success (1) -             Successful parse
;  176     0307  1 !
;  177     0308  1 ! SIDE EFFECTS:
;  178     0309  1 !
;  179     0310  1 !       None
;  180     0311  1 !
;  181     0312  1 !--
;  182     0313  1
;  183     0314  2     BEGIN
;  184     0315  2
;  185     0316  2     MAP
;  186     0317  2         INPUT_DESC: REF dbg$stg_desc,
;  187     0318  2         VERB_NODE : REF dbg$verb_node;
;  188     0319  2
;  189     0320  2     BIND
;  190     0321  2         dbg$cs_comma            = UPLIT BYTE (1, dbg$k_comma),
;  191     0322  2         dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
;  192     0323  2         dbg$cs_left_paren       = UPLIT BYTE (1, dbg$k_left_parenthesis),
;  193     0324  2         dbg$cs_right_paren      = UPLIT BYTE (1, dbg$k_right_parenthesis);
```

```
;  194          0325  2
;  195          0326  2      LOCAL
;  196          0327  2          first_flag: BYTE,                   ! True first time around loop that
;  197          0328  2                                              !    collect the parameters.
;  198          0329  2          link,                               ! Used for building linked lists
;  199          0330  2          noun_node: REF dbg$noun_node,       ! A noun node
;  200          0331  2          paren_flag: BYTE;                   ! True if there is a parenthesized
;  201          0332  2                                              !    parameter list.
;  202          0333  2
;  203          0334  2      ! Create and link a noun node
;  204          0335  2      !
;  205          0336  2      noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  206          0337  2      verb_node [dbg$l_verb_object_ptr] = .noun_node;
;  207          0338  2
;  208          0339  2      ! Get the filespec
;  209          0340  2      !
;  210          0341  2      IF NOT dbg$nsave_filesp (.input_desc, noun_node [dbg$l_noun_value], .message_vect)
;  211          0342  2      THEN
;  212          0343  2          RETURN sts$k_severe;
;  213          0344  2
;  214          0345  2      ! Check for end-of-line. In this case, there are no parameters so just
;  215          0346  2      ! return success.
;  216          0347  2      !
;  217          0348  2      IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  218          0349  2      OR .input_desc [dsc$w_length] EQL 0
;  219          0350  2      THEN
;  220          0351  2          RETURN sts$k_success;
;  221          0352  2
;  222          0353  2      ! Now pick up the parameter list, if one is present.
;  223          0354  2      !
;  224          0355  2      IF dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  225          0356  2      THEN
;  226          0357  2          paren_flag = TRUE
;  227          0358  2      ELSE
;  228          0359  2          paren_flag = FALSE;
;  229          0360  2
;  230          0361  2      ! Set a flag which is true only the first time around the loop.
;  231          0362  2      !
;  232          0363  2      first_flag = TRUE;
;  233          0364  2
;  234          0365  2      ! Loop through the list of parameters.
;  235          0366  2      !
;  236          0367  2      WHILE TRUE DO
;  237          0368  3          BEGIN
;  238          0369  3
;  239          0370  3          ! Allocate space for a new noun node.
;  240          0371  3          !
;  241          0372  3          link = noun_node [dbg$l_noun_link];
;  242          0373  3          noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  243          0374  3
;  244          0375  3          ! Attempt to read a parameter.
;  245          0376  3          !
;  246          0377  3          IF NOT get_param (.input_desc, .paren_flag,
;  247          0378  3                  noun_node [dbg$l_noun_value])
;  248          0379  3          THEN
;  249          0380  4              BEGIN
;  250          0381  4
```

```
;  251          0382  4          ! The only way this can legally fail is @FOO ()
;  252          0383  4          ! (no parameters). Check for this.
;  253          0384  4          !
;  254          0385  4          IF .first_flag
;  255          0386  4          THEN
;  256          0387  4
;  257          0388  4              ! If there had been an opening paren, then make sure we
;  258          0389  4              ! collect the closing paren
;  259          0390  4              !
;  260          0391  4              IF .paren_flag
;  261          0392  4              THEN
;  262          0393  4                  IF NOT dbg$nmatch (.input_desc, dbg$cs_right_paren, 1)
;  263          0394  4                  THEN
;  264          0395  5                      BEGIN
;  265          0396  5                      .message_vect = dbg$nmake_arg_vect (dbg$_unmtchparn);
;  266          0397  5                      RETURN sts$k_severe;
;  267          0398  5                      END
;  268          0399  4                  ELSE
;  269          0400  4                      EXITLOOP;
;  270          0401  4
;  271          0402  4          ! If we get here, we have read a separating comma, but we have
;  272          0403  4          ! failed to read the next parameter (E.G., @FOO A,)
;  273          0404  4          ! This is therefore an error.
;  274          0405  4          !
;  275          0406  4          .message_vect =
;  276          0407  5                  (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  277          0408  5                   THEN
;  278          0409  5                       dbg$nmake_arg_vect (dbg$_needmore)
;  279          0410  5                   ELSE
;  280          0411  4                       dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;  281          0412  4          RETURN sts$k_severe;
;  282          0413  3          END;
;  283          0414  3
;  284          0415  3      ! Set first_flag to FALSE for next time around.
;  285          0416  3      !
;  286          0417  3      first_flag = FALSE;
;  287          0418  3
;  288          0419  3      ! We must have succeeded in reading a parameter. Thus, link in
;  289          0420  3      ! the noun node.
;  290          0421  3      !
;  291          0422  3      .link = .noun_node;
;  292          0423  3
;  293          0424  3      ! Attempt to read a comma, which separates parameters.
;  294          0425  3      !
;  295          0426  3      IF NOT dbg$nmatch (.input_desc, dbg$cs_comma, 1)
;  296          0427  3      THEN
;  297          0428  3
;  298          0429  3          ! If we fail, check for the closing paren, if an opening
;  299          0430  3          ! paren was present.
;  300          0431  3          !
;  301          0432  3          IF .paren_flag
;  302          0433  3          THEN
;  303          0434  3              IF NOT dbg$nmatch (.input_desc, dbg$cs_right_paren, 1)
;  304          0435  3              THEN
;  305          0436  4                  BEGIN
;  306          0437  4                  .message_vect = dbg$nmake_arg_vect (dbg$_unmtchparn);
;  307          0438  4                  RETURN sts$k_severe;
```

```
;  308          0439  4                    END
;  309          0440  3                ELSE
;  310          0441  3                    EXITLOOP
;  311          0442  3
;  312          0443  3                ! The lack of a separating comma indicates the end of the
;  313          0444  3                ! parameter list, so we just exit the loop.
;  314          0445  3                !
;  315          0446  3                ELSE
;  316          0447  3                    EXITLOOP;
;  317          0448  3
;  318          0449  2            END; ! of WHILE loop
;  319          0450  2
;  320          0451  2        ! The command execution tree, containing a noun node with the filespec
;  321          0452  2        ! and a noun node for each parameter, has been constructed. Thus, we
;  322          0453  2        ! just return success.
;  323          0454  2        !
;  324          0455  2        RETURN sts$k_success;
;  325          0456  2
;  326          0457  1        END;                        ! End of dbg$nparse_at_sign
```

```
                                        .TITLE  DBGATSIGN
                                        .IDENT  \V04-000\

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

                        2C  01  00000 P.AAA:    .BYTE   1, 44                                   ;
                        0D  01  00002 P.AAB:    .BYTE   1, 13                                   ;
                        28  01  00004 P.AAC:    .BYTE   1, 40                                   ;
                        29  01  00006 P.AAD:    .BYTE   1, 41                                   ;

                                        .PSECT  DBG$GLOBAL,NOEXE,  PIC,2

                00000000  00000 DBG$GL_PR_NEST_LEVEL::
                                        .LONG   0                                               ;
                00000000# 00004 DBG$GL_PARAM_COUNT::
                                        .LONG   0[11]                                           ;

                                DBG$K_MAX_PR_NESTING==
                                                10
                                DBG$CS_COMMA=       P.AAA
                                DBG$CS_CR=          P.AAB
                                DBG$CS_LEFT_PAREN=  P.AAC
                                DBG$CS_RIGHT_PAREN= P.AAD
                                        .EXTRN  DBG$DEF_PR_ENTRY
                                        .EXTRN  DBG$DEF_SYM_ADD
                                        .EXTRN  DBG$DEF_SYM_FIND
                                        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
                                        .EXTRN  DBG$NCIS_ADD, DBG$NCIS_OPENICF
                                        .EXTRN  DBG$NCIS_REMOVE
                                        .EXTRN  DBG$NCOPY_DESC, DBG$NEWLINE
                                        .EXTRN  DBG$NGET_SYMID, DBG$NMAKE_ARG_VECT
                                        .EXTRN  DBG$NMATCH, DBG$NNEXT_WORD
                                        .EXTRN  DBG$NPARSE_ADDRESS
                                        .EXTRN  DBG$NPARSE_EXPRESSION
                                        .EXTRN  DBG$NREAD_NAME, DBG$NSAVE_BREAK_BUFFER
                                        .EXTRN  DBG$NSAVE_DECIMAL_INTEGER
```

```
                                        .EXTRN  DBG$NSAVE_FILESP
                                        .EXTRN  DBG$NSAVE_STRING
                                        .EXTRN  DBG$NSYNTAX_ERROR
                                        .EXTRN  DBG$PRINT, DBG$SCR_SCREEN_NORMAL
                                        .EXTRN  DBG$STA_LOCK_SYMID
                                        .EXTRN  LIB$ATTACH, LIB$FREE_EF
                                        .EXTRN  LIB$GET_EF, LIB$SPAWN
                                        .EXTRN  SMG$SET_KEYPAD_MODE
                                        .EXTRN  DBG$GB_KEYPAD_INPUT
                                        .EXTRN  DBG$GL_KEYBOARD_ID
                                        .EXTRN  DBG$GB_LANGUAGE
                                        .EXTRN  DBG$GB_RADIX, DBG$GB_DEF_OUT
                                        .EXTRN  DBG$GL_LOGFAB, DBG$GL_CISHEAD
                                        .EXTRN  DBG$GL_EXIT_STATUS
                                        .EXTRN  DBG$GV_CONTROL, DBG$GL_SMG_EXIT_HANDLER

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

                                03FC 00000      .ENTRY  DBG$NPARSE_AT_SIGN, Save R2,R3,R4,R5,R6,R7,-; 0268
                                                        R8,R9
                     59 00000000G   00   9E 00002      MOVAB   DBG$GET_TEMPMEM, R9
                     58 00000000G   00   9E 00009      MOVAB   DBG$NMATCH, R8
                     57 00000000'   EF   9E 00010      MOVAB   DBG$CS_CR, R7
                                    04   DD 00017      PUSHL   #4                                      ; 0336
                     69             01   FB 00019      CALLS   #1, DBG$GET_TEMPMEM
                     53             50   D0 0001C      MOVL    R0, NOUN_NODE
                     50       08    AC   D0 0001F      MOVL    VERB_NODE, R0                           ; 0337
               08    A0             53   D0 00023      MOVL    NOUN_NODE, 8(R0)
                            0C      AC   DD 00027      PUSHL   MESSAGE_VECT                            ; 0341
                                    53   DD 0002A      PUSHL   NOUN_NODE
                     52       04    AC   D0 0002C      MOVL    INPUT_DESC, R2
                                    52   DD 00030      PUSHL   R2
          00000000G  00             03   FB 00032      CALLS   #3, DBG$NSAVE_FILESP
                     03             50   E8 00039      BLBS    R0, 1$
                                00A6 31   0003C      BRW     13$
                                01   DD 0003F 1$:    PUSHL   #1                                      ; 0348
                              0084   8F   BB 00041      PUSHR   #^M<R2,R7>
                     68             03   FB 00045      CALLS   #3, DBG$NMATCH
                     03             50   E9 00048      BLBC    R0, 3$
                                009B 31   0004B 2$:    BRW     14$
                                    62   B5 0004E 3$:    TSTW    (R2)                                ; 0349
                                    F9   13 00050      BEQL    2$
                                    01   DD 00052      PUSHL   #1                                      ; 0355
                                 02 A7   9F 00054      PUSHAB  DBG$CS_LEFT_PAREN
                                    52   DD 00057      PUSHL   R2
                     68             03   FB 00059      CALLS   #3, DBG$NMATCH
                     05             50   E9 0005C      BLBC    R0, 4$
                     54             01   90 0005F      MOVB    #1, PAREN_FLAG                          ; 0357
                                    02   11 00062      BRB     5$
                                    54   94 00064 4$:    CLRB    PAREN_FLAG                          ; 0359
                     55             01   90 00066 5$:    MOVB    #1, FIRST_FLAG                      ; 0363
                     56       08    A3   9E 00069 6$:    MOVAB   8(R3), LINK                         ; 0372
                                    04   DD 0006D      PUSHL   #4                                      ; 0373
                     69             01   FB 0006F      CALLS   #1, DBG$GET_TEMPMEM
                     53             50   D0 00072      MOVL    R0, NOUN_NODE
                                    53   DD 00075      PUSHL   NOUN_NODE                               ; 0378
                     7E             54   9A 00077      MOVZBL  PAREN_FLAG, -(SP)
```

```
                                52  DD 0007A          PUSHL   R2                                      ;
                          0000V CF  03  FB 0007C      CALLS   #3, GET_PARAM                           ;
                                2E  50  E8 00081      BLBS    R0, 9$                                  ;
                                03  55  E9 00084      BLBC    FIRST_FLAG, 7$                          ; 0385
                                3D  54  E8 00087      BLBS    PAREN_FLAG, 10$                         ; 0391
                                    01  DD 0008A 7$:  PUSHL   #1                                      ; 0407
                                  0084  8F  BB 0008C  PUSHR   #^M<R2,R7>                              ;
                                68      03  FB 00090  CALLS   #3, DBG$NMATCH                         ;
                                08      50  E9 00093  BLBC    R0, 8$                                  ;
                                    000280D0  8F  DD 00096  PUSHL   #164048                         ; 0409
                                              3C  11 0009C  BRB     11$                               ;
                                              52  DD 0009E 8$:  PUSHL   R2                          ; 0411
                  00000000G     00          01  FB 000A0  CALLS   #1, DBG$NNEXT_WORD                 ;
                                            50  DD 000A7  PUSHL   R0                                  ;
                  00000000G     00          01  FB 000A9  CALLS   #1, DBG$NSYNTAX_ERROR              ;
                                            2F  11 000B0  BRB     12$                               ; 0407
                                            55  94 000B2 9$:  CLRB    FIRST_FLAG                    ; 0417
                                66          53  D0 000B4  MOVL    NOUN_NODE, (LINK)                 ; 0422
                                            01  DD 000B7  PUSHL   #1                                 ; 0426
                                  FE        A7  9F 000B9  PUSHAB  DBG$CS_COMMA                      ;
                                            52  DD 000BC  PUSHL   R2                                  ;
                                68          03  FB 000BE  CALLS   #3, DBG$NMATCH                     ;
                                A5          50  E8 000C1  BLBS    R0, 6$                            ;
                                22          54  E9 000C4  BLBC    PAREN_FLAG, 14$                   ; 0432
                                            01  DD 000C7 10$:  PUSHL   #1                            ; 0434
                                  04        A7  9F 000C9  PUSHAB  DBG$CS_RIGHT_PAREN                ;
                                            52  DD 000CC  PUSHL   R2                                  ;
                                68          03  FB 000CE  CALLS   #3, DBG$NMATCH                     ;
                                15          50  E8 000D1  BLBS    R0, 14$                           ;
                                    000287D0  8F  DD 000D4  PUSHL   #165840                         ; 0437
                  00000000G     00          01  FB 000DA 11$:  CALLS   #1, DBG$NMAKE_ARG_VECT       ;
                          0C    BC          50  D0 000E1 12$:  MOVL    R0, @MESSAGE_VECT             ;
                                50          04  D0 000E5 13$:  MOVL    #4, R0                        ; 0438
                                                04 000E8      RET                                     ;
                                50          01  D0 000E9 14$:  MOVL    #1, R0                        ; 0455
                                                04 000EC      RET                                     ; 0457

; Routine Size:  237 bytes,    Routine Base:  DBG$CODE + 0000

;  327          0458  1
```

```
;  329          0459  1 GLOBAL ROUTINE DBG$NEXECUTE_AT_SIGN (VERB_NODE, MESSAGE_VECT) =
;  330          0460  1 !
;  331          0461  1 ! FUNCTION
;  332          0462  1 !       This routine begins execution of an @ filespec command.  That is, a FAB
;  333          0463  1 !       and a RAB are allocated for the indirect command file, and calls are
;  334          0464  1 !       made to add the file to the command input string and connect the file.
;  335          0465  1 !
;  336          0466  1 ! INPUTS
;  337          0467  1 !       VERB_NODE - The command verb node which is the start of the command
;  338          0468  1 !                   execution tree.
;  339          0469  1 !
;  340          0470  1 !       MESSAGE_VECT - The address of a longword to contain the address of
;  341          0471  1 !                   a message argument vector.  On failure of this routine,
;  342          0472  1 !                   a message argument vector is returned to this parameter.
;  343          0473  1 !
;  344          0474  1 ! OUTPUTS
;  345          0475  1 !       An unsigned integer longword completion code is returned as the routine
;  346          0476  1 !       value:
;  347          0477  1 !
;  348          0478  1 !           STS$K_SUCCESS (1) - Indicates the file was created and connected
;  349          0479  1 !
;  350          0480  1 !           STS$K_SEVERE (4) -  Failure, file not created. Message argument vector returned
;  351          0481  1 !
;  352          0482  1
;  353          0483  2     BEGIN
;  354          0484  2
;  355          0485  2     BIND
;  356          0486  2         DEFICF_NAME     = UPLIT BYTE ('DEBUG.COM'),
;  357          0487  2         DEFICF_SIZE     = %CHARCOUNT ('DEBUG.COM');
;  358          0488  2
;  359          0489  2     MAP
;  360          0490  2         VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to input Verb Node
;  361          0491  2
;  362          0492  2     LOCAL
;  363          0493  2         FAB_PTR: REF $FAB_DECL,          ! Pointer to ICF FAB
;  364          0494  2         NOUN_NODE: REF DBG$NOUN_NODE,    ! Pointer to command Noun Node
;  365          0495  2         PARAM_COUNT,                     ! Count of number of parameters
;  366          0496  2         PARAM_NAME: REF VECTOR[,BYTE],   ! Pointer to ASCIC name of parameter
;  367          0497  2         RAB_PTR: REF $RAB_DECL,          ! Pointer to ICF RAB
;  368          0498  2         REPLACED_FLAG,                   ! Returned symbol-replaced flag from
;  369          0499  2                                          !       DBG$DEF_SYM_ADD (value not used)
;  370          0500  2         VALARRAY: REF VECTOR[,LONG],     ! Pointer to value field in the Value
;  371          0501  2                                          !       Descriptor for %PARCNT
;  372          0502  2         VALPTR: REF DBG$VALDESC;         ! Pointer to Value Descriptor we build
;  373          0503  2
;  374          0504  2
;  375          0505  2
;  376          0506  2     ! Obtain the Noun Node for the file name of the indirect command file.
;  377          0507  2     !
;  378          0508  2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
;  379          0509  2
;  380          0510  2
;  381          0511  2     ! Allocate a FAB and a RAB for the command file and initialize them.
;  382          0512  2     !
;  383          0513  2     FAB_PTR = DBG$GET_MEMORY ((FAB$C_BLN + 3) / %UPVAL);
;  384          0514  2     RAB_PTR = DBG$GET_MEMORY ((RAB$C_BLN + 3) / %UPVAL);
;  385        P 0515  2     $FAB_INIT (FAB=.FAB_PTR, FAC=GET, FNA=.NOUN_NODE[DBG$L_NOUN_VALUE] + 1,
```

```
;  386        P 0516  2                FNS=.(.NOUN_NODE [DBG$L_NOUN_VALUE]) <0, 8, 0>,
;  387          0517  2                DNA=DEFICF_NAME, DNS=DEFICF_SIZE);
;  388          0518  2      $RAB_INIT (RAB=.RAB_PTR, FAB=.FAB_PTR);
;  389          0519  2
;  390          0520  2
;  391          0521  2      ! Place the ICF on the command input stream.
;  392          0522  2      !
;  393          0523  2      IF NOT DBG$NCIS_ADD(.RAB_PTR, 0, CIS_RAB, 0, 0, 0, .MESSAGE_VECT)
;  394          0524  2      THEN
;  395          0525  2          RETURN STS$K_SEVERE;
;  396          0526  2
;  397          0527  2
;  398          0528  2      ! Set up the local define list for the procedure.
;  399          0529  2      !
;  400          0530  2      IF NOT DBG$DEF_PR_ENTRY(.MESSAGE_VECT)
;  401          0531  2      THEN
;  402          0532  2          RETURN STS$K_SEVERE;
;  403          0533  2
;  404          0534  2
;  405          0535  2      ! Add the parameters to the local define list.  To do this, we loop through
;  406          0536  2      ! the remaining Noun Nodes to get the parameter pointers.
;  407          0537  2      !
;  408          0538  2      PARAM_COUNT = 0;
;  409          0539  2      NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
;  410          0540  2      WHILE .NOUN_NODE NEQ 0 DO
;  411          0541  3          BEGIN
;  412          0542  3
;  413          0543  3
;  414          0544  3          ! Increment the parameter count to give the parameter number of the
;  415          0545  3          ! present parameter.  Then obtain the name for this parameter and add
;  416          0546  3          ! the parameter to the parameter list for the current command file.
;  417          0547  3          !
;  418          0548  3          PARAM_COUNT = .PARAM_COUNT + 1;
;  419          0549  3          IF NOT DBG$CONSTRUCT_PARAM_NAME(.PARAM_COUNT, PARAM_NAME, TRUE,
;  420          0550  3                                                          .MESSAGE_VECT)
;  421          0551  3          THEN
;  422          0552  3              RETURN STS$K_SEVERE;
;  423          0553  3
;  424          0554  3          IF NOT DBG$DEF_SYM_ADD(.PARAM_NAME, DEFINE_PARAMETER,
;  425          0555  3                  .NOUN_NODE[DBG$L_NOUN_VALUE], FALSE,
;  426          0556  3                  REPLACED_FLAG, .MESSAGE_VECT)
;  427          0557  3          THEN
;  428          0558  3              RETURN STS$K_SEVERE;
;  429          0559  3
;  430          0560  3          IF .REPLACED_FLAG THEN $DBG_ERROR('DBGATSIGN\DBG$NEXECUTE_AT_SIGN');
;  431          0561  3
;  432          0562  3
;  433          0563  3          ! Get the Noun Node for the next parameter and loop.
;  434          0564  3          !
;  435          0565  3          NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
;  436          0566  2          END;
;  437          0567  2
;  438          0568  2
;  439          0569  2      ! Build a Value Descriptor to hold the integer value that gives the number
;  440          0570  2      ! of parameters to the indirect command file we are opening.  Then create
;  441          0571  2      ! a local DEFINEd symbol called %PARCNT in the scope of this indirect com-
;  442          0572  2      ! mand file.  This symbol gives the number of parameters to the indirect
```

```
;  443          0573  2     ! command file when referenced by the user within the command file.
;  444          0574  2     !
;  445          0575  2     VALPTR = DBG$GET_MEMORY(DBG$K_VALDESC_BASE_SIZE + 4);
;  446          0576  2     VALPTR[DBG$B_DHDR_TYPE] = DBG$K_VALUE_DESC;
;  447          0577  2     VALPTR[DBG$W_DHDR_LENGTH] = (DBG$K_VALDESC_BASE_SIZE + 4)*%UPVAL;
;  448          0578  2     VALPTR[DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
;  449          0579  2     VALPTR[DBG$B_DHDR_KIND] = RST$K_DATA;
;  450          0580  2     VALPTR[DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;
;  451          0581  2     VALPTR[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
;  452          0582  2     VALPTR[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
;  453          0583  2     VALPTR[DBG$W_VALUE_LENGTH] = 4;
;  454          0584  2     VALPTR[DBG$L_VALUE_POINTER] = VALPTR[DBG$A_VALUE_ADDRESS];
;  455          0585  2     VALARRAY = VALPTR[DBG$A_VALUE_ADDRESS];
;  456          0586  2     VALARRAY[0] = .PARAM_COUNT;
;  457          0587  2     PARAM_NAME = DBG$GET_MEMORY(2);
;  458          0588  2     CH$MOVE(8, UPLIT BYTE(%ASCIC '%PARCNT'), PARAM_NAME[0]);
;  459          0589  2     IF NOT DBG$DEF_SYM_ADD(.PARAM_NAME, DEFINE_VALUE, .VALPTR,
;  460          0590  2                                            FALSE, REPLACED_FLAG, .MESSAGE_VECT)
;  461          0591  2     THEN
;  462          0592  2         RETURN STS$K_SEVERE;
;  463          0593  2
;  464          0594  2
;  465          0595  2     ! Open the specified indirect command file as the current input file.
;  466          0596  2     ! Then return.
;  467          0597  2     !
;  468          0598  2     IF NOT DBG$NCIS_OPENICF(.MESSAGE_VECT) THEN RETURN STS$K_SEVERE;
;  469          0599  2     RETURN STS$K_SUCCESS;
;  470          0600  2
;  471          0601  1     END;


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

                                 4D 4F 43 2E 47 55 42 45 44  00008 P.AAE:  .ASCII  \DEBUG.COM\                                    ;
24 47 42 44 5C 4E 47 49 53 54 41 47 42 44 1E  00011 P.AAF:  .ASCII  <30>\DBGATSIGN\<92>\DBG$NEXECUTE_AT_SI\                    ;
   49 53 5F 54 41 5F 45 54 55 43 45 58 45 4E  00020                                                                           ;
                                             4E 47  0002E          .ASCII  \GN\                                               ;
                              54 4E 43 52 41 50 25 07  00030 P.AAG:  .ASCII  <7>\%PARCNT\                                       ;

                                                                DEFICF_NAME=        P.AAE
                                                                DEFICF_SIZE=        9


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

                                             0FFC 00000         .ENTRY  DBG$NEXECUTE_AT_SIGN, Save R2,R3,R4,R5,R6,-              ; 0459
                                                                        R7,R8,R9,R10,R11                                        ;
                                 5B 00000000G   00  9E 00002    MOVAB   DBG$DEF_SYM_ADD, R11                                    ;
                                 5A 00000000'   EF  9E 00009    MOVAB   DEFICF_NAME, R10                                        ;
                                 59 00000000G   00  9E 00010    MOVAB   DBG$GET_MEMORY, R9                                      ;
                                 5E             08  C2 00017    SUBL2   #8, SP                                                  ;
                                 50       04    AC  D0 0001A    MOVL    VERB_NODE, R0                                           ; 0508
                                 58       08    A0  D0 0001E    MOVL    8(R0), NOUN_NODE                                        ;
                                                14  DD 00022    PUSHL   #20                                                     ; 0513
                                 69             01  FB 00024    CALLS   #1, DBG$GET_MEMORY                                      ;
                                 56             50  D0 00027    MOVL    R0, FAB_PTR                                             ;
```

```
                                                    11 DD 0002A          PUSHL   #17                                                    ; 0514
                                                69  01 FB 0002C          CALLS   #1, DBG$GET_MEMORY
                                                57  50 D0 0002F          MOVL    R0, RAB_PTR
    0050   8F            00                    6E  00 2C 00032          MOVC5   #0, (SP), #0, #80, (FAB_PTR)                           ; 0517
                                                66  00039
                                                66  5003  8F B0 0003A    MOVW    #20483, (FAB_PTR)
                                          16    A6  02 90 0003F          MOVB    #2, 22(FAB_PTR)
                                          1F    A6  02 90 00043          MOVB    #2, 31(FAB_PTR)
                                    2C    A6        68  01 C1 00047      ADDL3   #1, (NOUN_NODE), 44(FAB_PTR)
                                          30    A6  6A 9E 0004C          MOVAB   DEFICF_NAME, 48(FAB_PTR)
                                          34    A6    00  B8 90 00050    MOVB    @0(NOUN_NODE), 52(FAB_PTR)
                                          35    A6  09 90 00055          MOVB    #9, 53(FAB_PTR)
    0044   8F            00                    6E  00 2C 00059          MOVC5   #0, (SP), #0, #68, (RAB_PTR)                           ; 0518
                                                67  00060
                                                67  4401  8F B0 00061    MOVW    #17409, (RAB_PTR)
                                          3C    A7  56 D0 00066          MOVL    FAB_PTR, 60(RAB_PTR)
                                                56  08  AC D0 0006A      MOVL    MESSAGE_VECT, R6                                       ; 0523
                                                    56 DD 0006E          PUSHL   R6
                                                    7E 7C 00070          CLRQ    -(SP)
                                                7E      01 7D 00072      MOVQ    #1, -(SP)
                                                    7E D4 00075          CLRL    -(SP)
                                                    57 DD 00077          PUSHL   RAB_PTR
                                  00000000G     00    07 FB 00079        CALLS   #7, DBG$NCIS_ADD
                                                38    50 E9 00080        BLBC    R0, 2$
                                                    56 DD 00083          PUSHL   R6                                                     ; 0530
                                  00000000G     00    01 FB 00085        CALLS   #1, DBG$DEF_PR_ENTRY
                                                2C    50 E9 0008C        BLBC    R0, 2$
                                                    52 D4 0008F          CLRL    PARAM_COUNT                                            ; 0538
                                                58  08  A8 D0 00091  1$:  MOVL    8(NOUN_NODE), NOUN_NODE                                ; 0539
                                                    3F 13 00095          BEQL    3$                                                     ; 0540
                                                    52 D6 00097          INCL    PARAM_COUNT                                            ; 0548
                                                    56 DD 00099          PUSHL   R6                                                     ; 0550
                                                    01 DD 0009B          PUSHL   #1                                                     ; 0549
                                                    08 AE 9F 0009D       PUSHAB  PARAM_NAME
                                                    52 DD 000A0          PUSHL   PARAM_COUNT
                                      0000V     CF      04 FB 000A2      CALLS   #4, DBG$CONSTRUCT_PARAM_NAME
                                                7C      50 E9 000A7      BLBC    R0, 4$
                                                    56 DD 000AA          PUSHL   R6                                                     ; 0556
                                                    08 AE 9F 000AC       PUSHAB  REPLACED_FLAG                                          ; 0554
                                                    7E D4 000AF          CLRL    -(SP)
                                                    68 DD 000B1          PUSHL   (NOUN_NODE)                                            ; 0555
                                                    06 DD 000B3          PUSHL   #6                                                     ; 0554
                                                    14 AE DD 000B5       PUSHL   PARAM_NAME
                                                6B      06 FB 000B8      CALLS   #6, DBG$DEF_SYM_ADD
                                                77      50 E9 000BB  2$:  BLBC    R0, 5$
                                                CF      04 AE E9 000BE   BLBC    REPLACED_FLAG, 1$                                      ; 0560
                                                    09 AA 9F 000C2       PUSHAB  P.AAF
                                                    01 DD 000C5          PUSHL   #1
                                            00028362    8F DD 000C7      PUSHL   #164706
                                  00000000G     00      03 FB 000CD      CALLS   #3, LIB$SIGNAL
                                                    BB 11 000D4          BRB     1$                                                     ; 0565
                                                    0C DD 000D6  3$:     PUSHL   #12                                                    ; 0575
                                                69      01 FB 000D8      CALLS   #1, DBG$GET_MEMORY
                                                57      50 D0 000DB      MOVL    R0, VALPTR
                                          02    A7        7A  8F 90 000DE  MOVB    #122, 2(VALPTR)                                        ; 0576
                                                67      30 B0 000E3      MOVW    #48, (VALPTR)                                          ; 0577
                                          06    A7      0602  8F B0 000E6  MOVW    #1538, 6(VALPTR)                                       ; 0578
```

```
                                03    A7 00000000G   00    90 000EC          MOVB    DBG$GB_LANGUAGE, 3(VALPTR)                ; 0580
                                14    A7 01080004    8F    D0 000F4          MOVL    #17301508, 20(VALPTR)                     ; 0583
                                      50       20    A7    9E 000FC          MOVAB   32(R7), R0                                ; 0584
                                18    A7             50    D0 00100          MOVL    R0, 24(VALPTR)                            ;
                                      60             52    D0 00104          MOVL    PARAM_COUNT, (VALARRAY)                   ; 0586
                                                     02    DD 00107          PUSHL   #2                                        ; 0587
                                      69             01    FB 00109          CALLS   #1, DBG$GET_MEMORY                        ;
                                      6E             50    D0 0010C          MOVL    R0, PARAM_NAME                            ;
                     00   BE        28    AA       08    28 0010F          MOVC3   #8, P.AAG, @PARAM_NAME                    ; 0588
                                                     56    DD 00115          PUSHL   R6                                        ; 0590
                                            08       AE    9F 00117          PUSHAB  REPLACED_FLAG                             ; 0589
                                                     7E    D4 0011A          CLRL    -(SP)                                     ;
                                                     57    DD 0011C          PUSHL   VALPTR                                    ;
                                                     05    DD 0011E          PUSHL   #5                                        ;
                                            14       AE    DD 00120          PUSHL   PARAM_NAME                                ;
                                      6B             06    FB 00123          CALLS   #6, DBG$DEF_SYM_ADD                       ;
                                      0C             50    E9 00126 4$:      BLBC    R0, 5$                                    ;
                                                     56    DD 00129          PUSHL   R6                                        ; 0598
                          00000000G   00             01    FB 0012B          CALLS   #1, DBG$NCIS_OPENICF                      ;
                                      04             50    E8 00132          BLBS    R0, 6$                                    ;
                                      50             04    D0 00135 5$:      MOVL    #4, R0                                    ;
                                                           04 00138          RET                                               ;
                                      50             01    D0 00139 6$:      MOVL    #1, R0                                    ; 0599
                                                           05 0013C          RET                                               ; 0601

; Routine Size:  317 bytes,    Routine Base:  DBG$CODE + 00ED
```

```
;  473     0602  1 GLOBAL ROUTINE DBG$NPARSE_ATTACH (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;  474     0603  1 !
;  475     0604  1 ! FUNCTION
;  476     0605  1 !       Parses the ATTACH command. The "ATTACH" has already been
;  477     0606  1 !       recognized by the top-level parse routine. This routine picks
;  478     0607  1 !       up the integer argument and constructs a command execution tree.
;  479     0608  1 !       The tree has the verb node for ATTACH, and a noun node with
;  480     0609  1 !       the integer argument.
;  481     0610  1 !
;  482     0611  1 ! INPUTS
;  483     0612  1 !       INPUT_DESC      - A string descriptor for the remaining command line
;  484     0613  1 !
;  485     0614  1 !       VERB_NODE       - The already existing verb node.
;  486     0615  1 !
;  487     0616  1 !       MESSAGE_VECT    - The address of a message argument vector.
;  488     0617  1 !
;  489     0618  1 ! OUTPUTS
;  490     0619  1 !       The return value is one of:
;  491     0620  1 !               STS$K_SUCCESS   - Success. A command execution tree was
;  492     0621  1 !                                 constructed.
;  493     0622  1 !               STS$K_SEVERE    - Failure. An error message vector is
;  494     0623  1 !                                 constructed.
;  495     0624  1 !
;  496     0625  1
;  497     0626  2     BEGIN
;  498     0627  2
;  499     0628  2     MAP
;  500     0629  2         INPUT_DESC: REF DBG$STG_DESC,
;  501     0630  2         VERB_NODE: REF DBG$VERB_NODE;
;  502     0631  2
;  503     0632  2     BIND
;  504     0633  2         DEG$CS_CR       = UPLIT BYTE (1, DBG$K_CAR_RETURN);
;  505     0634  2
;  506     0635  2
;  507     0636  2
;  508     0637  2     ! Check for end-of-line. This is an error.
;  509     0638  2     !
;  510     0639  2     IF .INPUT_DESC [DSC$W_LENGTH] EQL 0
;  511     0640  2     THEN
;  512     0641  3         BEGIN
;  513     0642  3         .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE);
;  514     0643  3         RETURN STS$K_SEVERE;
;  515     0644  2         END;
;  516     0645  2
;  517     0646  2     IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
;  518     0647  2     THEN
;  519     0648  3         BEGIN
;  520     0649  3         .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE);
;  521     0650  3         RETURN STS$K_SEVERE;
;  522     0651  2         END;
;  523     0652  2
;  524     0653  2
;  525     0654  2     ! Pick up the string with the process name. Put a pointer to the
;  526     0655  2     ! string right in the verb node.
;  527     0656  2     !
;  528     0657  2     IF NOT DBG$NSAVE_STRING(.INPUT_DESC,
;  529     0658  2                 VERB_NODE [DBG$L_VERB_ADVERB_PTR], .MESSAGE_VECT)
```

```
;  530          0659  2     THEN
;  531          0660  2         RETURN STS$K_SEVERE;
;  532          0661  2
;  533          0662  2
;  534          0663  2     ! Return success.
;  535          0664  2     !
;  536          0665  2     RETURN STS$K_SUCCESS;
;  537          0666  1     END;


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

                                        0D  01  00038 P.AAH:    .BYTE   1, 13                                   ;

                                                      DBG$CS_CR=        P.AAH


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

                                          0004 00000            .ENTRY  DBG$NPARSE_ATTACH, Save R2               ; 0602
                          52        04    AC  D0 00002          MOVL    INPUT_DESC, R2                           ; 0639
                                          62  B5 00006          TSTW    (R2)                                     ;
                                          14  13 00008          BEQL    1$                                       ;
                                          01  DD 0000A          PUSHL   #1                                       ; 0646
                             00000000'    EF  9F 0000C          PUSHAB  DBG$CS_CR                                ;
                                          52  DD 00012          PUSHL   R2                                       ;
                    00000000G   00        03  FB 00014          CALLS   #3, DBG$NMATCH                           ;
                                13        50  E9 0001B          BLBC    R0, 2$                                   ;
                                     00028DD0   8F  DD 0001E 1$:  PUSHL   #164048                                 ; 0649
                    00000000G   00        01  FB 00024          CALLS   #1, DBG$NMAKE_ARG_VECT                   ;
                          0C    BC          50  D0 0002B          MOVL    R0, @MESSAGE_VECT                        ;
                                            14  11 0002F          BRB     3$                                       ; 0650
                                      0C    AC  DD 00031 2$:     PUSHL   MESSAGE_VECT                             ; 0658
                    7E          08    AC    04  C1 00034          ADDL3   #4, VERB_NODE, -(SP)                     ;
                                            52  DD 00039          PUSHL   R2                                       ;
                    00000000G   00        03  FB 0003B          CALLS   #3, DBG$NSAVE_STRING                     ;
                                04        50  E8 00042          BLBS    R0, 4$                                   ;
                                50        04  D0 00045 3$:      MOVL    #4, R0                                   ; 0660
                                            04 00048            RET                                              ;
                                50        01  D0 00049 4$:      MOVL    #1, R0                                   ; 0665
                                            04 0004C            RET                                              ; 0666

; Routine Size:  77 bytes,    Routine Base:  DBG$CODE + 022A
```

```
;  539          0667  1 GLOBAL ROUTINE DBG$NEXECUTE_ATTACH (VERB_NODE, MESSAGE_VECT) =
;  540          0668  1 !
;  541          0669  1 ! FUNCTIONAL DESCRIPTION:
;  542          0670  1 !
;  543          0671  1 !       This routine executes an ATTACH command.
;  544          0672  1 !
;  545          0673  1 ! FORMAL PARAMETERS:
;  546          0674  1 !
;  547          0675  1 !       verb_node -                     The command verb node which is the start
;  548          0676  1 !                                       of the command execution tree
;  549          0677  1 !
;  550          0678  1 !       message_vect -                  The address of a longword to contain the
;  551          0679  1 !                                       address of a message argument vector
;  552          0680  1 !
;  553          0681  1 ! IMPLICIT INPUTS:
;  554          0682  1 !
;  555          0683  1 !       NONE
;  556          0684  1 !
;  557          0685  1 ! IMPLICIT OUTPUTS:
;  558          0686  1 !
;  559          0687  1 !       On failure, a message argument vector is returned
;  560          0688  1 !
;  561          0689  1 ! ROUTINE VALUE:
;  562          0690  1 !
;  563          0691  1 !       An unsigned integer longword completion code
;  564          0692  1 !
;  565          0693  1 ! COMPLETION CODES:
;  566          0694  1 !
;  567          0695  1 !       sts$k_success (1) -     Indicates the file was created and connected
;  568          0696  1 !
;  569          0697  1 !       sts$k_severe (4) -      Failure, file not created. Message argument vector returned
;  570          0698  1 !
;  571          0699  1 ! SIDE EFFECTS:
;  572          0700  1 !
;  573          0701  1 !
;  574          0702  1
;  575          0703  2     BEGIN
;  576          0704  2
;  577          0705  2     MAP
;  578          0706  2         VERB_NODE: REF DBG$VERB_NODE;
;  579          0707  2
;  580          0708  2     LOCAL
;  581          0709  2         EVNT_FLAG,                          ! Event flag number
;  582          0710  2         ITEM: BLOCK [3,LONG],               ! Item list for $GETJPI
;  583          0711  2         PID,                                ! Process id
;  584          0712  2         PNAME: REF VECTOR[,BYTE],           ! Pointer to counted string
;  585          0713  2                                             !    with process name
;  586          0714  2         STATUS,                             ! Return status from LIB$ATTACH
;  587          0715  2         STG_DESC: BLOCK[8,BYTE],            ! String descriptor with name
;  588          0716  2         X;
;  589          0717  2
;  590          0718  2
;  591          0719  2     ! Obtain the process name of the process we are to attach to.
;  592          0720  2     !
;  593          0721  2     PID = 0;
;  594          0722  2     PNAME = .VERB_NODE [DBG$L_VERB_ADVERB_PTR];
;  595          0723  2     STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
```

```
;  596          0724  2      STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  597          0725  2      STG_DESC[DSC$W_LENGTH] = .PNAME[0];
;  598          0726  2      STG_DESC[DSC$A_POINTER] = PNAME[1];
;  599          0727  2
;  600          0728  2
;  601          0729  2      ! Obtain the process id of the process we are to attach to.
;  602          0730  2      !
;  603          0731  2      STATUS = LIB$GET_EF(EVNT_FLAG);
;  604          0732  2      IF NOT .STATUS THEN SIGNAL(.STATUS);
;  605          0733  2      CH$FILL(0, 12, ITEM);
;  606          0734  2      STATUS = $GETJPI(EFN=.EVNT_FLAG, PIDADR=PID, PRCNAM=STG_DESC, ITMLST=ITEM);
;  607          0735  2      IF NOT .STATUS
;  608          0736  2      THEN
;  609          0737  3          BEGIN
;  610          0738  3          LIB$FREE_EF(EVNT_FLAG);
;  611          0739  3          SIGNAL(DBG$_GETJPI, 0, .STATUS);
;  612          0740  2          END;
;  613          0741  2
;  614          0742  2      $WAITFR(EFN = .EVNT_FLAG);
;  615          0743  2      LIB$FREE_EF(EVNT_FLAG);
;  616          0744  2
;  617          0745  2
;  618          0746  2      ! Reset the terminal screen to "normal" if we are in screen mode.  This
;  619          0747  2      ! makes sure that the whole terminal screen scrolls when we are in the
;  620          0748  2      ! other process and that the screen is  efreshed when we reenter DEBUG.
;  621          0749  2      !
;  622          0750  2      DBG$SCR_SCREEN_NORMAL();
;  623          0751  2
;  624          0752  2
;  625          0753  2      ! Signal that we are about to do an attach.  Do the attach.  Signal an
;  626          0754  2      ! error if it fails.
;  627          0755  2      !
;  628          0756  2      SIGNAL(DBG$_ATTACHED, 1, .PNAME);
;  629          0757  2      STATUS = LIB$ATTACH(PID);
;  630          0758  2      IF NOT .STATUS THEN SIGNAL(DBG$_NOATTACH, 0, .STATUS);
;  631          0759  2
;  632          0760  2      ! After returning from the attach, set the terminal back to
;  633          0761  2      ! the appropriate mode. Then return.
;  634          0762  2      !
;  635          0763  2      IF .DBG$GB_KEYPAD_INPUT THEN X = 1 ELSE X = 0;
;  636          0764  2      SMG$SET_KEYPAD_MODE (DBG$GL_KEYBOARD_ID, X);
;  637          0765  2      RETURN STS$K_SUCCESS;
;  638          0766  1      END;


                                                .EXTRN  SYS$GETJPI, SYS$WAITFR

                                 03FC 00000     .ENTRY  DBG$NEXECUTE_ATTACH, Save R2,R3,R4,R5,R6,-  ; 0667
                                                        R7,R8,R9                                    ;
            59 00000000G   00    9E 00002     MOVAB   LIB$FREE_EF, R9                             ;
            58 00000000G   00    9E 00009     MOVAB   LIB$SIGNAL, R8                              ;
            5E             20    C2 00010     SUBL2   #32, SP                                     ;
                     04    AE    D4 00013     CLRL    PID                                         ; 0721
            50       04    AC    D0 00016     MOVL    VERB_NODE, R0                               ; 0722
            57       04    A0    D0 0001A     MOVL    4(R0), PNAME                                ;
         0E AE     010E    8F    B0 0001E     MOVW    #270, STG_DESC+2                            ; 0724
```

```
                        0C   AE          67  9B 00024          MOVZBW  (PNAME), STG_DESC                ; 0725
                        10   AE    01    A7  9E 00028          MOVAB   1(R7), STG_DESC+4                ; 0726
                                          5E  DD 0002D          PUSHL   SP                               ; 0731
                00000000G   00            01  FB 0002F          CALLS   #1, LIB$GET_EF                   ;
                            56            50  D0 00036          MOVL    R0, STATUS                       ;
                            05            56  E8 00039          BLBS    STATUS, 1$                       ; 0732
                                          56  DD 0003C          PUSHL   STATUS                           ;
                            68            01  FB 0003E          CALLS   #1, LIB$SIGNAL                   ;
        0C          00      6E            00  2C 00041  1$:     MOVC5   #0, (SP), #0, #12, ITEM          ; 0733
                                    14    AE      00046                                                  ;
                                          7E  7C 00048          CLRQ    -(SP)                            ; 0734
                                          7E  D4 0004A          CLRL    -(SP)                            ;
                                    20    AE  9F 0004C          PUSHAB  ITEM                             ;
                                    1C    AE  9F 0004F          PUSHAB  STG_DESC                         ;
                                    18    AE  9F 00052          PUSHAB  PID                              ;
                                    18    AE  DD 00055          PUSHL   EVNT_FLAG                        ;
                00000000G   00            07  FB 00058          CALLS   #7, SYS$GETJPI                   ;
                            56            50  D0 0005F          MOVL    R0, STATUS                       ;
                            12            56  E8 00062          BLBS    STATUS, 2$                       ; 0735
                                          5E  DD 00065          PUSHL   SP                               ; 0738
                            69            01  FB 00067          CALLS   #1, LIB$FREE_EF                  ;
                                          56  DD 0006A          PUSHL   STATUS                           ; 0739
                                          7E  D4 0006C          CLRL    -(SP)                            ;
                                00028A2A  8F  DD 0006E          PUSHL   #166442                          ;
                            68            03  FB 00074          CALLS   #3, LIB$SIGNAL                   ;
                                          6E  DD 00077  2$:     PUSHL   EVNT_FLAG                        ; 0742
                00000000G   00            01  FB 00079          CALLS   #1, SYS$WAITFR                   ;
                                          5E  DD 00080          PUSHL   SP                               ; 0743
                            69            01  FB 00082          CALLS   #1, LIB$FREE_EF                  ;
                00000000G   00            00  FB 00085          CALLS   #0, DBG$SCR_SCREEN_NORMAL        ; 0750
                                          57  DD 0008C          PUSHL   PNAME                            ; 0756
                                          01  DD 0008E          PUSHL   #1                               ;
                                00286FB   8F  DD 00090          PUSHL   #165627                          ;
                            68            03  FB 00096          CALLS   #3, LIB$SIGNAL                   ;
                                    04    AE  9F 00099          PUSHAB  PID                              ; 0757
                00000000G   00            01  FB 0009C          CALLS   #1, LIB$ATTACH                   ;
                            56            50  D0 000A3          MOVL    R0, STATUS                       ;
                            0D            56  E8 000A6          BLBS    STATUS, 3$                       ; 0758
                                          56  DD 000A9          PUSHL   STATUS                           ;
                                          7E  D4 000AB          CLRL    -(SP)                            ;
                                00028A22  8F  DD 000AD          PUSHL   #166434                          ;
                            68            03  FB 000B3          CALLS   #3, LIB$SIGNAL                   ;
                            06  00000000G   00  E9 000B6  3$:     BLBC    DBG$GB_KEYPAD_INPUT, 4$          ; 0763
                    08      AE            01  D0 000BD          MOVL    #1, X                            ;
                                          03  11 000C1          BRB     5$                               ;
                                    08    AE  D4 000C3  4$:     CLRL    X                                ;
                                    08    AE  9F 000C6  5$:     PUSHAB  X                                ; 0764
                                00000000G   00  9F 000C9          PUSHAB  DBG$GL_KEYBOARD_ID               ;
                00000000G   00            02  FB 000CF          CALLS   #2, SMG$SET_KEYPAD_MODE          ;
                            50            01  D0 000D6          MOVL    #1, R0                           ; 0765
                                              04 000D9          RET                                      ; 0766

; Routine Size:  218 bytes,    Routine Base:  DBG$CODE + 0277
```

```
;  640          0767  1 GLOBAL ROUTINE DBG$NPARSE_SPAWN (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;  641          0768  1 !
;  642          0769  1 ! FUNCTION
;  643          0770  1 !       Parses the spawn command. The "SPAWN" has already been recognized in the routine
;  644          0771  1 !       dbg$nparse_cmd network. This routine merely aquires the argument string
;  645          0772  1 !       and causes the noun node ( noun value ) field to point to the argument which
;  646          0773  1 !       is stored as a counted string in dynamically allocated storage.
;  647          0774  1 !
;  648          0775  1 ! FORMAL PARAMETERS:
;  649          0776  1 !
;  650          0777  1 !       INPUT_DESC - The present command line input descriptor.
;  651          0778  1 !
;  652          0779  1 !       VERB_NODE  - The already existing command Verb Node.
;  653          0780  1 !
;  654          0781  1 !       MESSAGE_VECT - The address of a longword to contain the address of a
;  655          0782  1 !                 message argument vector.
;  656          0783  1 !
;  657          0784  1 !
;  658          0785  1 ! ROUTINE VALUE:
;  659          0786  1 !       An unsigned integer longword completion code:
;  660          0787  1 !
;  661          0788  1 !               STS$K_SEVERE (4) - Error in parsing
;  662          0789  1 !
;  663          0790  1 !               STS$K_SUCCESS (1) - Successful parse
;  664          0791  1 !
;  665          0792  1
;  666          0793  2     BEGIN
;  667          0794  2
;  668          0795  2     MAP
;  669          0796  2         INPUT_DESC: REF DBG$STG_DESC,
;  670          0797  2         VERB_NODE: REF DBG$VERB_NODE;
;  671          0798  2
;  672          0799  2     BIND
;  673          0800  2         DBG$CS_NOWAIT           = UPLIT BYTE (6, 'NOWAIT'),
;  674          0801  2         DBG$CS_CR               = UPLIT BYTE (1, DBG$K_CAR_RETURN),
;  675          0802  2         DBG$CS_DBLQUOTE         = UPLIT BYTE (1, DBG$K_DBLQUOTE),
;  676          0803  2         DBG$CS_SLASH            = UPLIT BYTE (1, DBG$K_SLASH);
;  677          0804  2
;  678          0805  2     LOCAL
;  679          0806  2         CHAR: BYTE,                             ! Holds a single character
;  680          0807  2         COUNT,                                  ! Count of number of chars in arg
;  681          0808  2         COUNTED_STRING: REF VECTOR[,BYTE],      ! Points to a counted string
;  682          0809  2                                                 !     holding the argument.
;  683          0810  2         FLAGS: BITVECTOR[32],                   ! Flags for /NOWAIT etc
;  684          0811  2         NOUN_NODE: REF DBG$NOUN_NODE,           ! A pointer to a noun node
;  685          0812  2         OUTPUT_PTR,                             ! Pointer into counted string
;  686          0813  2         PREV_CHAR: BYTE,                        ! Holds a single character
;  687          0814  2         QUOTE_FLAG: BYTE;                       ! TRUE if argument is quoted
;  688          0815  2
;  689          0816  2
;  690          0817  2
;  691          0818  2     ! Initialize flags.
;  692          0819  2     !
;  693          0820  2     FLAGS = 0;
;  694          0821  2
;  695          0822  2
;  696          0823  2     ! Collect qualifiers.
```

```
;  697          0824  2      !
;  698          0825  2      WHILE DBG$NMATCH (.INPUT_DESC, DBG$CS_SLASH, 1) DO
;  699          0826  2          BEGIN
;  700          0827  3          SELECTONE TRUE OF
;  701          0828  3              SET
;  702          0829  3
;  703          0830  3              ! /NOWAIT
;  704          0831  3              !
;  705          0832  3              [DBG$NMATCH (.INPUT_DESC, DBG$CS_NOWAIT, 3)]:
;  706          0833  3                  FLAGS[NOWAIT_FLAG] = TRUE;
;  707          0834  3
;  708          0835  3              [DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)]:
;  709          0836  3                  SIGNAL(DBG$_NEEDMORE);
;  710          0837  3
;  711          0838  3              [OTHERWISE]:
;  712          0839  3                  SIGNAL(DBG$_ILLQUALIF);
;  713          0840  3
;  714          0841  3              TES;
;  715          0842  2          END;
;  716          0843  2
;  717          0844  2
;  718          0845  2      ! Put the flags in the adverb field.
;  719          0846  2      !
;  720          0847  2      VERB_NODE[DBG$L_VERB_ADVERB_PTR] = .FLAGS;
;  721          0848  2
;  722          0849  2
;  723          0850  2      ! Check for end-of-line. If so, there was no argument to SPAWN.
;  724          0851  2      ! We do not allocate a noun node in this case. Just return success.
;  725          0852  2      !
;  726          0853  2      IF .INPUT_DESC [DSC$W_LENGTH] EQL 0 THEN RETURN STS$K_SUCCESS;
;  727          0854  2      IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1) THEN RETURN STS$K_SUCCESS;
;  728          0855  2
;  729          0856  2
;  730          0857  2      ! Create and link a Noun Node.
;  731          0858  2      !
;  732          0859  2      NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE);
;  733          0860  2      VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;
;  734          0861  2
;  735          0862  2
;  736          0863  2      ! Check whether the user has enclosed the argument in quotes.
;  737          0864  2      !
;  738          0865  2      IF DBG$NMATCH(.INPUT_DESC, DBG$CS_DBLQUOTE, 1)
;  739          0866  2      THEN
;  740          0867  2          QUOTE_FLAG = TRUE
;  741          0868  2
;  742          0869  2      ELSE
;  743          0870  2          QUOTE_FLAG = FALSE;
;  744          0871  2
;  745          0872  2
;  746          0873  2      ! Allocate space to store the argument string.
;  747          0874  2      !
;  748          0875  2      COUNTED_STRING = DBG$GET_TEMPMEM(1 + (1 + .INPUT_DESC[DSC$W_LENGTH])/4);
;  749          0876  2
;  750          0877  2
;  751          0878  2      ! Initialize some variables.
;  752          0879  2      !
;  753          0880  2      CHAR = 0;
```

```
;  754          0881  2      COUNT = 0;
;  755          0882  2      OUTPUT_PTR = COUNTED_STRING[1];
;  756          0883  2
;  757          0884  2
;  758          0885  2      ! Loop through the input, copying characters into the counted string.
;  759          0886  2      !
;  760          0887  2      WHILE .INPUT_DESC[DSC$W_LENGTH] GTR 0 DO
;  761          0888  3          BEGIN
;  762          0889  3
;  763          0890  3
;  764          0891  3          ! Get the next character.
;  765          0892  3          !
;  766          0893  3          PREV_CHAR = .CHAR;
;  767          0894  3          CHAR = CH$RCHAR_A (INPUT_DESC [DSC$A_POINTER]);
;  768          0895  3          INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
;  769          0896  3
;  770          0897  3
;  771          0898  3          ! Check for carriage return.
;  772          0899  3          !
;  773          0900  3          IF .CHAR EQL DBG$K_CAR_RETURN THEN EXITLOOP;
;  774          0901  3
;  775          0902  3
;  776          0903  3          ! If argument was quoted, check for either doubled quotes or
;  777          0904  3          ! closing quote.
;  778          0905  3          !
;  779          0906  3          IF .QUOTE_FLAG
;  780          0907  3          THEN
;  781          0908  4              BEGIN
;  782          0909  4              IF .CHAR EQL DBG$K_DBLQUOTE
;  783          0910  4              THEN
;  784          0911  5                  BEGIN
;  785          0912  5
;  786          0913  5
;  787          0914  5                  ! Undouble double quotes if that is what we have.  Otherwise,
;  788          0915  5                  ! we exit the quoted-string loop on the clonsing quote.
;  789          0916  5                  !
;  790          0917  5                  IF .PREV_CHAR NEQ DBG$K_DBLQUOTE THEN EXITLOOP;
;  791          0918  5                  CHAR = CH$RCHAR_A(.INPUT_DESC [DSC$A_POINTER]);
;  792          0919  5                  INPUT_DESC[DSC$W_LENGTH] = .INPUT_DESC[DSC$W_LENGTH] - 1;
;  793          0920  4                  END;
;  794          0921  4
;  795          0922  3              END;
;  796          0923  3
;  797          0924  3
;  798          0925  3          ! Write the character to the output buffer and increment count
;  799          0926  3          ! of number of chars in output buffer.
;  800          0927  3          !
;  801          0928  3          CH$WCHAR_A(.CHAR, OUTPUT_PTR);
;  802          0929  3          COUNT = .COUNT + 1;
;  803          0930  2          END;
;  804          0931  2
;  805          0932  2
;  806          0933  2      ! Fill in the count field.
;  807          0934  2      !
;  808          0935  2      COUNTED_STRING [0] = .COUNT;
;  809          0936  2
;  810          0937  2
```

```
;  811          0938  2     ! Fill in the Noun Node.
;  812          0939  2     !
;  813          0940  2     NOUN_NODE [DBG$L_NOUN_VALUE] = .COUNTED_STRING;
;  814          0941  2
;  815          0942  2
;  816          0943  2     ! Return success.
;  817          0944  2     !
;  818          0945  2     RETURN STS$K_SUCCESS;
;  819          0946  2
;  820          0947  1     END:


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

                                        06  0003A P.AAI:   .BYTE   6                                       ;
                              54 49 41 57 4F 4E 0003B       .ASCII  \NOWAIT\                                ;
                                        0D  01  00041 P.AAJ:   .BYTE   1, 13                              ;
                                        22  01  00043 P.AAK:   .BYTE   1, 34                              ;
                                        2F  01  00045 P.AAL:   .BYTE   1, 47                              ;

                                                DBG$CS_NOWAIT=      P.AAI
                                                DBG$CS_CR=          P.AAJ
                                                DBG$CS_DBLQUOTE=    P.AAK
                                                DBG$CS_SLASH=       P.AAL

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

                                    0FFC 00000          .ENTRY  DBG$NPARSE_SPAWN, Save R2,R3,R4,R5,R6,R7,-  ; 0767
                                                                R8,R9,R10,R11                               ;
5B 000000C0'G   00  9E  00002       MOVAB   DBG$GET_TEMPMEM, R11                                            ;
5A 00000000'    EF  9E  00009       MOVAB   DBG$CS_CR, R10                                                  ;
59 00000000G    00  9E  00010       MOVAB   DBG$NMATCH, R9                                                  ;
                54  D4  00017       CLRL    FLAGS                                                       ; 0820
53      04      AC  D0  00019       MOVL    INPUT_DESC, R3                                              ; 0825
                01  DD  0001D 1$:   PUSHL   #1                                                          ;
        04      AA  9F  0001F       PUSHAB  DBG$CS_SLASH                                                ;
                53  DD  00022       PUSHL   R3                                                          ;
69              03  FB  00024       CALLS   #3, DBG$NMATCH                                              ;
39              50  E9  00027       BLBC    R0, 5$                                                      ;
                03  DD  0002A       PUSHL   #3                                                          ; 0832
        F9      AA  9F  0002C       PUSHAB  DBG$CS_NOWAIT                                               ;
                53  DD  0002F       PUSHL   R3                                                          ;
69              03  FB  00031       CALLS   #3, DBG$NMATCH                                              ;
01              50  D1  00034       CMPL    R0, #1                                                      ;
                05  12  00037       BNEQ    2$                                                          ;
54              01  88  00039       BISB2   #1, FLAGS                                                   ; 0833
                DF  11  0003C       BRB     1$                                                          ;
                01  DD  0003E 2$:   PUSHL   #1                                                          ; 0835
      0408      8F  BB  00040       PUSHR   #^M<R3,R10>                                                 ;
69              03  FB  00044       CALLS   #3, DBG$NMATCH                                              ;
01              50  D1  00047       CMPL    R0, #1                                                      ;
                08  12  0004A       BNEQ    3$                                                          ;
  000280D0      8F  DD  0004C       PUSHL   #164048                                                     ; 0836
                06  11  00052       BRB     4$                                                          ;
  00028D20      8F  DD  00054 3$:   PUSHL   #167200                                                     ; 0839
```

```
00000000G  00        01  FB 0005A 4$:    CALLS   #1, LIB$SIGNAL                        ;
                     BA  11 00061        BRB     1$                                    ; 0825
           52    08  AC  D0 00063 5$:    MOVL    VERB_NODE, R2                         ; 0847
       04  A2        54  D0 00067        MOVL    FLAGS, 4(R2)                          ;
                     63  B5 0006B        TSTW    (R3)                                  ; 0853
                     03  12 0006D        BNEQ    6$                                    ;
                   0080  31 0006F        BRW     12$                                   ;
                     01  DD 00072 6$:    PUSHL   #1                                    ; 0854
               0408  8F  BB 00074        PUSHR   #^M<R3,R10>                           ;
           69        03  FB 00078        CALLS   #3, DBG$NMATCH                        ;
           74        50  E8 0007B        BLBS    R0, 12$                               ;
                     04  DD 0007E        PUSHL   #4                                    ; 0859
           6B        01  FB 00080        CALLS   #1, DBG$GET_TEMPMEM                   ;
           57        50  D0 00083        MOVL    R0, NOUN_NODE                         ;
       08  A2        57  D0 00086        MOVL    NOUN_NODE, 8(R2)                      ; 0860
                     01  DD 0008A        PUSHL   #1                                    ; 0865
                 02  AA  9F 0008C        PUSHAB  DBG$CS_DBLQUOTE                       ;
                     53  DD 0008F        PUSHL   R3                                    ;
           69        03  FB 00091        CALLS   #3, DBG$NMATCH                        ;
           05        50  E9 00094        BLBC    R0, 7$                                ;
           56        01  90 00097        MOVB    #1, QUOTE_FLAG                        ; 0867
                     02  11 0009A        BRB     8$                                    ;
                     56  94 0009C 7$:    CLRB    QUOTE_FLAG                            ; 0870
           50        63  3C 0009E 8$:    MOVZWL  (R3), R0                              ; 0875
                     50  D6 000A1        INCL    R0                                    ;
           50        04  C6 000A3        DIVL2   #4, R0                                ;
                 01  A0  9F 000A6        PUSHAB  1(R0)                                 ;
           6B        01  FB 000A9        CALLS   #1, DBG$GET_TEMPMEM                   ;
           54        50  D0 000AC        MOVL    R0, COUNTED_STRING                    ;
                     51  94 000AF        CLRB    CHAR                                  ; 0880
                     55  D4 000B1        CLRL    COUNT                                 ; 0881
           50    01  A4  9E 000B3        MOVAB   1(R4), OUTPUT_PTR                     ; 0882
                     63  B5 000B7 9$:    TSTW    (R3)                                  ; 0887
                     31  13 000B9        BEQL    11$                                   ;
           52        51  90 000BB        MOVB    CHAR, PREV_CHAR                       ; 0893
           51    04  B3  90 000BE        MOVB    @4(R3), CHAR                          ; 0894
                 04  A3  D6 000C2        INCL    4(R3)                                 ;
                     63  B7 000C5        DECW    (R3)                                  ; 0895
           0D        51  91 000C7        CMPB    CHAR, #13                             ; 0900
                     20  13 000CA        BEQL    11$                                   ;
           16        56  E9 000CC        BLBC    QUOTE_FLAG, 10$                       ; 0906
           22        51  91 000CF        CMPB    CHAR, #34                             ; 0909
                     11  12 000D2        BNEQ    10$                                   ;
           22        52  91 000D4        CMPB    PREV_CHAR, #34                        ; 0917
                     13  12 000D7        BNEQ    11$                                   ;
           58    04  B3  D0 000D9        MOVL    @4(R3), R8                            ; 0918
           51        68  90 000DD        MOVB    (R8), CHAR                            ;
                 04  B3  D6 000E0        INCL    @4(R3)                                ;
                     63  B7 000E3        DECW    (R3)                                  ; 0919
           80        51  90 000E5 10$:   MOVB    CHAR, (OUTPUT_PTR)+                   ; 0928
                     55  D6 000E8        INCL    COUNT                                 ; 0929
                     CB  11 000EA        BRB     9$                                    ; 0887
           64        55  90 000EC 11$:   MOVB    COUNT, (COUNTED_STRING)               ; 0935
           67        54  D0 000EF        MOVL    COUNTED_STRING, (NOUN_NODE)           ; 0940
           50        01  D0 000F2 12$:   MOVL    #1, R0                                ; 0945
                         04 000F5        RET                                           ; 0947
```

; Routine Size: 246 bytes,   Routine Base: DBG$CODE + 0351

```
;  822          0948  1 GLOBAL ROUTINE DBG$NEXECUTE_SPAWN (VERB_NODE, MESSAGE_VECT) =
;  823          0949  1 !
;  824          0950  1 ! FUNCTION
;  825          0951  1 !       This routine executes a SPAWN command.
;  826          0952  1 !
;  827          0953  1 ! FORMAL PARAMETERS:
;  828          0954  1 !
;  829          0955  1 !       VERB_NODE - The command verb node which is the start of the command
;  830          0956  1 !                   execution tree.
;  831          0957  1 !
;  832          0958  1 !       MESSAGE_VECT - The address of a longword to contain the address of
;  833          0959  1 !                   a message argument vector.
;  834          0960  1 !
;  835          0961  1 ! ROUTINE VALUE:
;  836          0962  1 !       An unsigned integer longword completion code:
;  837          0963  1 !
;  838          0964  1 !               STS$K_SUCCESS (1) - Indicates the file was created and connected
;  839          0965  1 !
;  840          0966  1 !               STS$K_SEVERE (4) - Failure, file not created. Message argument
;  841          0967  1 !                                  vector returned.
;  842          0968  1 !
;  843          0969  1
;  844          0970  2     BEGIN
;  845          0971  2
;  846          0972  2     MAP
;  847          0973  2         VERB_NODE: REF DBG$VERB_NODE;
;  848          0974  2
;  849          0975  2     LOCAL
;  850          0976  2         COUNTED_STRING:                 ! Contains arg to SPAWN
;  851          0977  2                 REF VECTOR[,BYTE],      !
;  852          0978  2         EVNT_FLAG,                      ! Event flag for $GETJPI call
;  853          0979  2         FLAGS: BITVECTOR[32],           ! Flags for /NOWAIT etc.
;  854          0980  2         ITEM: BLOCK[6, LONG],           ! Item list for GETJPI
;  855          0981  2         LEN: WORD,                      ! Length of process name
;  856          0982  2         NAME_BUFFER: VECTOR[32,BYTE],   ! Process name
;  857          0983  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Command object noun
;  858          0984  2         STATUS,                         ! Return status from LIB$SPANW
;  859          0985  2         STG_DESC: DBG$STG_DESC,         ! A string descriptor
;  860          0986  2         X;
;  861          0987  2
;  862          0988  2
;  863          0989  2     ! Reset the terminal screen to "normal" if we are in screen mode.  This
;  864          0990  2     ! makes sure that the whole terminal screen scrolls when we are in the
;  865          0991  2     ! other process and that the screen is refreshed when we reenter DEBUG.
;  866          0992  2     !
;  867          0993  2     DBG$SCR_SCREEN_NORMAL();
;  868          0994  2
;  869          0995  2
;  870          0996  2     ! Obtain the qualifier flags and the noun node.
;  871          0997  2     ! The flags will tell us whether /NOWAIT was specified (this
;  872          0998  2     ! is currently the only qualifier we support).
;  873          0999  2     ! The noun node tells whether a DCL command was given as argument,
;  874          1000  2     ! or whether we saw a bare SPAWN.
;  875          1001  2     !
;  876          1002  2     FLAGS = .VERB_NODE [DBG$L_VERB_ADVERB_PTR];
;  877          1003  2     NOUN_NODE = .VERB_NODE [DBG$L_VERB_OBJECT_PTR];
;  878          1004  2
```

```
;  879          1005  2
;  880          1006  2      ! Note - in both cases below we would like to put out the
;  881          1007  2      ! "SPAWNED - process XXX spawned; terminal now attached to ..."
;  882          1008  2      ! as DCL does. But LIB$SPAWN does not put out this message, and
;  883          1009  2      ! there appears to be no way we can put it out at this level
;  884          1010  2      ! (we don't know if LIB$SPAWN will succeed, and even if it does,
;  885          1011  2      ! we don't know the name of the spawned subprocess).
;  886          1012  2      !
;  887          1013  2      ! If no command string is given, just call LIB$SPAWN with flags argument.
;  888          1014  2      !
;  889          1015  2      IF .NOUN_NODE EQL 0
;  890          1016  2      THEN
;  891          1017  2          STATUS = LIB$SPAWN(0,0,0,FLAGS,0,0)
;  892          1018  2
;  893          1019  2
;  894          1020  2      ! If a command string was given, we pass it into LIB$SPAWN along with the
;  895          1021  2      ! flags argument.
;  896          1022  2      !
;  897          1023  2      ELSE
;  898          1024  3          BEGIN
;  899          1025  3          COUNTED_STRING = .NOUN_NODE[DBG$L_NOUN_VALUE];
;  900          1026  3          STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  901          1027  3          STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  902          1028  3          STG_DESC[DSC$W_LENGTH] = .COUNTED_STRING[0];
;  903          1029  3          STG_DESC[DSC$A_POINTER] = COUNTED_STRING[1];
;  904          1030  3          STATUS = LIB$SPAWN(STG_DESC,0,0,FLAGS,0,0);
;  905          1031  2          END;
;  906          1032  2
;  907          1033  2
;  908          1034  2      ! If a bad status was returned then signal a DEBUG error and include
;  909          1035  2      ! the return status as the secondary error.
;  910          1036  2      !
;  911          1037  2      IF NOT .STATUS
;  912          1038  2      THEN
;  913          1039  2          SIGNAL(DBG$_NOSPAWN, 0, .STATUS)
;  914          1040  2
;  915          1041  2      ELSE
;  916          1042  3          BEGIN
;  917          1043  3          IF .FLAGS[NOWAIT_FLAG]
;  918          1044  3          THEN
;  919          1045  3
;  920          1046  3              ! We can put out an informational message if a SPAWN/NOWAIT
;  921          1047  3              ! subprocess was created.
;  922          1048  3              !
;  923          1049  3              SIGNAL(DBG$_SPAWNED)
;  924          1050  3
;  925          1051  3          ELSE
;  926          1052  4              BEGIN
;  927          1053  4
;  928          1054  4
;  929          1055  4              ! On a successful return just give an informational message that
;  930          1056  4              ! control is being returned to the parent process.
;  931          1057  4              !
;  932          1058  4              ! Note - if we return to the parent via an ATTACH, we will
;  933          1059  4              ! go through this code, which we don't really want to do.
;  934          1060  4              !
;  935          1061  4              ! We have to first go through some work just to determine the
```

```
;  936          1062  4              ! name of the parent process.
;  937          1063  4              !
;  938          1064  4              ITEM[0,0,16,0] = 15;
;  939          1065  4              ITEM[0,16,16,0] = JPI$_PRCNAM;
;  940          1066  4              ITEM[1,0,32,0] = NAME_BUFFER;
;  941          1067  4              ITEM[2,0,32,0] = LEN;
;  942          1068  4              CH$FILL(0, 12, ITEM[3, A_]);
;  943          1069  4              STATUS = LIB$GET_EF(EVNT_FLAG);
;  944          1070  4              IF NOT .STATUS THEN SIGNAL(.STATUS);
;  945          1071  4              STATUS = $GETJPI(EFN=.EVNT_FLAG, ITMLST=ITEM);
;  946          1072  4              LIB$FREE_EF(EVNT_FLAG);
;  947          1073  4              SIGNAL(DBG$_RETURNED, 2, .LEN, NAME_BUFFER);
;  948          1074  3              END;
;  949          1075  3
;  950          1076  2          END;
;  951          1077  2
;  952          1078  2      ! After returning from the spawn, set the terminal back to
;  953          1079  2      ! the appropriate mode. Then return.
;  954          1080  2      !
;  955          1081  2      IF .DBG$GB_KEYPAD_INPUT THEN X = 1 ELSE X = 0;
;  956          1082  2      SMG$SET_KEYPAD_MODE (DBG$GL_KEYBOARD_ID, X);
;  957          1083  2      RETURN STS$K_SUCCESS;
;  958          1084  1      END;
; INFO#250                L1:1073
; Referenced LOCAL symbol LEN is probably not initialized
```

```
                                00FC 00000          .ENTRY  DBG$NEXECUTE_SPAWN, Save R2,R3,R4,R5,R6,R7 ; 0948
                     57 00000000G   00   9E 00002   MOVAB   LIB$SIGNAL, R7                             ;
                     5E        AC   AE   9E 00009   MOVAB   -84(SP), SP                               ;
00000000G            00             00   FB 0000D   CALLS   #0, DBG$SCR_SCREEN_NORMAL                 ; 0993
                     50        04   AC   D0 00014   MOVL    VERB_NODE, R0                            ; 1002
                     6E        04   A0   D0 00018   MOVL    4(R0), FLAGS                             ;
                     50        08   A0   D0 0001C   MOVL    8(R0), NOUN_NODE                         ; 1003
                                    0B   12 00020   BNEQ    1$                                       ; 1015
                                    7E   7C 00022   CLRQ    -(SP)                                    ; 1017
                              08    AE   9F 00024   PUSHAB  FLAGS                                    ;
                                    7E   7C 00027   CLRQ    -(SP)                                    ;
                                    7E   D4 00029   CLRL    -(SP)                                    ;
                                    1C   11 0002B   BRB     2$                                       ;
                     50             60   D0 0002D 1$:  MOVL    (NOUN_NODE), COUNTED_STRING              ; 1025
                12   AE       010E   8F   B0 00030   MOVW    #270, STG_DESC+2                         ; 1027
                10   AE              60   9B 00036   MOVZBW  (COUNTED_STRING), STG_DESC               ; 1028
                14   AE        01    A0   9E 0003A   MOVAB   1(R0), STG_DESC+4                        ; 1029
                                    7E   7C 0003F   CLRQ    -(SP)                                    ; 1030
                              08    AE   9F 00041   PUSHAB  FLAGS                                    ;
                                    7E   7C 00044   CLRQ    -(SP)                                    ;
                              24    AE   9F 00046   PUSHAB  STG_DESC                                 ;
00000000G            00             06   FB 00049 2$:  CALLS   #6, LIB$SPAWN                        ;
                     56             50   D0 00050   MOVL    R0, STATUS                               ;
                     0F             56   E8 00053   BLBS    STATUS, 3$                               ; 1037
                                    56   DD 00056   PUSHL   STATUS                                   ; 1039
                                    7E   D4 00058   CLRL    -(SP)                                    ;
                          00028A1A  8F   DD 0005A   PUSHL   #166426                                  ;
```

```
                                67              03 FB 00060          CALLS   #3, LIB$SIGNAL                                  ;
                                                6E 11 00063          BRB     6$                                              ;
                                0B              6E E9 00065 3$:      BLBC    FLAGS, 4$                                       ; 1043
                            000286F3            8F DD 00068          PUSHL   #165619                                         ; 1049
                                67              01 FB 0006E          CALLS   #1, LIB$SIGNAL                                  ;
                                                60 11 00071          BRB     6$                                              ;
                          3C    AE  031C000F    8F D0 00073 4$:      MOVL    #52166671, ITEM                                 ; 1064
                          40    AE        1C    AE 9E 0007B          MOVAB   NAME_BUFFER, ITEM+4                             ; 1066
                          44    AE        04    AE 9E 00080          MOVAB   LEN, ITEM+8                                     ; 1067
              0C          00              6E    00 2C 00085          MOVC5   #0, (SP), #0, #12, ITEM+12                      ; 1068
                                                48 AE    0008A
                                                08 AE 9F 0008C          PUSHAB  EVNT_FLAG                                    ; 1069
                    00000000G   00              01 FB 0008F          CALLS   #1, LIB$GET_EF
                                56              50 D0 00096          MOVL    R0, STATUS
                                05              56 E8 00099          BLBS    STATUS, 5$                                      ; 1070
                                                56 DD 0009C          PUSHL   STATUS
                                67              01 FB 0009E          CALLS   #1, LIB$SIGNAL
                                                7E 7C 000A1 5$:      CLRQ    -(SP)                                           ; 1071
                                                7E D4 000A3          CLRL    -(SP)
                                                48 AE 9F 000A5          PUSHAB  ITEM
                                                7E 7C 000A8          CLRQ    -(SP)
                                                20 AE DD 000AA          PUSHL   EVNT_FLAG
                    00000000G   00              07 FB 000AD          CALLS   #7, SYS$GETJPI
                                56              50 D0 000B4          MOVL    R0, STATUS
                                                08 AE 9F 000B7          PUSHAB  EVNT_FLAG                                    ; 1072
                    00000000G   00              01 FB 000BA          CALLS   #1, LIB$FREE_EF
                                                1C AE 9F 000C1          PUSHAB  NAME_BUFFER                                  ; 1073
                                7E      08      AE 3C 000C4          MOVZWL  LEN, -(SP)
                                                02 DD 000C8          PUSHL   #2
                            000286D3            8F DD 000CA          PUSHL   #165587
                                67              04 FB 000D0          CALLS   #4, LIB$SIGNAL
                                06  00000000G   00 E9 000D3 6$:      BLBC    DBG$GB_KEYPAD_INPUT, 7$                         ; 1081
                          0C    AE              01 D0 000DA          MOVL    #1, X
                                                03 11 000DE          BRB     8$
                                      0C    AE D4 000E0 7$:      CLRL    X
                                      0C    AE 9F 000E3 8$:      PUSHAB  X                                                ; 1082
                                00000000G   00 9F 000E6          PUSHAB  DBG$GL_KEYBOARD_ID
                    00000000G   00              02 FB 000EC          CALLS   #2, SMG$SET_KEYPAD_MODE
                                50              01 D0 000F3          MOVL    #1, R0                                          ; 1083
                                                04 000F6          RET                                                     ; 1084

; Routine Size:  247 bytes,    Routine Base:  DBG$CODE + 0447
```

```
;  960          1085  1 GLOBAL ROUTINE dbg$construct_param_name (param_count, param_name,
;  961          1086  1                                 perm_flag, message_vect) =
;  962          1087  1 !++
;  963          1088  1 ! Routine Description
;  964          1089  1 !
;  965          1090  1 !       This routine constructs a name of the form "%Pi", given the
;  966          1091  1 !       number i. The reason there is a global routine for this is
;  967          1092  1 !       that this needs to be called from within DBG$NEXECUTE_DECLARE
;  968          1093  1 !       as well as DBG$NEXECUTE_AT_SIGN.
;  969          1094  1 !
;  970          1095  1 ! Inputs
;  971          1096  1 !
;  972          1097  1 !       param_count     -       The number "i" which is to be made into the
;  973          1098  1 !                               name "%Pi".
;  974          1099  1 !       param_name      -       An address to receive the constructed name.
;  975          1100  1 !       perm_flag       -       If TRUE, construct param name out of permanent
;  976          1101  1 !                               memory
;  977          1102  1 !       message_vect    -       An error message vector.
;  978          1103  1 !
;  979          1104  1 ! Outputs
;  980          1105  1 !
;  981          1106  1 !       A counted string is constructed and param_name is filled in with
;  982          1107  1 !       its address.
;  983          1108  1 !       A condition code is returned. This is one of:
;  984          1109  1 !       STS$K_SUCCESS - success.
;  985          1110  1 !       STS$K_SEVERE - error. An error message vector is constructed.
;  986          1111  1 !
;  987          1112  1 ! Side Effects
;  988          1113  1 !
;  989          1114  1 !       Memory is allocated for the name.
;  990          1115  1 !--
;  991          1116  2     BEGIN
;  992          1117  2
;  993          1118  2     LOCAL
;  994          1119  2         outlen: WORD,                           ! Receives length from SYS$FAO
;  995          1120  2         param_len: WORD,                        ! Length of parameter name
;  996          1121  2         stg_desc1       : dbg$stg_desc,         ! Used in call to SYS$FAO
;  997          1122  2         stg_desc2       : dbg$stg_desc,         ! Used in call to SYS$FAO
;  998          1123  2         temp_name: REF VECTOR [,BYTE];          ! Holds name temporarily
;  999          1124  2
; 1000          1125  2     EXTERNAL ROUTINE
; 1001          1126  2         sys$fao;
; 1002          1127  2
; 1003          1128  2     ! Determine the length of the name.
; 1004          1129  2     !
; 1005          1130  2     IF .param_count GEQ 1 AND .param_count LEQ 9
; 1006          1131  2     THEN
; 1007          1132  2         param_len = 3
; 1008          1133  2     ELSE IF .param_count GEQ 10 AND .param_count LEQ 99
; 1009          1134  2     THEN
; 1010          1135  2         param_len = 4
; 1011          1136  2     ELSE
; 1012          1137  2         $DBG_ERROR('DBGATSIGN\DBG$CONSTRUCT_PARAM_NAME');
; 1013          1138  2
; 1014          1139  2     ! Allocate space for the name.
; 1015          114   2     !
; 1016          1141  2     IF .perm_flag
```

```
;  1017          1142  2      THEN
;  1018          1143  3          BEGIN
;  1019          1144  3          temp_name = dbg$get_memory (1+ (1+.param_len)/4);
;  1020          1145  3          END
;  1021          1146  2      ELSE
;  1022          1147  3          BEGIN
;  1023          1148  3          temp_name = dbg$get_tempmem (1+ (1+.param_len)/4);
;  1024          1149  2          END;
;  1025          1150  2
;  1026          1151  2      temp_name[0] = .param_len;
;  1027          1152  2      ch$move (2, UPLIT BYTE(%ASCII '%P'), temp_name[1]);
;  1028          1153  2
;  1029          1154  2      ! Fill in the string descriptor for use by SYS$FAO.
;  1030          1155  2      !
;  1031          1156  2      stg_desc1[dsc$w_length] = 3;
;  1032          1157  2      stg_desc1[dsc$a_pointer] = UPLIT BYTE (%ASCII '!SL');
;  1033          1158  2      stg_desc2[dsc$w_length] = .param_len-2;
;  1034          1159  2      stg_desc2[dsc$a_pointer] = temp_name[3];
;  1035          1160  2
;  1036          1161  2      ! Call SYS$FAO to convert the number into a string.
;  1037          1162  2      !
;  1038          1163  2      IF NOT sys$fao (stg_desc1, outlen, stg_desc2, .param_count)
;  1039          1164  2      THEN
;  1040          1165  2          $DBG_ERROR('DBGATSIGN\DBG$CONSTRUCT_PARAM_NAME');
;  1041          1166  2
;  1042          1167  2      ! Check that SYS$FAO filled in the expected number of characters.
;  1043          1168  2      !
;  1044          1169  2      IF .outlen NEQ .param_len-2
;  1045          1170  2      THEN
;  1046          1171  2          $DBG_ERROR('DBGATSIGN\DBG$CONSTRUCT_PARAM_NAME');
;  1047          1172  2
;  1048          1173  2      ! Fill in output parameter.
;  1049          1174  2      !
;  1050          1175  2      .param_name = .temp_name;
;  1051          1176  2
;  1052          1177  2      ! Return success.
;  1053          1178  2      !
;  1054          1179  2      RETURN sts$k_success;
;  1055          1180  2
;  1056          1181  1      END; ! dbg$construct_param_name


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

24 47 42 44 5C 4E 47 49 53 54 41 47 42 44 22  00047 P.AAM:  .ASCII  \"DBGATSIGN\<92>\DBG$CONSTRUCT_PARAM_NAM\   ;
4D 41 52 41 50 5F 54 43 55 52 54 53 4E 4F 43  00056                                                          ;
                                  4D 41 4E 5F 00065                                                          ;
                                            45 00069          .ASCII  \E\                                    ;
                                         50 25 0006A P.AAN:   .ASCII  \%P\                                   ;
                                      4C 53 21 0006C P.AAO:   .ASCII  \!SL\                                  ;
24 47 42 44 5C 4E 47 49 53 54 41 47 42 44 22  0006F P.AAP:   .ASCII  \"DBGATSIGN\<92>\DBG$CONSTRUCT_PARAM_NAM\   ;
4D 41 52 41 50 5F 54 43 55 52 54 53 4E 4F 43  0007E                                                          ;
                                  4D 41 4E 5F 0008D                                                          ;
                                            45 00091          .ASCII  \E\                                    ;
24 47 42 44 5C 4E 47 49 53 54 41 47 42 44 22  00092 P.AAQ:   .ASCII  \"DBGATSIGN\<92>\DBG$CONSTRUCT_PARAM_NAM\   ;
4D 41 52 41 50 5F 54 43 55 52 54 53 4E 4F 43  000A1                                                          ;
```

```
                        4D  41  4E  5F  000B0
                                    45  000B4        .ASCII  \E\                                                     ;
                                                     .EXTRN  SYS$FAO                                                 ;
                                                     .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0                            ;
                                        007C 00000   .ENTRY  DBG$CONSTRUCT_PARAM_NAME, Save R2,R3,R4,R5,-           ; 1085
                                                             R6                                                      ;
                    56 00000000G   00    9E 00002   MOVAB   LIB$SIGNAL, R6                                          ;
                    55 00000000'   EF    9E 00009   MOVAB   P.AAM, R5                                               ;
                    5E             1C    C2 00010   SUBL2   #28, SP                                                 ;
                    54       04    AC    D0 00013   MOVL    PARAM_COUNT, R4                                         ; 1130
                                   0A    15 00017   BLEQ    1$                                                      ;
                    09             54    D1 00019   CMPL    R4, #9                                                  ;
                                   05    14 0001C   BGTR    1$                                                      ;
                    53             03    B0 0001E   MOVW    #3, PARAM_LEN                                           ; 1132
                                   20    11 00021   BRB     3$                                                      ;
                    0A             54    D1 00023 1$:     CMPL    R4, #10                                         ; 1133
                                   0E    19 00026   BLSS    2$                                                      ;
              00000063   8F        54    D1 00028   CMPL    R4, #99                                                 ;
                                   05    14 0002F   BGTR    2$                                                      ;
                    53             04    B0 00031   MOVW    #4, PARAM_LEN                                           ; 1135
                                   0D    11 00034   BRB     3$                                                      ;
                                   55    DD 00036 2$:     PUSHL   R5                                              ; 1137
                                   01    DD 00038   PUSHL   #1                                                      ;
                       00028362    8F    DD 0003A   PUSHL   #164706                                                 ;
                    66             03    FB 00040   CALLS   #3, LIB$SIGNAL                                         ;
                    50             53    3C 00043 3$:     MOVZWL  PARAM_LEN, R0                                   ; 1144
                                   50    D6 00046   INCL    R0                                                      ;
                    50             04    C6 00048   DIVL2   #4, R0                                                  ;
                                   50    D6 0004B   INCL    R0                                                      ;
                    0B       0C    AC    E9 0004D   BLBC    PERM_FLAG, 4$                                           ; 1141
                                   50    DD 00051   PUSHL   R0                                                      ; 1144
              00000000G      00    01    FB 00053   CALLS   #1, DBG$GET_MEMORY                                      ;
                                   09    11 0005A   BRB     5$                                                      ;
                                   50    DD 0005C 4$:     PUSHL   R0                                              ; 1148
              00000000G      00    01    FB 0005E   CALLS   #1, DBG$GET_TEMPMEM                                     ;
                    52             50    D0 00065 5$:     MOVL    R0, TEMP_NAME                                   ;
                    62             53    90 00068   MOVB    PARAM_LEN, (TEMP_NAME)                                  ; 1151
                 01 A2       23    A5    B0 0006B   MOVW    P.AAN, 1(TEMP_NAME)                                     ; 1152
                 10 AE             03    B0 00070   MOVW    #3, STG_DESC1                                           ; 1156
                 14 AE       25    A5    9E 00074   MOVAB   P.AAO, STG_DESC1+4                                     ; 1157
                    53             53    3C 00079   MOVZWL  PARAM_LEN, R3                                          ; 1158
                    53             02    C2 0007C   SUBL2   #2, R3                                                  ;
                 04 AE             53    B0 0007F   MOVW    R3, STG_DESC2                                           ;
                 08 AE       03    A2    9E 00083   MOVAB   3(R2), STG_DESC2+4                                     ; 1159
                                   54    DD 00088   PUSHL   R4                                                      ; 1163
                                08 AE    9F 0008A   PUSHAB  STG_DESC2                                               ;
                                08 AE    9F 0008D   PUSHAB  OUTLEN                                                  ;
                                1C AE    9F 00090   PUSHAB  STG_DESC1                                               ;
              00000000G      00    04    FB 00093   CALLS   #4, SYS$FAO                                             ;
                    0E             50    E8 0009A   BLBS    R0, 6$                                                  ;
                                28 A5    9F 0009D   PUSHAB  P.AAP                                                   ; 1165
                                   01    DD 000A0   PUSHL   #1                                                      ;
                       00028362    8F    DD 000A2   PUSHL   #164706                                                 ;
                    66             03    FB 000A8   CALLS   #3, LIB$SIGNAL                                         ;
```

```
                53              6E              10              00  ED 000AB 6$:      CMPZV   #0, #16, OUTLEN, R3             ; 1169
                                                                0E  13 000B0          BEQL    7$                              ;
                                                        4B      A5  9F 000B2          PUSHAB  P.AAQ                           ; 1171
                                                                01  DD 000B5          PUSHL   #1                              ;
                                                00028362        8F  DD 000B7          PUSHL   #164706                         ;
                                                66              03  FB 000BD          CALLS   #3, LIB$SIGNAL                  ;
                                        08      BC              52  D0 000C0 7$:      MOVL    TEMP_NAME, @PARAM_NAME          ; 11?3
                                                50              01  D0 000C4          MOVL    #1, R0                          ; 1179
                                                                04  000C7          RET                                     ; 1181

; Routine Size:  200 bytes,    Routine Base:  DBG$CODE + 053E
```

```
;1058          1182  1 GLOBAL ROUTINE dbg$parse_spawn (parse_stg_desc) =
;1059          1183  1 !+
;1060          1184  1 ! Functional Description
;1061          1185  1 !
;1062          1186  1 !       This routine provides an interface from the old language parsers to
;1063          1187  1 !       the new debugger parse network for SPAWN. It is passed a string
;1064          1188  1 !       descriptor for the remainder of the input line.
;1065          1189  1 !       It calls DBG$NPARSE_SPAWN to construct
;1066          1190  1 !       a command execution network, and returns a pointer to the verb node.
;1067          1191  1 !
;1068          1192  1 ! Inputs
;1069          1193  1 !
;1070          1194  1 !       parse_stg_desc  -       A string descriptor for the remainder of the
;1071          1195  1 !                               input line.
;1072          1196  1 !
;1073          1197  1 ! Outputs
;1074          1198  1 !
;1075          1199  1 !       A command execution network is constructed,
;1076          1200  1 !       consisting of a verb node for the SPAWN
;1077          1201  1 !       verb, and possibly a noun node for the argument
;1078          1202  1 !--
;1079          1203  2     BEGIN
;1080          1204  2     MAP
;1081          1205  2         parse_stg_desc : REF BLOCK [,BYTE];
;1082          1206  2
;1083          1207  2     LOCAL
;1084          1208  2         char,
;1085          1209  2         dummy_mess_vect: REF VECTOR,    ! Address for message vector returned
;1086          1210  2                                         !    from DBG$NPARSE_SPAWN
;1087          1211  2         len,                            ! Length of command line
;1088          1212  2         parse_stg_ptr,                  ! Pointer into command line
;1089          1213  2         stg : REF VECTOR [,BYTE],       ! Pointer to a new copy of the
;1090          1214  2                                         !    command line
;1091          1215  2         verb_node;                      ! Pointer to the head of the command
;1092          1216  2                                         !    execution tree for SPAWN
;1093          1217  2
;1094          1218  2     ! Call the 'new style' parse network for the spawn
;1095          1219  2     ! command. This builds a command execution network.
;1096          1220  2     ! We return a pointer to the verb node.
;1097          1221  2     !
;1098          1222  2     ! First allocate space for the verb node.
;1099          1223  2     !
;1100          1224  2     verb_node = dbg$get_tempmem (dbg$k_verb_node_size);
;1101          1225  2
;1102          1226  2     ! Then stuff a carriage return character at the end
;1103          1227  2     ! of the input line since this is what the new style
;1104          1228  2     ! parser expects to see. Also, translate the line to
;1105          1229  2     ! upper case (the new debugger does this; the ol. does not)
;1106          1230  2     !
;1107          1231  2     len = .parse_stg_desc[dsc$w_length];
;1108          1232  2     stg = dbg$get_tempmem (1+(1+.len)/%UPVAL);
;1109          1233  2     parse_stg_ptr = ch$ptr(.parse_stg_desc[dsc$a_pointer]);
;1110          1234  2     INCR i FROM 0 TO .len-1 DO
;1111          1235  3         BEGIN
;1112          1236  3         char = ch$rchar_a(parse_stg_ptr);
;1113          1237  3         IF .char GEQ %C'a' AND .char LEQ %C'z'
;1114          1238  3         THEN
```

```
;  1115     1239  4              stg[ i] = .char - (%C'a'-%C'A')
;  1116     1240  3          ELSE
;  1117     1241  3              stg[.i] = .char;
;  1118     1242  2          END;
;  1119     1243  2      stg[.len] = dbg$k_car_return;
;  1120     1244  2      parse_stg_desc[dsc$a_pointer] = .stg;
;  1121     1245  2      parse_stg_desc[dsc$w_length] =
;  1122     1246  2          .parse_stg_desc[dsc$w_length] + 1;
;  1123     1247  2
;  1124     1248  2      ! Now call the parser on the remainder of the input line
;  1125     1249  2      !
;  1126     1250  2      IF NOT dbg$nparse_spawn (.parse_stg_desc,
;  1127     1251  2          .verb_node, dummy_mess_vect)
;  1128     1252  2      THEN
;  1129     1253  2          ! If the above routine does not return success, then we signal
;  1130     1254  2          ! an error using the error message vector that we got back.
;  1131     1255  2          !
;  1132     1256  3          BEGIN
;  1133     1257  3          EXTERNAL ROUTINE
;  1134     1258  3              LIB$SIGNAL : ADDRESSING_MODE(GENERAL);
;  1135     1259  3          BUILTIN
;  1136     1260  3              CALLG;
;  1137     1261  3          CALLG (.dummy_mess_vect, lib$signal);
;  1138     1262  2          END;
;  1139     1263  2
;  1140     1264  2      ! Restore pointer field of PARSE_STG_DESC since this can be wiped out
;  1141     1265  2      ! during new style parsing.
;  1142     1266  2      !
;  1143     1267  2      IF .parse_stg_desc[dsc$a_pointer] EQL 0
;  1144     1268  2      THEN
;  1145     1269  2          parse_stg_desc[dsc$a_pointer] = .stg+.len;
;  1146     1270  2
;  1147     1271  2      ! Finally, return a pointer to the verb node.
;  1148     1272  2      !
;  1149     1273  2      RETURN .verb_node
;  1150     1274  2
;  1151     1275  1      END; ! dbg$parse_spawn


                                                .EXTRN  LIB$SIGNAL

                                00FC 00000      .ENTRY  DBG$PARSE_SPAWN, Save R2,R3,R4,R5,R6,R7      ; 1182
    57 00000000G   00    9E 00002      MOVAB   DBG$GET_TEMPMEM, R7                                  ;
    5E             04    C2 00009      SUBL2   #4, SP                                               ;
                   03    DD 0000C      PUSHL   #3                                                   ; 1224
    67             01    FB 0000E      CALLS   #1, DBG$GET_TEMPMEM                                  ;
    56             50    D0 00011      MOVL    R0, VERB_NODE                                        ;
    54       04    AC    D0 00014      MOVL    PARSE_STG_DESC, R4                                   ; 1231
    52             64    3C 00018      MOVZWL  (R4), LEN                                            ;
    50       01    A2    9E 0001B      MOVAB   1(R2), R0                                            ; 1232
    50             04    C6 0001F      DIVL2   #4, R0                                               ;
             01    A0    9F 00022      PUSHAB  1(R0)                                                ;
    67             01    FB 00025      CALLS   #1, DBG$GET_TEMPMEM                                  ;
    53             50    D0 00028      MOVL    R0, STG                                              ;
    55       04    A4    D0 0002B      MOVL    4(R4), PARSE_STG_PTR                                 ; 1233
    50             01    CE 0002F      MNEGL   #1, I                                                ; 1239
```

```
                                                20  11 00032          BRB     3$                                      ;
                                          51    85  9A 00034 1$:      MOVZBL  (PARSE_STG_PTR)+, CHAR                  ; 1236
                                00000061  8F    51  D1 00037          CMPL    CHAR, #97                               ; 1237
                                                10  19 0003E          BLSS    2$                                      ;
                                0000007A  8F    51  D1 00040          CMPL    CHAR, #122                              ;
                                                07  14 00047          BGTR    2$                                      ;
                              6043        51    20  83 00049          SUBB3   #32, CHAR, (I)[STG]                     ; 1239
                                                04  11 0004E          BRB     3$                                      ;
                                    6043  51    51  90 00050 2$:      MOVB    CHAR, (I)[STG]                          ; 1241
                                DC        50    52  F2 00054 3$:      AOBLSS  LEN, I, 1$                              ; 1234
                                    6243  50    0D  90 00058          MOVB    #13, (LEN)[STG]                         ; 1243
                                04  A4    53    D0  0005C          MOVL    STG, 4(R4)                              ; 1244
                                            64    B6  00060          INCW    (R4)                                    ; 1246
                                      4050  5F    BB  00062          PUSHR   #^M<R4,R6,SP>                           ; 1250
                                FCE0    CF  03    FB  00066          CALLS   #3, DBG$NPARSE_SPAWN                    ;
                                        08  50    E8  0006B          BLBS    R0, 4$                                  ;
                          00000000G     00  BE    FA  0006E          CALLG   @DUMMY_MESS_VECT, LIB$SIGNAL            ; 1261
                                        50  04    AC  D0  00076 4$:   MOVL    PARSE_STG_DESC, R0                      ; 1267
                                            04    A0  D5  0007A          TSTL    4(R0)                                   ;
                                                  05  12  0007D          BNEQ    5$                                      ;
                          04    A0              53    52  C1  0007F          ADDL3   LEN, STG, 4(R0)                 ; 1269
                                                50    56  D0  00084 5$:      MOVL    VERB_NODE, R0                   ; 1273
                                                      04  00087          RET                                     ; 1275

; Routine Size:  136 bytes,    Routine Base:  DBG$CODE + 0606
```

```
;  1153          1276  1 ROUTINE get_param (input_desc, paren_flag: BYTE, result_addr) =
;  1154          1277  1 !++
;  1155          1278  1 ! Routine Description
;  1156          1279  1 !
;  1157          1280  1 !       This routine collects a parameter from the input stream. It allocates
;  1158          1281  1 !       permanent memory and copies the parameter into that memory. A
;  1159          1282  1 !       parameter may be inside of double quotes ("), in which case this
;  1160          1283  1 !       routine reads to the closing quote. Or, if it is not inside of
;  1161          1284  1 !       quotes then comma (or possibly right paren) is the terminator.
;  1162          1285  1 !
;  1163          1286  1 ! Inputs
;  1164          1287  1 !
;  1165          1288  1 !       input_desc -    A string descriptor for the remaining input.
;  1166          1289  1 !       paren_flag -    TRUE if the parameter list is parenthesized. This
;  1167          1290  1 !                       affects the parsing of an individual parameter in
;  1168          1291  1 !                       that a closing paren is treated as a terminator.
;  1169          1292  1 !       result_addr -   The address to leave a pointer to the result.
;  1170          1293  1 !
;  1171          1294  1 ! Outputs
;  1172          1295  1 !
;  1173          1296  1 !       A counted string is constructed out of permanent memory and filled
;  1174          1297  1 !       in with the parameter that is read, and a pointer to this string
;  1175          1298  1 !       is left in result_addr.
;  1176          1299  1 !       The value TRUE is returned if a parameter is successfully read.
;  1177          1300  1 !       The value FALSE is returned if a parameter cannot be read.
;  1178          1301  1 !--
;  1179          1302  2     BEGIN
;  1180          1303  2
;  1181          1304  2     MAP
;  1182          1305  2         input_desc: REF dbg$stg_desc,
;  1183          1306  2         paren_flag: BYTE;
;  1184          1307  2
;  1185          1308  2     LOCAL
;  1186          1309  2         char: BYTE,                     ! Holds a character from the stream
;  1187          1310  2         chars_read,                     ! Count of number of characters read
;  1188          1311  2         input_ptr,                      ! Pointer into current position in input stream
;  1189          1312  2         output_ptr,                     ! Pointer into result string
;  1190          1313  2         prev_char: BYTE,                ! Holds previously-read character
;  1191          1314  2         prev_input_ptr,                 ! Pointer to previous position in input stream
;  1192          1315  2         quote_flag: BYTE,               ! True if the parameter is quoted
;  1193          1316  2         result: REF VECTOR[,BYTE],      ! Points to the result string
;  1194          1317  2         saved_input_ptr,                ! Saved value of input pointer.
;  1195          1318  2         tot_chars_read;                 ! Count of total characters read
;  1196          1319  2
;  1197          1320  2     ! The get_char macro is used when we want to read another character.
;  1198          1321  2     !
;  1199        M 1322  2     MACRO get_char =
;  1200        M 1323  2         prev_input_ptr = .input_ptr;
;  1201        M 1324  2         char = ch$rchar_a (input_ptr);
;  1202          1325  2         tot_chars_read = .tot_chars_read + 1%;
;  1203          1326  2
;  1204          1327  2     ! Initialize some variables.
;  1205          1328  2     !
;  1206          1329  2     input_ptr = .input_desc [dsc$a_pointer];
;  1207          1330  2     tot_chars_read = 0;
;  1208          1331  2     chars_read = 0;
;  1209          1332  2     prev_char = %C' ';
```

```
;  1210          1333  2      char = %C' ';
;  1211          1334  2
;  1212          1335  2      ! First skip leading blanks.
;  1213          1336  2      !
;  1214          1337  2      WHILE .char EQL dbg$k_blank
;  1215          1338  2        AND .tot_chars_read LSS .input_desc [dsc$w_length]
;  1216          1339  2          DO
;  1217          1340  3          BEGIN
;  1218          1341  3          get_char;
;  1219          1342  2          END;
;  1220          1343  2
;  1221          1344  2      ! Check for leading quote.
;  1222          1345  2      !
;  1223          1346  2      IF .char EQL dbg$k_dblquote
;  1224          1347  2      THEN
;  1225          1348  3          BEGIN
;  1226          1349  3          (.prev_input_ptr)<0,8,0> = 0;
;  1227          1350  3          quote_flag = TRUE;
;  1228          1351  3          get_char;
;  1229          1352  3          END
;  1230          1353  2      ELSE
;  1231          1354  2          quote_flag = FALSE;
;  1232          1355  2
;  1233          1356  2      saved_input_ptr = ch$plus (.input_ptr, -1);
;  1234          1357  2
;  1235          1358  2      ! Loop through the string representing the parameter,
;  1236          1359  2      ! until a terminator is found.
;  1237          1360  2      !
;  1238          1361  2      WHILE .char NEQ dbg$k_car_return
;  1239          1362  2          AND .tot_chars_read LSS .input_desc[dsc$w_length]
;  1240          1363  2          DO
;  1241          1364  3          BEGIN
;  1242          1365  3
;  1243          1366  3          ! If we are inside of quotes, the only terminator is a quote (').
;  1244          1367  3          ! Doubled quotes (''), however, must be undoubled.
;  1245          1368  3          !
;  1246          1369  3          IF .quote_flag
;  1247          1370  3          THEN
;  1248          1371  4              BEGIN
;  1249          1372  4
;  1250          1373  4              ! Check for ''
;  1251          1374  4              !
;  1252          1375  4              IF .char EQL dbg$k_dblquote
;  1253          1376  4              THEN
;  1254          1377  4
;  1255          1378  4                  ! Check for doubled quotes
;  1256          1379  4                  !
;  1257          1380  4                  IF .prev_char EQL dbg$k_dblquote
;  1258          1381  4                  THEN
;  1259          1382  5                      BEGIN
;  1260          1383  5
;  1261          1384  5                      ! Mark the doubled quote with a 0.
;  1262          1385  5                      !
;  1263          1386  5                      (.prev_input_ptr)<0,8,0> = 0;
;  1264          1387  5                      chars_read = .chars_read - 1;
;  1265          1388  5                      END
;  1266          1389  4                  ELSE
```

```
;1267     1390  4                    ! Must be single quote. Since this terminates the
;1268     1391  4                    ! string, exit the loop.
;1269     1392  4                    !
;1270     1393  4                    EXITLOOP;
;1271     1394  4                END
;1272     1395  3            ELSE
;1273     1396  3
;1274     1397  3                ! This else clause is reached when quote_flag is false.
;1275     1398  3                !
;1276     1399  4                BEGIN
;1277     1400  4
;1278     1401  4                ! Comma is a terminator if not inside of quotes.
;1279     1402  4                !
;1280     1403  4                IF .char EQL dbg$k_comma
;1281     1404  4                THEN
;1282     1405  5                    BEGIN
;1283     1406  5                    ! Decrement tot_chars_read to back up to before the comma.
;1284     1407  5                    tot_chars_read = .tot_chars_read - 1;
;1285     1408  5                    EXITLOOP;
;1286     1409  4                    END;
;1287     1410  4
;1288     1411  4                ! Right paren is a terminator if not inside of quotes
;1289     1412  4                ! and paren_flag is true.
;1290     1413  4                !
;1291     1414  4                IF .paren_flag AND .char EQL dbg$k_right_parenthesis
;1292     1415  4                THEN
;1293     1416  5                    BEGIN
;1294     1417  5                    ! Decrement tot_chars_read to back up to before the comma.
;1295     1418  5                    tot_chars_read = .tot_chars_read - 1;
;1296     1419  5                    EXITLOOP;
;1297     1420  4                    END;
;1298     1421  3                END;
;1299     1422  3
;1300     1423  3                ! Set up for next time around loop.
;1301     1424  3                !
;1302     1425  3                prev_char = .char;
;1303     1426  3                get_char;
;1304     1427  3                chars_read = .chars_read + 1;
;1305     1428  2                END;
;1306     1429  2
;1307     1430  2        ! If no characters were read, then we failed to parse a parameter.
;1308     1431  2        !
;1309     1432  2        IF .chars_read EQL 0
;1310     1433  2        THEN
;1311     1434  2            RETURN FALSE;
;1312     1435  2
;1313     1436  2        ! Allocate space for the result.
;1314     1437  2        !
;1315     1438  2        result = dbg$get_memory ((.chars_read+1+3)/%UPVAL);
;1316     1439  2
;1317     1440  2        ! Copy the parameter into the allocated area.
;1318     1441  2        !
;1319     1442  2        result[0] = .chars_read;
;1320     1443  2        output_ptr = result[1];
;1321     1444  2        INCR i FROM 1 TO .chars_read DO
;1322     1445  3            BEGIN
;1323     1446  3            char = ch$rchar_a (saved_input_ptr);
```

```
;  1324          1447  3          WHILE .char EQL 0 DO
;  1325          1448  3              char = ch$rchar_a (saved_input_ptr);
;  1326          1449  3          ch$wchar_a (.char, output_ptr);
;  1327          1450  2          END;
;  1328          1451  2
;  1329          1452  2      ! Fill in result_addr.
;  1330          1453  2      !
;  1331          1454  2      .result_addr = .result;
;  1332          1455  2
;  1333          1456  2      ! Update input descriptor.
;  1334          1457  2      !
;  1335          1458  2      input_desc [dsc$w_length] = .input_desc [dsc$w_length] - .tot_chars_read;
;  1336          1459  2      input_desc [dsc$a_pointer] =
;  1337          1460  2          ch$plus (.input_desc[dsc$a_pointer], .tot_chars_read);
;  1338          1461  2
;  1339          1462  2      RETURN TRUE;
;  1340          1463  2
;  1341          1464  1      END; ! get_param


                                01FC 00000 GET_PARAM:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8                ;  1276
                          54    04   AC  D0 00002          MOVL    INPUT_DESC, R4            ;  1329
                          50    04   A4  D0 00006          MOVL    4(R4), INPUT_PTR          ;
                                     55  D4 0000A          CLRL    TOT_CHARS_READ            ;  1330
                                     52  D4 0000C          CLRL    CHARS_READ                ;  1331
                          57         20  90 0000E          MOVB    #32, PREV_CHAR            ;  1332
                          56         20  90 00011          MOVB    #32, CHAR                 ;  1333
                          20         56  91 00014 1$:      CMPB    CHAR, #32                 ;  1337
                                     11  12 00017          BNEQ    2$                        ;
                    55          64   10  00  ED 00019      CMPZV   #0, #16, (R4), TOT_CHARS_READ ;  1338
                                     0A  15 0001E          BLEQ    2$                        ;
                          51         50  D0 00020          MOVL    INPUT_PTR, PREV_INPUT_PTR ;  1340
                          56         80  90 00023          MOVB    (INPUT_PTR)+, CHAR        ;
                                     55  D6 00026          INCL    TOT_CHARS_READ            ;
                                     EA  11 00028          BRB     1$                        ;  1337
                          22         56  91 0002A 2$:      CMPB    CHAR, #34                 ;  1346
                                     0F  12 0002D          BNEQ    3$                        ;
                                     61  94 0002F          CLRB    (PREV_INPUT_PTR)          ;  1349
                          58         01  90 00031          MOVB    #1, QUOTE_FLAG            ;   350
                          51         50  D0 00034          MOVL    INPUT_PTR, PREV_INPUT_PTR ;
                          56         80  90 00037          MOVB    (INPUT_PTR)+, CHAR        ;
                                     55  D6 0003A          INCL    TOT_CHARS_READ            ;
                                     02  11 0003C          BRB     4$                        ;  1346
                                     58  94 0003E 3$:      CLRB    QUOTE_FLAG                ;  1354
                          53    FF   A0  9E 00040 4$:      MOVAB   -1(R0), SAVED_INPUT_PTR   ;  1356
                          0D         56  91 00044 5$:      CMPB    CHAR, #13                 ;  1361
                                     3B  13 00047          BEQL    9$                        ;
                    55          64   10  00  ED 00049      CMPZV   #0, #16, (R4), TOT_CHARS_READ ;  1362
                                     34  15 0004E          BLEQ    9$                        ;
                          10         58  E9 00050          BLBC    QUOTE_FLAG, 6$            ;  1369
                          22         56  91 00053          CMPB    CHAR, #34                 ;  1375
                                     1D  12 00056          BNEQ    8$                        ;
                          22         57  91 00058          CMPB    PREV_CHAR, #34            ;  1380
```

```
                                 27  12 0005B          BNEQ    9$                                      ;
                                 61  94 0005D          CLRB    (PREV_INPUT_PTR)                        ; 1386
                                 52  D7 0005F          DECL    CHARS_READ                              ; 1387
                                 12  11 00061          BRB     8$                                      ; 1380
                          2C     56  91 00063 6$:      CMPB    CHAR, #44                               ; 1403
                                 09  13 00066          BEQL    7$                                      ;
                          09 08  AC  E9 00068          BLBC    PAREN_FLAG, 8$                          ; 1414
                          29     56  91 0006C          CMPB    CHAR, #41                               ;
                                 04  12 0006F          BNEQ    8$                                      ;
                                 55  D7 00071 7$:      DECL    TOT_CHARS_READ                          ; 1418
                                 0F  11 00073          BRB     9$                                      ; 1416
                          57     56  90 00075 8$:      MOVB    CHAR, PREV_CHAR                         ; 1425
                          51     50  D0 00078          MOVL    INPUT_PTR, PREV_INPUT_PTR               ;
                          56     80  90 0007B          MOVB    (INPUT_PTR)+, CHAR                      ;
                                 55  D6 0007E          INCL    TOT_CHARS_READ                          ;
                                 52  D6 00080          INCL    CHARS_READ                              ; 1427
                                 C0  11 00082          BRB     5$                                      ; 1361
                                 52  D5 00084 9$:      TSTL    CHARS_READ                              ; 1432
                                 38  13 00086          BEQL    12$                                     ;
                          50     04  A2 00088          MOVAB   4(R2), R0                               ; 1438
                       7E 50     04  C7 0008C          DIVL3   #4, R0, -(SP)                           ;
                00000000G 00     01  FB 00090          CALLS   #1, DBG$GET_MEMORY                      ;
                          57     50  D0 00097          MOVL    R0, RESULT                              ;
                          67     52  90 0009A          MOVB    CHARS_READ, (RESULT)                    ; 1442
                          50     01  A7 9E 0009D       MOVAB   1(R7), OUTPUT_PTR                       ; 1443
                                 51  D4 000A1          CLRL    I                                       ; 1444
                                 08  11 000A3          BRB     11$                                     ;
                          56     83  90 000A5 10$:     MOVB    (SAVED_INPUT_PTR)+, CHAR                ; 1446
                                 FB  13 000A8          BEQL    10$                                     ; 1447
                          80     56  90 000AA          MOVB    CHAR, (OUTPUT_PTR)+                     ; 1449
                       F4 51     52  F3 000AD 11$:     AOBLEQ  CHARS_READ, I, 10$                      ; 1444
                    0C BC        57  D0 000B1          MOVL    RESULT, @RESULT_ADDR                    ; 1454
                          64     55  A2 000B5          SUBW2   TOT_CHARS_READ, (R4)                    ; 1458
                    04 A4        55  C0 000B8          ADDL2   TOT_CHARS_READ, 4(R4)                   ; 1460
                          50     01  D0 000BC          MOVL    #1, R0                                  ; 1462
                                     04 000BF          RET                                             ;
                                 50  D4 000C0 12$:     CLRL    R0                                      ; 1464
                                     04 000C2          RET                                             ;

; Routine Size:  195 bytes,    Routine Base:  DBG$CODE + 068E
```

```
;  1343          1465  1 GLOBAL ROUTINE DBG$NPARSE_EXIT (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;  1344          1466  1
;  1345          1467  1 !++
;  1346          1468  1 ! FUNCTIONAL DESCRIPTION:
;  1347          1469  1 !
;  1348          1470  1 !       This routine is the parse network for the EXIT command. Since the EXIT
;  1349          1471  1 !       verb has already been recognized in dbg$nparse_cmd, and nothing else may
;  1350          1472  1 !       follow that keyword, simply return success.
;  1351          1473  1 !
;  1352          1474  1 ! FORMAL PARAMETERS:
;  1353          1475  1 !
;  1354          1476  1 !       input_desc -            string descriptor for remainder of input line
;  1355          1477  1 !
;  1356          1478  1 !       verb_node -             command verb node - the head of the command execution
;  1357          1479  1 !                               tree.
;  1358          1480  1 !
;  1359          1481  1 !       message_vect -          the address of a longword to contain the address
;  1360          1482  1 !                               of a standard message argument vector
;  1361          1483  1 !
;  1362          1484  1 ! IMPLICIT INPUTS:
;  1363          1485  1 !
;  1364          1486  1 !       NONE
;  1365          1487  1 !
;  1366          1488  1 ! IMPLICIT OUTPUTS:
;  1367          1489  1 !
;  1368          1490  1 !       NONE -    The command execution tree is unaffected by this network.
;  1369          1491  1 !
;  1370          1492  1 ! ROUTINE VALUE:        unsigned integer lomgword completion code
;  1371          1493  1 !
;  1372          1494  1 ! COMPLETION CODES:
;  1373          1495  1 !
;  1374          1496  1 !       sts$k_success (1) -     Always returned as the value of this routine
;  1375          1497  1 !
;  1376          1498  1 ! SIDE EFFECTS:
;  1377          1499  1 !
;  1378          1500  1 !       NONE
;  1379          1501  1 !
;  1380          1502  1 !--
;  1381          1503  1
;  1382          1504  2     BEGIN
;  1383          1505  2
;  1384          1506  2     RETURN sts$k_success;
;  1385          1507  2
;  1386          1508  1     END;                            ! End of dbg$nparse_exit


                                                  0000 00000          .ENTRY  DBG$NPARSE_EXIT, Save nothing                   ; 1465
                                        50      01  D0 00002          MOVL    #1, R0                                          ; 1506
                                                    04 00005          RET                                                     ; 1508

; Routine Size:  6 bytes,    Routine Base:  DBG$CODE + 0751

; 1387          1509  1
```

```
;1389     1510  1 GLOBAL ROUTINE DBG$NEXECUTE_EXIT (VERB_NODE, MESSAGE_VECT) =
;1390     1511  1
;1391     1512  1 !++
;1392     1513  1 ! FUNCTIONAL DESCRIPTION:
;1393     1514  1 !
;1394     1515  1 !       This routine is the execution network for the EXIT command. An EXIT issued
;1395     1516  1 !       from an indirect command file or DO action buffer means to cease taking
;1396     1517  1 !       commands from the present buffer. In this case, a buffer is removed from
;1397     1518  1 !       the command input stream. An EXIT issued from the terminal means to exit
;1398     1519  1 !       the debugger.
;1399     1520  1 !
;1400     1521  1 ! FORMAL PARAMETERS:
;1401     1522  1 !
;1402     1523  1 !       verb_node -     The head of the command excution tree which in the case
;1403     1524  1 !                       of EXIT consists only of this one node. The value of the
;1404     1525  1 !                       verb node is used only to direct processing to this routine
;1405     1526  1 !                       and is not used by dbg$nexecute_exit itself.
;1406     1527  1 !
;1407     1528  1 !       message_vect -  The address of a longword to contain the address of a
;1408     1529  1 !                       message argument vector
;1409     1530  1 !
;1410     1531  1 ! IMPLICIT INPUTS:
;1411     1532  1 !
;1412     1533  1 !
;1413     1534  1 !       dbg$gb_def_out -                        "   "   "     output configuration structure
;1414     1535  1 !
;1415     1536  1 !       dbg$gl_cishead -                        "   "   "     head of command input stream
;1416     1537  1 !
;1417     1538  1 ! IMPLICIT OUTPUTS:
;1418     1539  1 !
;1419     1540  1 !       The above data structures may be altered.
;1420     1541  1 !
;1421     1542  1 !       On failure, a message argument vector is returned.
;1422     1543  1 !
;1423     1544  1 ! ROUTINE VALUE:
;1424     1545  1 !
;1425     1546  1 !       An unsigned integer longword completion code.
;1426     1547  1 !
;1427     1548  1 ! COMPLETION CODES:
;1428     1549  1 !
;1429     1550  1 !       sts$k_severe (4) -      The command could not be executed.
;1430     1551  1 !
;1431     1552  1 !       sts$k_success (1) -     The command was executed.
;1432     1553  1 !
;1433     1554  1 ! SIDE EFFECTS:
;1434     1555  1 !
;1435     1556  1 !       This routine may force the exit of the program image. Nodes may be
;1436     1557  1 !       removed from the command input stream.
;1437     1558  1 !
;1438     1559  1 !--
;1439     1560  1
;1440     1561  2     BEGIN
;1441     1562  2
;1442     1563  2     LOCAL
;1443     1564  2         count,                      ! Temporary counter for number of levels to exit.
;1444     1565  2         link: REF cis$link;! Pointer to a CIS link.
;1445     1566  2
```

```
;1446     1567  2      !++
;1447     1568  2      ! When an EXIT is entered to the debugger we want to 'go up a level' to
;1448     1569  2      ! the next link in the command input stream. This is done for indirect command
;1449     1570  2      ! files by removing the links for the RAB as well as the associated buffer.
;1450     1571  2      !--
;1451     1572  2
;1452     1573  2      ! First walk past any layers of IF, WHILE, or REPEAT that we may be in.
;1453     1574  2      !
;1454     1575  2      count = 0;
;1455     1576  2      link = .dbg$gl_cishead;
;1456     1577  2      WHILE  .link [cis$b_input_type] EQL dbg$k_cis_if
;1457     1578  2          OR .link [cis$b_input_type] EQL dbg$k_cis_while
;1458     1579  2          OR .link [cis$b_input_type] EQL dbg$k_cis_repeat
;1459     1580  2          OR .link [cis$b_input_type] EQL dbg$k_cis_for
;1460     1581  3          DO BEGIN
;1461     1582  3          count = .count + 1;
;1462     1583  3          link = .link [cis$a_next_link]
;1463     1584  2          END;
;1464     1585  2
;1465     1586  2      ! Now case on the type of buffer we are looking at.
;1466     1587  2      !
;1467     1588  2      CASE .link[cis$b_input_type] FROM dbg$k_cis_minimum TO dbg$k_cis_maximum OF
;1468     1589  2          SET
;1469     1590  2
;1470     1591  2          ! Ordinary input buffer.
;1471     1592  2          !
;1472     1593  2          [dbg$k_cis_inpbuf] :
;1473     1594  3                  BEGIN
;1474     1595  3                  LOCAL
;1475     1596  3                      prev_link : REF cis$link;
;1476     1597  3
;1477     1598  3                  ! Look at the CIS above the input buffer CIS.
;1478     1599  3                  !
;1479     1600  3                  prev_link = .link [cis$a_next_link];
;1480     1601  3                  CASE .prev_link [cis$b_input_type] FROM dbg$k_cis_minimum
;1481     1602  3                                                    TO dbg$k_cis_maximum OF
;1482     1603  3                      SET
;1483     1604  3
;1484     1605  3                      ! If it is cis_dbg$input then we are taking commands
;1485     1606  3                      ! from DBG$INPUT. In this case we want to exit DEBUG.
;1486     1607  3                      !
;1487     1608  3                      [dbg$k_cis_dbg$input] :
;1488     1609  4                          BEGIN
;1489     1610  4
;1490     1611  4                          ! We are accepting command from DBG$INPUT, so just
;1491     1612  4                          ! exit to the system CLI.
;1492     1613  4                          !
;1493     1614  4
;1494     1615  4                          IF .dbg$gb_def_out [out_log]
;1495     1616  4                          THEN
;1496     1617  4                              $CLOSE (FAB = dbg$gl_logfab);       ! Close log file
;1497     1618  4                          dbg$gb_def_out[out_log] = FALSE;
;1498     1619  4                          dbg$gv_control[dbg$v_control_user] = True;
;1499     1620  4                          dbg$gv_control[dbg$v_control_exit] = True;
;1500     1621  4
;1501     1622  4                          ! Call the SMG exit handler which restores the
;1502     1623  4                          ! terminal back to application or numeric mode.
```

```
;  1503          1624  4                    ! depending on what it was at entry.
;  1504          1625  4                    !
;  1505          1626  4                    IF .dbg$gl_smg_exit_handler NEQ 0
;  1506          1627  4                    THEN
;  1507          1628  4                        (.dbg$gl_smg_exit_handler)();
;  1508          1629  4
;  1509          1630  4                    $EXIT (code = .dbg$gl_exit_status OR sts$m_inhib_msg);
;  1510          1631  3                    END;
;  1511          1632  3
;  1512          1633  3
;  1513          1634  3                ! If it is a RAB, we are in an indirect command file.
;  1514          1635  3                !
;  1515          1636  3                [dbg$k_cis_rab] :
;  1516          1637  4                    BEGIN
;  1517          1638  4
;  1518          1639  4                    ! Remove all CIS buffers we have walked past.
;  1519          1640  4                    !
;  1520          1641  4                    INCR i FROM 1 to .count+2 DO
;  1521          1642  4                        IF NOT dbg$ncis_remove (TRUE, .message_vect)
;  1522          1643  4                        THEN
;  1523          1644  4                            RETURN sts$k_severe;
;  1524          1645  4                    RETURN sts$k_success;
;  1525          1646  3                    END;
;  1526          1647  3
;  1527          1648  3                ! We can get inpbuf's nested inside of each other
;  1528          1649  3                ! when executing DO command lists.
;  1529          1650  3                !
;  1530          1651  3                [INRANGE] :
;  1531          1652  4                    BEGIN
;  1532          1653  4
;  1533          1654  4                    ! Remove all CIS buffers we have walked past.
;  1534          1655  4                    !
;  1535          1656  4                    INCR i FROM 1 to .count+1 DO
;  1536          1657  4                        IF NOT dbg$ncis_remove (TRUE, .message_vect)
;  1537          1658  4                        THEN
;  1538          1659  4                            RETURN sts$k_severe;
;  1539          1660  4                    RETURN sts$k_success;
;  1540          1661  3                    END;
;  1541          1662  3
;  1542          1663  3
;  1543          1664  3                    ! Anything else is a DEBUG error.
;  1544          1665  3                    !
;  1545          1666  3                    [OUTRANGE] :
;  1546          1667  4                        BEGIN
;  1547          1668  4                        $DBG_ERROR('DBGATSIGN\DBG$NEXECUTE_EXIT');
;  1548          1669  3                        END;
;  1549          1670  3
;  1550          1671  3                    TES;
;  1551          1672  3
;  1552          1673  2                END; ! cis_inpbuf case
;  1553          1674  2
;  1554          1675  2            ! If we are accepting commands from an action buffer
;  1555          1676  2            ! we go up a level by removing that buffer (and any
;  1556          1677  2            ! buffers that may be present for loops).
;  1557          1678  2            !
;  1558          1679  2            [dbg$k_cis_acbuf,
;  1559          1680  2             dbg$k_cis_screen] :
```

```
;  1560          1681  3                 BEGIN
;  1561          1682  3                 INCR i FROM 1 TO .count+1 DO
;  1562          1683  3                     IF NOT dbg$ncis_remove (TRUE, .message_vect)
;  1563          1684  3                     THEN
;  1564          1685  3                         RETURN sts$k_severe;
;  1565          1686  2                 END;
;  1566          1687  2
;  1567          1688  2             ! Anything else is a DEBUG internal coding error.
;  1568          1689  2             !
;  1569          1690  2             [INRANGE,
;  1570          1691  2              OUTRANGE] :
;  1571          1692  3                     BEGIN
;  1572          1693  3                     $DBG_ERROR('DBGATSIGN\DBG$NEXECUTE_EXIT');
;  1573          1694  2                     END;
;  1574          1695  2
;  1575          1696  2             TES;
;  1576          1697  2
;  1577          1698  2         RETURN sts$k_success;
;  1578          1699  2
;  1579          1700  1         END;                          ! End of dbg$nexecute_exit


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

24  47  42  44  5C  4E  47  49  53  54  41  47  42  44  1B  000B5 P.AAR:  .ASCII  <27>\DBGATSIGN\<92>\DBG$NEXECUTE_EXIT\      ;
        54  49  58  45  5F  45  54  55  43  45  58  45  4E  000C4                                                              ;
24  47  42  44  5C  4E  47  49  53  54  41  47  42  44  1B  000D1 P.AAS:  .ASCII  <27>\DBGATSIGN\<92>\DBG$NEXECUTE_EXIT\      ;
        54  49  58  45  5F  45  54  55  43  45  58  45  4E  000E0                                                              ;

                                                                .EXTRN  SYS$CLOSE, SYS$EXIT

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

                                                  003C 00000    .ENTRY  DBG$NEXECUTE_EXIT, Save R2,R3,R4,R5                 ; 1510
                                  55 00000000G    00   9E 00002 MOVAB   DBG$GB_DEF_OUT, R5                                 ;
                                  54 00000000G    00   9E 00009 MOVAB   DBG$NCIS_REMOVE, R4                               ;
                                                  52   D4 00010 CLRL    COUNT                                           ; 1575
                                  53 00000000G    00   D0 00012 MOVL    DBG$GL_CISHEAD, LINK                            ; 1576
                                  06        02    A3   91 00019 1$:     CMPB    2(LINK), #6                             ; 1577
                                                  12   13 0001D BEQL    2$                                              ;
                                  05        02    A3   91 0001F CMPB    2(LINK), #5                                     ; 1578
                                                  0C   13 00023 BEQL    2$                                              ;
                                  04        02    A3   91 00025 CMPB    2(LINK), #4                                     ; 1579
                                                  06   13 00029 BEQL    2$                                              ;
                                  07        02    A3   91 0002B CMPB    2(LINK), #7                                     ; 1580
                                                  08   12 0002F BNEQ    3$                                              ;
                                                  52   D6 00031 2$:     INCL    COUNT                                   ; 1582
                                  53        08    A3   D0 00033 MOVL    8(LINK), LINK                                   ; 1583
                                                  E0   11 00037 BRB     1$                                              ;
                    08            00        02    A3   8F 00039 3$:     CASEB   2(LINK), #0, #8                         ; 1588
0087                001A          0012            0012     0003E 4$:     .WORD   5$-4$,-                                 ;
0012                0012          0012            0012     00046         5$-4$,-                                         ;
                                                    00B7     0004E         6$-4$,-                                         ;
                                                                           19$-4$,-                                        ;
                                                                           5$-4$,-                                         ;
                                                                           5$-4$,-                                         ;
```

```
                                                                5$-4$,-
                                                                5$-4$,-
                                                                19$-4$
                                00000000'   EF  9F 00050 5$:    PUSHAB  P.AAS                                   ;1693
                                            21  11 00056       BRB     8$
                             50       08    A3  D0 00058 6$:    MOVL    8(LINK), PREV_LINK                      ;1600
                          08 00       02    A0  8F 0005C       CASEB   2(PREV_LINK), #0, #8                    ;1601
                007D         0065         0029    00061 7$:    .WORD   9$-7$,-
                007D         007D         007D    00067        12$-7$,-
                                          007D    00071        15$-7$,-
                                                               15$-7$,-
                                                               15$-7$,-
                                                               15$-7$,-
                                                               15$-7$,-
                                                               15$-7$
                                00000000'   EF  9F 00073       PUSHAB  P.AAR                                   ;1668
                                            01  DD 00079 8$:    PUSHL   #1
                                00028362    8F  DD 0007B       PUSHL   #164706
                   00000000G 00             03  FB 00081       CALLS   #3, LIB$SIGNAL
                                            69  11 00088       BRB     18$                                     ;1601
                             0D             65  E9 0008A 9$:    BLBC    DBG$GB_DEF_OUT, 10$                     ;1615
                                00000000G 00 9F  0008D       PUSHAB  DBG$GL_LOGFAB                           ;1617
                   00000000G 00             01  FB 00093       CALLS   #1, SYS$CLOSE
                                            65  94 0009A 10$:   CLRB    DBG$GB_DEF_OUT                          ;1618
                   00000000G 00    0110     8F  A8 0009C       BISW2   #272, DBG$GV_CONTROL                    ;1619
                             50 00000000G 00 D0  000A5       MOVL    DBG$GL_SMG_EXIT_HANDLER, R0             ;1626
                                            03  13 000AC       BEQL    11$
                             60             00  FB 000AE       CALLS   #0, (R0)                                ;1628
                7E 00000000G 00 10000000    8F  C9 000B1 11$:   BISL3   #268435456, DBG$GL_EXIT_STATUS, -(SP)  ;1630
                   00000000G 00             01  FB 000BD       CALLS   #1, SYS$EXIT
                                            48  11 000C4       BRB     23$                                     ;1601
                          52                02  C0 000C6 12$:   ADDL2   #2, R2                                 ;1641
                                            53  D4 000C9       CLRL    I                                       ;1642
                                            0B  11 000CB       BRB     14$
                                      08    AC  DD 000CD 13$:   PUSHL   MESSAGE_VECT
                                            01  DD 000D0       PUSHL   #1
                             64             02  FB 000D2       CALLS   #2, DBG$NCIS_REMOVE
                             2E             50  E9 000D5       BLBC    R0, 21$
                F1           53             52  F3 000D8 14$:   AOBLEQ  R2, I, 13$
                                            30  11 000DC       BRB     23$                                     ;1645
                                            52  D6 000DE 15$:   INCL    R2                                      ;1656
                                            53  D4 000E0       CLRL    I                                       ;1657
                                            0B  11 000E2       BRB     17$
                                      08    AC  DD 000E4 16$:   PUSHL   MESSAGE_VECT
                                            01  DD 000E7       PUSHL   #1
                             64             02  FB 000E9       CALLS   #2, DBG$NCIS_REMOVE
                             17             50  E9 000EC       BLBC    R0, 21$
                F1           53             52  F3 000EF 17$:   AOBLEQ  R2, I, 16$
                                            19  11 000F3 18$:   BRB     23$                                     ;1660
                                            52  D6 000F5 19$:   INCL    R2                                      ;1682
                                            53  D4 000F7       CLRL    I                                       ;1683
                                            0F  11 000F9       BRB     22$
                                      08    AC  DD 000FB 20$:   PUSHL   MESSAGE_VECT
                                            01  DD 000FE       PUSHL   #1
                             64             02  FB 00100       CALLS   #2, DBG$NCIS_REMOVE
                             04             50  E8 00103       BLBS    R0, 22$
```

```
                          50        04  D0 00106 21$:   MOVL    #4, R0                          ; 1685
                                        04 00109         RET                                     ;
                       ED 53        52  F3 0010A 22$:   AOBLEQ  R2, I, 20$                      ; 1683
                          50        01  D0 0010E 23$:   MOVL    #1, R0                          ; 1698
                                        04 00111         RET                                     ; 1700

; Routine Size:  274 bytes,    Routine Base:  DBG$CODE + 0757

; 1580          1701  1
```

```
;  1582          1702  1 GLOBAL ROUTINE DBG$NPARSE_EXITLOOP (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;  1583          1703  1
;  1584          1704  1 !++
;  1585          1705  1 ! FUNCTIONAL DESCRIPTION:
;  1586          1706  1 !
;  1587          1707  1 !       This routine is the parse network for the EXITLOOP command.
;  1588          1708  1 !       A single noun node is constructed containing the integer
;  1589          1709  1 !       argument, if there is one.
;  1590          1710  1 !
;  1591          1711  1 ! FORMAL PARAMETERS:
;  1592          1712  1 !
;  1593          1713  1 !       input_desc -            string descriptor for the remaining input.
;  1594          1714  1 !
;  1595          1715  1 !       verb_node -             command verb node - the head of the command execution
;  1596          1716  1 !                               tree.
;  1597          1717  1 !
;  1598          1718  1 !       message_vect -          the address of a longword to contain the address
;  1599          1719  1 !                               of a standard message argument vector
;  1600          1720  1 !
;  1601          1721  1 ! IMPLICIT INPUTS:
;  1602          1722  1 !
;  1603          1723  1 !       NONE
;  1604          1724  1 !
;  1605          1725  1 ! IMPLICIT OUTPUTS:
;  1606          1726  1 !
;  1607          1727  1 !       A command execution tree is contructed for use by DBG$NEXECUTE_EXITLOOP
;  1608          1728  1 !
;  1609          1729  1 ! ROUTINE VALUE:
;  1610          1730  1 !
;  1611          1731  1 !       An unsigned integer lomgword completion code
;  1612          1732  1 !
;  1613          1733  1 ! COMPLETION CODES:
;  1614          1734  1 !
;  1615          1735  1 !       sts$k_success (1) -     success
;  1616          1736  1 !       sts$k_severe  (4) -     error in input
;  1617          1737  1 !
;  1618          1738  1 ! SIDE EFFECTS:
;  1619          1739  1 !
;  1620          1740  1 !       NONE
;  1621          1741  1 !
;  1622          1742  1 !--
;  1623          1743  1
;  1624          1744  2     BEGIN
;  1625          1745  2
;  1626          1746  2     MAP
;  1627          1747  2         verb_node : REF dbg$verb_node;
;  1628          1748  2
;  1629          1749  2     LOCAL
;  1630          1750  2         noun_node : REF dbg$noun_node;
;  1631          1751  2
;  1632          1752  2     BIND
;  1633          1753  2         dbg$cs_cr = UPLIT BYTE (1, dbg$k_car_return);
;  1634          1754  2
;  1635          1755  2     noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;  1636          1756  2
;  1637          1757  2     verb_node [dbg$l_verb_object_ptr] = .noun_node;
;  1638          1758  2
```

```
;1639          1759  2     IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;1640          1760  2     THEN
;1641          1761  2         noun_node [dbg$l_noun_value] = 1
;1642          1762  2     ELSE
;1643          1763  2         IF NOT dbg$nsave_decimal_integer (.input_desc,
;1644          1764  2                                           noun_node [dbg$l_noun_value],
;1645          1765  2                                           .message_vect)
;1646          1766  2         THEN
;1647          1767  2             RETURN sts$k_severe;
;1648          1768  2
;1649          1769  2     RETURN sts$k_success;
;1650          1770  2
;1651          1771  1     END;                          ! End of dbg$nparse_exitloop


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

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


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

                                          0004 00000            .ENTRY  DBG$NPARSE_EXITLOOP, Save R2            ;1702
                                       04   DD 00002            PUSHL   #4                                      ;1755
                  00000000G  00        01   FB 00004            CALLS   #1, DBG$GET_TEMPMEM                     ;
                             52        50   DO 0000B            MOVL    R0, NOUN_NODE                           ;
                             50   08   AC   DO 0000E            MOVL    VERB_NODE, R0                           ;1757
                        08   A0        52   DO 00012            MOVL    NOUN_NODE, 8(R0)                        ;
                                       01   DD 00016            PUSHL   #1                                      ;1759
                                00000000' EF   9F 00018        PUSHAB  DBG$CS_CR                               ;
                                     04   AC   DD 0001E        PUSHL   INPUT_DESC                              ;
                  00000000G  00             03   FB 00021      CALLS   #3, DBG$NMATCH                          ;
                             05             50   E9 00028      BLBC    R0, 1$                                  ;
                             62             01   DO 0002B      MOVL    #1, (NOUN_NODE)                         ;1761
                                            16   11 0002E      BRB     2$                                      ;
                                       0C   AC   DD 00030 1$:  PUSHL   MESSAGE_VECT                            ;1765
                                            52   DD 00033      PUSHL   NOUN_NODE                               ;1764
                                       04   AC   DD 00035      PUSHL   INPUT_DESC                              ;
                  00000000G  00             03   FB 00038      CALLS   #3, DBG$NSAVE_DECIMAL_INTEGER           ;
                             04             50   E8 0003F      BLBS    R0, 2$                                  ;
                             50             04   DO 00042      MOVL    #4, R0                                  ;1767
                                                 04 00045      RET                                             ;
                             50             01   DO 00046 2$:  MOVL    #1, R0                                  ;1769
                                                 04 00049      RET                                             ;1771

; Routine Size:  74 bytes,    Routine Base:  DBG$CODE + 0869
```

```
;1653     1772  1 GLOBAL ROUTINE DBG$NEXECUTE_EXITLOOP (VERB_NODE, MESSAGE_VECT) =
;1654     1773  1
;1655     1774  1 !++
;1656     1775  1 ! FUNCTIONAL DESCRIPTION:
;1657     1776  1 !
;1658     1777  1 !       This routine is the execution network for the EXITLOOP command.
;1659     1778  1 !       EXITLOOP N takes you out of N levels of loops. This is done
;1660     1779  1 !       by removing the appropriate buffers from the linked list
;1661     1780  1 !       of command buffers.
;1662     1781  1 !
;1663     1782  1 ! FORMAL PARAMETERS:
;1664     1783  1 !
;1665     1784  1 !       verb_node     -  The head of the command excution tree which in the case
;1666     1785  1 !                        of EXITLOOP consists of this verb node, and a single
;1667     1786  1 !                        noun node with the integer argument.
;1668     1787  1 !
;1669     1788  1 !       message_vect  -  The address of a longword to contain the address of a
;1670     1789  1 !                        message argument vector
;1671     1790  1 !
;1672     1791  1 ! IMPLICIT INPUTS:
;1673     1792  1 !
;1674     1793  1 !       dbg$gl_cishead - head of command input stream
;1675     1794  1 !
;1676     1795  1 ! IMPLICIT OUTPUTS:
;1677     1796  1 !
;1678     1797  1 !       The CIS data structure may be altered.
;1679     1798  1 !
;1680     1799  1 !       On failure, a message argument vector is returned.
;1681     1800  1 !
;1682     1801  1 ! ROUTINE VALUE:
;1683     1802  1 !
;1684     1803  1 !       An unsigned integer longword completion code.
;1685     1804  1 !
;1686     1805  1 ! COMPLETION CODES:
;1687     1806  1 !
;1688     1807  1 !       sts$k_severe (4) -      The command could not be executed.
;1689     1808  1 !
;1690     1809  1 !       sts$k_success (1) -     The command was executed.
;1691     1810  1 !
;1692     1811  1 ! SIDE EFFECTS:
;1693     1812  1 !
;1694     1813  1 !       This routine may force the exit of WHILE or REPEAT loops. Nodes may be
;1695     1814  1 !       removed from the command input stream.
;1696     1815  1 !
;1697     1816  1 !--
;1698     1817  1
;1699     1818  2     BEGIN
;1700     1819  2
;1701     1820  2     MAP
;1702     1821  2         verb_node : REF dbg$verb_node;
;1703     1822  2
;1704     1823  2     LOCAL
;1705     1824  2         cis_ptr : REF cis$link,          ! A pointer to a CIS buffer
;1706     1825  2         count,                           ! Local counter of number of cislinks
;1707     1826  2                                          !    to be removed.
;1708     1827  2         init_num_levels,                 ! Number of levels of loop to exit
;1709     1828  2         num_levels,                      ! (copy of) Number of levels of loop to exit
```

```
;1710     1829  2          noun_node : REF dbg$noun_node;  ! Local pointer to the noun node
;1711     1830  2
;1712     1831  2
;1713     1832  2      ! Recover the argument to EXITLOOP.
;1714     1833  2      !
;1715     1834  2      noun_node = .verb_node [dbg$l_verb_object_ptr];
;1716     1835  2      num_levels = .noun_node [dbg$l_noun_value];
;1717     1836  2      init_num_levels = .num_levels;
;1718     1837  2
;1719     1838  2      ! Initialize cis_ptr and count.
;1720     1839  2      !
;1721     1840  2      cis_ptr = .dbg$gl_cishead;
;1722     1841  2      count = 0;
;1723     1842  2
;1724     1843  2      ! Loop past NUM_LEVELS occurences of repeat or while loop buffers.
;1725     1844  2      !
;1726     1845  2      WHILE .num_levels GTR 0 DO
;1727     1846  3          BEGIN
;1728     1847  3
;1729     1848  3          CASE .cis_ptr[cis$b_input_type] FROM dbg$k_cis_minimum TO
;1730     1849  3                                              dbg$k_cis_maximum OF
;1731     1850  3
;1732     1851  3              SET
;1733     1852  3
;1734     1853  3              ! If the top level CIS indicates that we are not inside
;1735     1854  3              ! of a loop, then the EXITLOOP command is an error.
;1736     1855  3              !
;1737     1856  3              [cis_dbg$input,
;1738     1857  3               cis_rab,
;1739     1858  3               cis_inpbuf,
;1740     1859  3               cis_acbuf]:
;1741     1860  4                          BEGIN
;1742     1861  4
;1743     1862  4                          ! If the number of loops encountered so far
;1744     1863  4                          ! is zero, report that we
;1745     1864  4                          ! are not in a loop.
;1746     1865  4                          !
;1747     1866  4                          IF .num_levels EQL .init_num_levels
;1748     1867  4                          THEN
;1749     1868  4                              .message_vect = dbg$nmake_arg_vect (
;1750     1869  4                                  dbg$_notinloop)
;1751     1870  4
;1752     1871  4                          ! If we have encountered loops, but we get here,
;1753     1872  4                          ! then we were
;1754     1873  4                          ! in a loop but the argument to EXITLOOP
;1755     1874  4                          ! was too large.
;1756     1875  4                          !
;1757     1876  4                          ELSE
;1758     1877  4                              .message_vect = dbg$nmake_arg_vect (
;1759     1878  4                                  dbg$_exitarg, 1, .init_num_levels);
;1760     1879  4                          RETURN sts$k_severe;
;1761     1880  3                          END;
;1762     1881  3
;1763     1882  3              ! If we are in an if clause, then we want to peel off that
;1764     1883  3              ! CIS. We increment the count of the number of CIS buffers
;1765     1884  3              ! to remove.
;1766     1885  3              !
```

```
;  1767          1886  3            [cis_if] :
;  1768          1887  4                        BEGIN
;  1769          1888  4                        count = .count + 1;
;  1770          1889  4                        cis_ptr = .cis_ptr [cis$a_next_link];
;  1771          1890  3                        END;
;  1772          1891  3
;  1773          1892  3            ! If the top layer is a loop, increment the count of the
;  1774          1893  3            ! the number of layets of buffers to remove. Decrement
;  1775          1894  3            ! the count of the number of remaining loops.
;  1776          1895  3            !
;  1777          1896  3            [cis_repeat,
;  1778          1897  3             cis_while,
;  1779          1898  3             cis_for] :
;  1780          1899  4                        BEGIN
;  1781          1900  4                        count = .count + 1;
;  1782          1901  4                        num_levels = .num_levels - 1;
;  1783          1902  4                        cis_ptr = .cis_ptr [cis$a_next_link];
;  1784          1903  3                        END;
;  1785          1904  3
;  1786          1905  3            ! Anything else is an error.
;  1787          1906  3            !
;  1788          1907  3            [INRANGE,
;  1789          1908  3             OUTRANGE] :
;  1790          1909  4                        BEGIN
6 ;  1791          1910  4                        $DBG_ERROR('DBGATSIGN\DBG$NEXECUTE_EXITLOOP');
9 ;  1792          1911  3                        END;
  ;  1793          1912  3
0 ;  1794          1913  3            TES;
1 ;  1795          1914  2            END; ! While loop
2 ;  1796          1915  2
3 ;  1797          1916  2        ! Now, if we get to this point without an error, we strip off
7 ;  1798          1917  2        ! the CIS buffers.
  ;  1799          1918  2        !
8 ;  1800          1919  2        INCR i FROM 1 TO .count DO
  ;  1801          1920  2            IF NOT dbg$ncis_remove (TRUE, .message_vect)
0 ;  1802          1921  2            THEN
  ;  1803          1922  2                RETURN sts$k_severe;
  ;  1804          1923  2
7 ;  1805          1924  2        RETURN sts$k_success;
6 ;  1806          1925  2
  ;  1807          1926  1        END;                            ! End of dbg$nexecute_exitloop
9
0


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

6 24 47 42 44 5C 4E 47 49 53 54 41 47 42 44 1F  000EF P.AAU:  .ASCII  <31>\DBGATSIGN\<92>\DBG$NEXECUTE_EXITL\     ;
4     4C 54 49 58 45 5F 45 54 55 43 45 58 45 4E  000FE                                                          ;
6                                 50 4F 4F  0010C           .ASCII  \OOP\                                       ;
1

2
                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0
9
5                                          007C 00000           .ENTRY  DBG$NEXECUTE_EXITLOOP, Save R2,R3,R4,R5,R6  ; 1772
                                 56 00000000G  00  9E 00002     MOVAB   DBG$NMAKE_ARG_VECT, R6                      ;
0                                50       04   AC  D0 00009     MOVL    VERB_NODE, R0                               ; 1834
```

```
                              50        08   A0  D0 0000D          MOVL    8(R0), NOUN_NODE                          :
                              53             60  D0 00011          MOVL    (NOUN_NODE), NUM_LEVELS                   : 1835
                              55             53  D0 00014          MOVL    NUM_LEVELS, INIT_NUM_LEVELS               : 1836
                              52 00000000G   00  D0 00017          MOVL    DBG$GL_CISHEAD, CIS_PTR                   : 1840
                                             54  D4 0001E          CLRL    COUNT                                     : 1841
                                             53  D5 00020 1$:      TSTL    NUM_LEVELS                                : 1845
                                             5F  15 00022          BLEQ    10$                                       :
                  08          00        02   A2  8F 00024          CASEB   2(CIS_PTR), #0, #8                        : 1848
         0029     0029        0029         0029     00029 2$:      .WORD   4$-2$,-                                   :
         0050     004C        0050         0050     00031                  4$-2$,-                                   :
                                           0012     00039                  4$-2$,-                                   :
                                                                           4$-2$,-                                   :
                                                                           8$-2$,-                                   :
                                                                           8$-2$,-                                   :
                                                                           7$-2$,-                                   :
                                                                           8$-2$,-                                   :
                                                                           3$-2$                                     :
                                 00000000'   EF  9F 0003B 3$:      PUSHAB  P.AAU                                     : 1910
                                             01  DD 00041          PUSHL   #1                                        :
                                 00028362    8F  DD 00043          PUSHL   #164706                                   :
                      00000000G  00          03  FB 00049          CALLS   #3, LIB$SIGNAL                            :
                                             CE  11 00050          BRB     1$                                        : 1848
                                 55          53  D1 00052 4$:      CMPL    NUM_LEVELS, INIT_NUM_LEVELS               : 1866
                                             0B  12 00055          BNEQ    5$                                        :
                                 00028E40    8F  DD 00057          PUSHL   #167488                                   : 1868
                                 66          01  FB 0005D          CALLS   #1, DBG$NMAKE_ARG_VECT                    :
                                             0D  11 00060          BRB     6$                                        :
                                             55  DD 00062 5$:      PUSHL   INIT_NUM_LEVELS                           : 1878
                                             01  DD 00064          PUSHL   #1                                        : 1877
                                 00028E48    8F  DD 00066          PUSHL   #167496                                   :
                                 66          03  FB 0006C          CALLS   #3, DBG$NMAKE_ARG_VECT                    :
                           08    BC          50  D0 0006F 6$.      MOVL    R0, @MESSAGE_VECT                         :
                                             21  11 00073          BRB     12$                                       : 1879
                                             54  D6 00075 7$:      INCL    COUNT                                     : 1888
                                             04  11 00077          BRB     9$                                        : 1889
                                             54  D6 00079 8$:      INCL    COUNT                                     : 1900
                                             53  D7 0007B          DECL    NUM_LEVELS                                : 1901
                                 52     08   A2  D0 0007D 9$:      MOVL    8(CIS_PTR), CIS_PTR                       : 1902
                                             9D  11 00081          BRB     1$                                        : 1845
                                             53  D4 00083 10$:     CLRL    I                                         : 1920
                                             13  11 00085          BRB     13$                                       :
                                        08   AC  DD 00087 11$:     PUSHL   MESSAGE_VECT                              :
                                             01  DD 0008A          PUSHL   #1                                        :
                      00000000G  00          02  FB 0008C          CALLS   #2, DBG$NCIS_REMOVE                       :
                                 04          50  E8 00093          BLBS    R0, 13$                                   :
                                 50          04  D0 00096 12$:     MOVL    #4, R0                                    : 1922
                                                 04 00099          RET                                               :
                  E9             53          54  F3 0009A 13$:     AOBLEQ  COUNT, I, 11$                             : 1920
                                 50          01  D0 0009E          MOVL    #1, R0                                    : 1924
                                                 04 000A1          RET                                               : 1926

; Routine Size:  162 bytes,    Routine Base:  DBG$CODE + 08B3
```

```
;1809     1927  1 GLOBAL ROUTINE DBG$NPARSE_DECLARE (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;1810     1928  1 !++
;1811     1929  1 ! FUNCTIONAL DESCRIPTION:
;1812     1930  1 !
;1813     1931  1 !       Parses the DECLARE command. A command execution tree is constructed.
;1814     1932  1 !       This tree contains a noun node for each name that is declared.
;1815     1933  1 !
;1816     1934  1 ! FORMAL PARAMETERS:
;1817     1935  1 !
;1818     1936  1 !       input_desc -                    The present command line input descriptor
;1819     1937  1 !
;1820     1938  1 !       verb_node  -                    The already existing command verb node
;1821     1939  1 !
;1822     1940  1 !       message_vect -                  The address of a longword to contain the
;1823     1941  1 !                                       address of a message argument vector
;1824     1942  1 !
;1825     1943  1 ! IMPLICIT INPUTS:
;1826     1944  1 !
;1827     1945  1 !       NONE
;1828     1946  1 !
;1829     1947  1 ! IMPLICIT OUTPUTS:
;1830     1948  1 !
;1831     1949  1 !       On success, the appropriate command execution tree is constructed.
;1832     1950  1 !
;1833     1951  1 !       On failure, a message argument vector is constructed
;1834     1952  1 !
;1835     1953  1 ! ROUTINE VALUE:
;1836     1954  1 !
;1837     1955  1 !       An unsigned integer longword completion code
;1838     1956  1 !
;1839     1957  1 ! COMPLETION CODES:
;1840     1958  1 !
;1841     1959  1 !       sts$k_severe (4) -              Error in parsing
;1842     1960  1 !
;1843     1961  1 !       sts$k_success (1) -             Successful parse
;1844     1962  1 !
;1845     1963  1 ! SIDE EFFECTS:
;1846     1964  1 !
;1847     1965  1 !       None
;1848     1966  1 !
;1849     1967  1 !--
;1850     1968  1
;1851     1969  2     BEGIN
;1852     1970  2
;1853     1971  2     MAP
;1854     1972  2         INPUT_DESC: REF dbg$stg_desc,
;1855     1973  2         VERB_NODE : REF dbg$verb_node;
;1856     1974  2
;1857     1975  2     BIND
5    ;1858     1976  2         dbg$cs_colon            = UPLIT BYTE (1, dbg$k_colon),
6    ;1859     1977  2         dbg$cs_comma            = UPLIT BYTE (1, dbg$k_comma),
8    ;1860     1978  2         dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
;1861     1979  2         dbg$cs_address          = UPLIT BYTE (7, 'ADDRESS'),
;1862     1980  2         dbg$cs_command          = UPLIT BYTE (7, 'COMMAND'),
;1863     1981  2         dbg$cs_procedure        = UPLIT BYTE (9, 'PROCEDURE'),
;1864     1982  2         dbg$cs_string           = UPLIT BYTE (6, 'STRING'),
;1865     1983  2         dbg$cs_value            = UPLIT BYTE (5, 'VALUE');
```

```
;1866        1984  2
;1867        1985  2      LOCAL
;1868        1986  2          first_time: BYTE,                   ! Flag which is true the first time
;1869        1987  2                                              !    around a loop.
;1870        1988  2          link,                               ! A link in the command execution tree.
;1871        1989  2          noun_node: REF dbg$noun_node,       ! Pointer to a noun node.
;1872        1990  2
;1873        1991  2      ! Set the first_time flag to true for the first time around the loop.
;1874        1992  2      !
;1875        1993  2      first_time = TRUE;
;1876        1994  2
;1877        1995  2      ! Loop through the list of declarations.
;1878        1996  2      !
;1879        1997  2      WHILE TRUE DO
;1880        1998  3          BEGIN
;1881        1999  3
;1882        2000  3          ! Allocate a noun node and link it in.
;1883        2001  3          !
;1884        2002  3          IF .first_time
;1885        2003  3          THEN
;1886        2004  4              BEGIN
;1887        2005  4              link = verb_node [dbg$l_verb_object_ptr];
;1888        2006  4              first_time = FALSE;
;1889        2007  4              END
;1890        2008  3          ELSE
;1891        2009  3              link = noun_node [dbg$l_noun_link];
;1892        2010  3          noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
;1893        2011  3          .link = .noun_node;
;1894        2012  3
;1895        2013  3          ! Read the name being declared.
;1896        2014  3          !
;1897        2015  3          IF NOT dbg$nread_name (.input_desc, noun_node [dbg$l_noun_value],
;1898        2016  3                                 .message_vect)
;1899        2017  3          THEN
;1900        2018  3              RETURN sts$k_severe;
;1901        2019  3
;1902        2020  3          ! Look for a colon
;1903        2021  3          !
;1904        2022  3          IF dbg$nmatch (.input_desc, dbg$cs_colon, 1)
;1905        2023  3          THEN
;1906        2024  3
;1907        2025  3              ! Look for a keyword following the colon.
;1908        2026  3              !
;1909        2027  3              SELECTONE TRUE OF
;1910        2028  3                  SET
;1911        2029  3
;1912        2030  3                  [dbg$nmatch (.input_desc, dbg$cs_address, 1)] :
;1913        2031  3                      noun_node [dbg$l_noun_value2] = define_address;
;1914        2032  3
;1915        2033  3                  [dbg$nmatch (.input_desc, dbg$cs_command, 1)] :
;1916        2034  3                      noun_node [dbg$l_noun_value2] = define_command;
;1917        2035  3
;1918        2036  3                  ! [dbg$nmatch (.input_desc, dbg$cs_procedure, 1)] :
;1919        2037  3                  !     noun_node [dbg$l_noun_value2] = define_procedure;
;1920        2038  3
;1921        2039  3                  ! [dbg$nmatch (.input_desc, dbg$cs_string, 1)] :
;1922        2040  3                  !     noun_node [dbg$l_noun_value2] = define_string;
```

```
;  1923          2041  3
;  1924          2042  3                [dbg$nmatch (.input_desc, dbg$cs_value, 1)] :
;  1925          2043  3                    noun_node [dbg$l_noun_value2] = define_value;
;  1926          2044  3
;  1927          2045  3                ! Anything else following the colon is an error.
;  1928          2046  3                !
;  1929          2047  3                [OTHERWISE] :
;  1930          2048  3                    report_error;
;  1931          2049  3
;  1932          2050  3                TES
;  1933          2051  3
;  1934          2052  3            ! The default type is address.
;  1935          2053  3            !
;  1936          2054  3            ELSE
;  1937          2055  3                noun_node [dbg$l_noun_value2] = define_address;
;  1938          2056  3
;  1939          2057  3            ! Look for the comma which separates declarations.
;  1940          2058  3            !
;  1941          2059  3            IF NOT dbg$nmatch (.input_desc, dbg$cs_comma, 1)
;  1942          2060  3            THEN
;  1943          2061  3
;  1944          2062  3                ! If end-of-line, then just exit the loop.
;  1945          2063  3                !
;  1946          2064  3                IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  1947          2065  3                THEN
;  1948          2066  3                    EXITLOOP
;  1949          2067  3
;  1950          2068  3                ! Anything other than end-of-line or comma is a syntax error.
;  1951          2069  3                !
;  1952          2070  3                ELSE
;  1953          2071  3                    report_error;
;  1954          2072  3
;  1955          2073  2            END; ! While loop
;  1956          2074  2
;  1957          2075  2        ! If we get here, the parse was successful.
;  1958          2076  2        !
;  1959          2077  2        RETURN sts$k_success;
;  1960          2078  2
;  1961          2079  1        END;
;  INFO#250                  L1:2009
;  Referenced LOCAL symbol NOUN_NODE is probably not initialized


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

                                                  3A  01  0010F P.AAV:  .BYTE   1, 58                                   ;
                                                  2C  01  00111 P.AAW:  .BYTE   1, 44                                   ;
                                                  0D  01  00113 P.AAX:  .BYTE   1, 13                                   ;
                                                      07  00115 P.AAY:  .BYTE   7                                       ;
                                  53  53  45  52  44  44  41  00116     .ASCII  \ADDRESS\                                 ;
                                                      07  0011D P.:AZ:  .BYTE   7                                       ;
                                  44  4E  41  4D  4D  4F  43  0011E     .ASCII  \COMMAND\                                 ;
                                                      09  00125 P.ABA:  .BYTE   9                                       ;
                              45  52  55  44  45  43  4F  52  50  00126     .ASCII  \PROCEDURE\                           ;
                                                      06  0012F P.ABB:  .BYTE   6                                       ;
                                      47  4E  49  52  54  53  00130     .ASCII  \STRING\                                  ;
```

```
                                        05  00136 P.ABC:   .BYTE   5                                                        ;
                                    45  55  4C  41  56  00137          .ASCII  \VALUE\                                       ;

                                                        DBG$CS_COLON=       P.AAV
                                                        DBG$CS_COMMA=       P.AAW
                                                        DBG$CS_CR=          P.AAX
                                                        DBG$CS_ADDRESS=     P.AAY
                                                        DBG$CS_COMMAND=     P.AAZ
                                                        DBG$CS_PROCEDURE=   P.ABA
                                                        DBG$CS_STRING=      P.ABB
                                                        DBG$CS_VALUE=       P.ABC

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

                                                  00FC 00000            .ENTRY  DBG$NPARSE_DECLARE, Save R2,R3,R4,R5,R6,R7  ; 1927
                                    57 00000000G  00   9E 00002         MOVAB   DBG$NMATCH, R7                              ;
                                    56 00000000'  EF   9E 00009         MOVAB   DBG$CS_CR, R6                               ;
                                    55            01   90 00010         MOVB    #1, FIRST_TIME                              ; 1993
                                    53        04  AC   D0 00013         MOVL    INPUT_DESC, R3                              ; 2015
                                    09            55   E9 00017 1$:      BLBC    FIRST_TIME, 2$                              ; 2002
                                54        08  AC             08   C1 0001A         ADDL3   #8, VERB_NODE, LINK                 ; 2005
                                              55   94 0001F         CLRB    FIRST_TIME                                  ; 2006
                                              04   11 00021         BRB     3$                                          ; 2002
                                    54        08   A2   9E 00023 2$:      MOVAB   8(R2), LINK                         ; 2009
                                              04   DD 00027 3$:      PUSHL   #4                                          ; 2010
                                00000000G     00   FB 00029         CALLS   #1, DBG$GET_TEMPMEM                         ;
                                    52            50   D0 00030         MOVL    R0, NOUN_NODE                               ;
                                    64            52   D0 00033         MOVL    NOUN_NODE, (LINK)                           ; 2011
                                        0C    AC   DD 00036         PUSHL   MESSAGE_VECT                                ; 2016
                                              52   DD 00039         PUSHL   NOUN_NODE                                   ; 2015
                                              53   DD 0003B         PUSHL   R3                                          ;
                                00000000G     00   FB 0003D         CALLS   #3, DBG$NREAD_NAME                          ;
                                    03            50   E8 00044         BLBS    R0, 4$                                      ;
                                            0097  31   00047         BRW     12$                                         ;
                                              01   DD 0004A 4$:      PUSHL   #1                                          ; 2022
                                          FC  A6   9F 0004C         PUSHAB  DBG$CS_COLON                                ;
                                              53   DD 0004F         PUSHL   R3                                          ;
                                    67            03   FB 00051         CALLS   #3, DBG$NMATCH                      ;
                                    39            50   E9 00054         BLBC    R0, 6$                              ;
                                              01   DD 00057         PUSHL   #1                                          ; 2030
                                          02  A6   9F 00059         PUSHAB  DBG$CS_ADDRESS                              ;
                                              53   DD 0005C         PUSHL   R3                                          ;
                                    67            03   FB 0005E         CALLS   #3, DBG$NMATCH                      ;
                                    01            50   D1 00061         CMPL    R0, #1                              ;
                                              2A   13   00064         BEQL    6$                                          ;
                                              01   DD 00066         PUSHL   #1                                          ; 2033
                                          0A  A6   9F 00068         PUSHAB  DBG$CS_COMMAND                              ;
                                              53   DD 0006B         PUSHL   R3                                          ;
                                    67            03   FB 0006D         CALLS   #3, DBG$NMATCH                      ;
                                    01            50   D1 00070         CMPL    R0, #1                              ;
                                              06   12   00073         BNEQ    5$                                          ;
                                        0C  A2            02   D0 00075         MOVL    #2, 12(NOUN_NODE)                   ; 2034
                                              19   11   00079         BRB     7$                                          ;
                                              01   DD 0007B 5$:      PUSHL   #1                                          ; 2042
                                          23  A6   9F 0007D         PUSHAB  DBG$CS_VALUE                                ;
                                              53   DD 00080         PUSHL   R3                                          ;
```

```
                67              03  FB 00082          CALLS   #3, DBG$NMATCH                          ;
                01              50  D1 00085          CMPL    R0, #1                                  ;
                                26  12 00088          BNEQ    9$                                      ;
            0C  A2              05  D0 0008A          MOVL    #5, 12(NOUN_NODE)                       ; 2043
                                04  11 0008E          BRB     7$                                      ;
            0C  A2              01  D0 00090 6$:      MOVL    #1, 12(NOUN_NODE)                       ; 2055
                                01  DD 00094 7$:      PUSHL   #1                                      ; 2059
                        FE      A6  9F 00096          PUSHAB  DBG$CS_COMMA                            ;
                                53  DD 00099          PUSHL   R3                                      ;
                67              03  FB 0009B          CALLS   #3, DBG$NMATCH                          ;
                03              50  E9 0009E          BLBC    R0, 8$                                  ;
                            FF73  31 000A1          BRW     1$                                      ;
                                01  DD 000A4 8$:      PUSHL   #1                                      ; 2064
                        0048    8F  BB 000A6          PUSHR   #^M<R3,R6>                              ;
                67              03  FB 000AA          CALLS   #3, DBG$NMATCH                          ;
                35              50  E8 000AD          BLBS    R0, 13$                                 ;
                                01  DD 000B0 9$:      PUSHL   #1                                      ; 2070
                        0048    8F  BB 000B2          PUSHR   #^M<R3,R6>                              ;
                67              03  FB 000B6          CALLS   #3, DBG$NMATCH                          ;
                0F              50  E9 000B9          BLBC    R0, 10$                                 ;
                    000280D0    8F  DD 000BC          PUSHL   #164048                                 ;
    00000000G   00              01  FB 000C2          CALLS   #1, DBG$NMAKE_ARG_VECT                  ;
                                12  11 000C9          BRB     11$                                     ;
                                53  DD 000CB 10$:      PUSHL   R3                                      ;
    00000000G   00              01  FB 000CD          CALLS   #1, DBG$NNEXT_WORD                      ;
                                50  DD 000D4          PUSHL   R0                                      ;
    00000000G   00              01  FB 000D6          CALLS   #1, DBG$NSYNTAX_ERROR                   ;
            0C  BC              50  D0 000DD 11$:      MOVL    R0, @MESSAGE_VECT                       ;
                50              04  D0 000E1 12$:      MOVL    #4, R0                                  ;
                                    04 000E4          RET                                             ;
                50              01  D0 000E5 13$:      MOVL    #1, R0                                  ; 2077
                                    04 000E8          RET                                             ; 2079

; Routine Size:  233 bytes,    Routine Base:  DBG$CODE + 0955
```

```
;1963     2080  1 GLOBAL ROUTINE DBG$NEXECUTE_DECLARE (VERB_NODE, MESSAGE_VECT) =
;1964     2081  1
;1965     2082  1 !++
;1966     2083  1 ! FUNCTIONAL DESCRIPTION:
;1967     2084  1 !
;1968     2085  1 !       This routine executes the DECLARE command. This is done by
;1969     2086  1 !       retrieving the strings that were passed in at a, and binding
;1970     2087  1 !       them to the names being declared.
;1971     2088  1 !
;1972     2089  1 ! FORMAL PARAMETERS:
;1973     2090  1 !
;1974     2091  1 !       verb_node -                     The command verb node which is the start
;1975     2092  1 !                                       of the command execution tree
;1976     2093  1 !
;1977     2094  1 !       message_vect -                  The address of a longword to contain the
;1978     2095  1 !                                       address of a message argument vector
;1979     2096  1 !
;1980     2097  1 ! IMPLICIT INPUTS:
;1981     2098  1 !
;1982     2099  1 !       NONE
;1983     2100  1 !
;1984     2101  1 ! IMPLICIT OUTPUTS:
;1985     2102  1 !
;1986     2103  1 !       The local DEFINE list for the current procedure is modified.
;1987     2104  1 !       On failure, a message argument vector is returned
;1988     2105  1 !
;1989     2106  1 ! ROUTINE VALUE:
;1990     2107  1 !
;1991     2108  1 !       An unsigned integer longword completion code
;1992     2109  1 !
;1993     2110  1 ! COMPLETION CODES:
;1994     2111  1 !
;1995     2112  1 !       sts$k_success (1) -     Indicates the file was created and connected
;1996     2113  1 !
;1997     2114  1 !       sts$k_severe (4) -      Failure, file not created. Message argument vector returned
;1998     2115  1 !
;1999     2116  1 ! SIDE EFFECTS:
;2000     2117  1 !
;2001     2118  1 !       None.
;2002     2119  1 !--
;2003     2120  1
;2004     2121  2     BEGIN
;2005     2122  2
;2006     2123  2     MAP
;2007     2124  2         VERB_NODE       : REF dbg$verb_node;
;2008     2125  2
;2009     2126  2     BIND
;2010     2127  2         dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
;2011     2128  2         dbg$cs_left_paren       = UPLIT BYTE (1, dbg$k_left_parenthesis);
;2012     2129  2
;2013     2130  2     LOCAL
;2014     2131  2         global_flag,                        ! Flag that is returned from dbg$def_sym_find
;2015     2132  2         input_desc: dbg$stg_desc,           ! String descriptor for use with
;2016     2133  2                                             !    AE] and E].
;2017     2134  2         len,                                ! Length of string
;2018     2135  2         new_string: REF VECTOR[,BYTE],      ! Pointer to char string
;2019     2136  2         noun_node: REF dbg$noun_node,       ! Points to a noun node
```

```
;  2020          2137  2          param_name,                         ! Points to a string with "%Pi"
;  2021          2138  2          replaced_flag,                      ! Flag returned from dbg$def_sym_add
;  2022          2139  2          returned_kind,                      ! Kind returned from dbg$def_sym_find
;  2023          2140  2          returned_value:REF VECTOR[,BYTE],   ! Value returned from dbg$def_sym_find
;  2024          2141  2          symid_list,                         ! Points to a symid list
;  2025          2142  2          value;                              ! Points to the "value" of the symbol
;  2026          2143  2
;  2027          2144  2      ! Obtain the first noun node.
;  2028          2145  2      !
;  2029          2146  2      noun_node = .verb_node [dbg$l_verb_object_ptr];
;  2030          2147  2
;  2031          2148  2      ! Loop through the list of declared names.
;  2032          2149  2      !
;  2033          2150  2      WHILE .noun_node NEQ 0 DO
;  2034          2151  3          BEGIN
;  2035          2152  3
;  2036          2153  3          ! Increment the count of the number of names processed.
;  2037          2154  3          !
;  2038          2155  3          dbg$gl_param_count [.dbg$gl_pr_nest_level] =
;  2039          2156  3                  1 + .dbg$gl_param_count [.dbg$gl_pr_nest_level];
;  2040          2157  3
;  2041          2158  3          ! Obtain the name of the corresponding parameter.
;  2042          2159  3          !
;  2043          2160  3          IF NOT dbg$construct_param_name (
;  2044          2161  3                          .dbg$gl_param_count [.dbg$gl_pr_nest_level],
;  2045          2162  3                          param_name,
;  2046          2163  3                          FALSE,
;  2047          2164  3                          .message_vect)
;  2048          2165  3          THEN
;  2049          2166  3              RETURN sts$k_severe;
;  2050          2167  3
;  2051          2168  3          ! Look up the corresponding value.
;  2052          2169  3          !
;  2053          2170  3          IF NOT dbg$def_sym_find (.param_name, returned_kind, returned_value,
;  2054          2171  3                  global_flag, .message_vect)
;  2055          2172  3          THEN
;  2056          2173  4              BEGIN
;  2057          2174  4
;  2058          2175  4              ! A parameter definition was not found. This most likely
;  2059          2176  4              ! indicates more declarations than there were parameters.
;  2060          2177  4              ! Report an error.
;  2061          2178  4              !
;  2062          2179  4              .message_vect = dbg$nmake_arg_vect (dbg$_declarerr, 1,
;  2063          2180  4                                  .noun_node [dbg$l_noun_value]);
;  2064          2181  4              RETURN sts$k_severe;
;  2065          2182  3              END;
;  2066          2183  3
;  2067          2184  3
;  2068          2185  3          ! Check that the definition that was found was of type parameter.
;  2069          2186  3          ! If not, report an internal DEBUG error.
;  2070          2187  3          !
;  2071          2188  3          IF .returned_kind NEQ define_parameter
;  2072          2189  3          THEN
;  2073          2190  3              $DBG_ERROR('DBGNDECL\NEXECUTE_DECLARE 10');
;  2074          2191  3
;  2075          2192  3
;  2076          2193  3          ! Check that the definition was local and not global.  If it was global,
```

```
;2077     2194  3          ! we report an internal error.
;2078     2195  3          !
;2079     2196  3          IF .global_flag
;2080     2197  3          THEN
;2081     2198  3              $DBG_ERROR('DBGNDECL\NEXECUTE_DECLARE 20');
;2082     2199  3
;2083     2200  3
;2084     2201  3          ! Convert the definition to the appropriate type.
;2085     2202  3          ! First turn the string into an input descriptor.
;2086     2203  3          ! Note - we need to append a carriage return to the end
;2087     2204  3          ! of the string first.
;2088     2205  3          !
;2089     2206  3          len = .returned_value[0];
;2090     2207  3          new_string = dbg$get_tempmem (1+.len/4);
;2091     2208  3          ch$move (.len,returned_value[1],.new_string);
;2092     2209  3          new_string[.len] = dbg$k_car_return;
;2093     2210  3          input_desc [dsc$w_length] = .len+1;
;2094     2211  3          input_desc [dsc$a_pointer] = .new_string;
;2095     2212  3
;2096     2213  3
;2097     2214  3          ! Case on the kind of symbol.
;2098     2215  3          !
;2099     2216  3          CASE .noun_node[dbg$l_noun_value2]
;2100     2217  3              FROM define_lowest TO define_highest OF
;2101     2218  3              SET
;2102     2219  3
;2103     2220  3              [define_address] :
;2104     2221  4                  BEGIN
;2105     2222  4                  LOCAL
;2106     2223  4                      primptr;
;2107     2224  4
;2108     2225  4                  ! Call the address expression interpreter.
;2109     2226  4                  !
;2110     2227  4                  IF NOT DBG$NPARSE_ADDRESS (INPUT_DESC, PRIMPTR,
;2111     2228  4                          .DBG$GB_RADIX[DBG$B_RADIX_INPUT],
;2112     2229  4                          TOKEN$K_TERM_NONE, .MESSAGE_VECT)
;2113     2230  4                  THEN
;2114     2231  4                      RETURN sts$k_severe;
;2115     2232  4
;2116     2233  4                  IF NOT dbg$nget_symid (.primptr, symid_list, .message_vect)
;2117     2234  4                  THEN
;2118     2235  4                      RETURN sts$k_severe;
;2119     2236  4                  IF NOT dbg$ncopy_desc (.primptr, value, .message_vect)
;2120     2237  4                  THEN
;2121     2238  4                      RETURN sts$k_severe;
;2122     2239  4                  dbg$sta_lock_symid (.symid_list);
;2123     2240  3                  END;
;2124     2241  3
;2125     2242  3              [define_command, define_string] :
;2126     2243  4                  BEGIN
;2127     2244  4
;2128     2245  4                  MAP
;2129     2246  4                      value: REF VECTOR[,BYTE];
;2130     2247  4
;2131     2248  4                  ! In these cases we already have the definition in
;2132     2249  4                  ! the appropriate form (a counted string). We need to
;2133     2250  4                  ! allocate space for a copy and then copy the counted
```

```
;  2134          2251  4                ! string.
;  2135          2252  4                !
;  2136          2253  4                value = dbg$get_memory (((.returned_value[0]+1)+3)/4);
;  2137          2254  4                value[0] = .returned_value[0];
;  2138          2255  4                ch$move (.returned_value[0], returned_value[1], value[1]);
;  2139          2256  3                END;
;  2140          2257  3
;  2141          2258  3            [define_procedure] :
;  2142          2259  4                BEGIN
;  2143          2260  4
;  2144          2261  4                ! For this case we want to pick up a sequence of DEBUG
;  2145          2262  4                ! commands inside of parenthesis.
;  2146          2263  4                ! First we eat the left paren.
;  2147          2264  4                !
;  2148          2265  4                IF NOT dbg$nmatch (input_desc, dbg$cs_left_paren, 1)
;  2149          2266  4                THEN
;  2150          2267  4                    report_error;
;  2151          2268  4
;  2152          2269  4                ! Call the routine which picks up a sequence of
;  2153          2270  4                ! DEBUG commands.
;  2154          2271  4                !
;  2155          2272  4                IF NOT dbg$nsave_break_buffer (input_desc,
;  2156          2273  4                        value,
;  2157          2274  4                        .message_vect)
;  2158          2275  4                THEN
;  2159          2276  4                    RETURN sts$k_severe;
;  2160          2277  4
;  2161          2278  3                END;
;  2162          2279  3
;  2163          2280  3            [define_value] :
;  2164          2281  4                BEGIN
;  2165          2282  4                LOCAL
;  2166          2283  4                    temp_desc;  ! Variable to hold a pointer to a value descriptor
;  2167          2284  4
;  2168          2285  4                ! For this case we just call the expression interpreter
;  2169          2286  4                ! to parse a language expression.
;  2170          2287  4                !
;  2171          2288  4                IF NOT DBG$NPARSE_EXPRESSION (INPUT_DESC,
;  2172          2289  4                        .DBG$GB_RADIX[DBG$B_RADIX_INPUT],
;  2173          2290  4                        TEMP_DESC, TOKEN$K_TERM_NONE, .MESSAGE_VECT)
;  2174          2291  4                THEN
;  2175          2292  4                    RETURN sts$k_severe;
;  2176          2293  4
;  2177          2294  4                ! Copy the descriptor into permanent memory.
;  2178          2295  4                !
;  2179          2296  4                IF NOT dbg$nget_symid (.temp_desc, symid_list, .message_vect)
;  2180          2297  4                THEN
;  2181          2298  4                    RETURN sts$k_severe;
;  2182          2299  4                IF NOT dbg$ncopy_desc (.temp_desc,
;  2183          2300  4                    value, .message_vect)
;  2184          2301  4                THEN
;  2185          2302  4                    RETURN sts$k_severe;
;  2186          2303  4                dbg$sta_lock_symid (.symid_list);
;  2187          2304  4
;  2188          2305  3                END;
;  2189          2306  3
;  2190          2307  3
```

```
;  2191          2308  3            ! Anything else is an error.
;  2192          2309  3            !
;  2193          2310  3            [INRANGE,OUTRANGE] :
;  2194          2311  3                $DBG_ERROR('DBGNDECL\NEXECUTE_DECLARE 30');
;  2195          2312  3
;  2196          2313  3            TES;
;  2197          2314  3
;  2198          2315  3
;  2199          2316  3        ! Add the appropriate symbol definition.
;  2200          2317  3        !
;  2201          2318  3        IF NOT dbg$def_sym_add (.noun_node [dbg$l_noun_value],
;  2202          2319  3                .noun_node [dbg$l_noun_value2], .value,
;  2203          2320  3                FALSE, replaced_flag, .message_vect)
;  2204          2321  3        THEN
;  2205          2322  3            RETURN sts$k_severe;
;  2206          2323  3
;  2207          2324  3        ! Set up for the next time around the loop.
;  2208          2325  3        !
;  2209          2326  3        noun_node = .noun_node [dbg$l_noun_link];
;  2210          2327  2        END; ! while loop
;  2211          2328  2
;  2212          2329  2    ! Return success.
;  2213          2330  2    !
;  2214          2331  2    RETURN sts$k_success;
;  2215          2332  2
;  2216          2333  1    END;        ! End of dbg$nexecute_decl. .

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

                                                 0D  01  0013C P.ABD:   .BYTE   1, 13                                          ;
                                                 28  01  0013E P.ABE:   .BYTE   1, 40                                          ;
43 45 58 45 4E 5C 4C 43 45 44 4E 47 42  44  1C  00140 P.ABF:   .ASCII  <28>\DBGNDECL\<92>\NEXECUTE_DECLARE 10\                 ;
   30 31 20 45 52 41 4C 43 45 44 5F 45  54  55  0014F
43 45 58 45 4E 5C 4C 43 45 44 4E 47 42  44  1C  0015D P.ABG:   .ASCII  <28>\DBGNDECL\<92>\NEXECUTE_DECLARE 20\                 ;
   30 32 20 45 52 41 4C 43 45 44 5F 45  54  55  0016C
43 45 58 45 4E 5C 4C 43 45 44 4E 47 42  44  1C  0017A P.ABH:   .ASCII  <28>\DBGNDECL\<92>\NEXECUTE_DECLARE 30\                 ;
   30 33 20 45 52 41 4C 43 45 44 5F 45  54  55  00189

                                                        DBG$CS_CR=          P.ABD
                                                        DBG$CS_LEFT_PAREN=  P.ABE

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

                                          0FFC 00000            .ENTRY  DBG$NEXECUTE_DECLARE, Save R2,R3,R4,R5,R6,-          ; 2080
                                                                        R7,R8,R9,R10,R11                                        ;
                                5B 00000000'  EF  9E 00002      MOVAB   P.ABF, R11                                            ;
                                5E            30  C2 00009      SUBL2   #48, SP                                               ;
                                50      04    AC  D0 0000C      MOVL    VERB_NODE, R0                                         ; 2146
                                58      08    A0  D0 00010      MOVL    8(R0), NOUN_NODE                                      ;
                                59      08    AC  D0 00014      MOVL    MESSAGE_VECT, R9                                      ; 2164
                                              58  D5 00018 1$:   TSTL    NOUN_NODE                                             ; 2150
                                              03  12 0001A      BNEQ    2$                                                    ;
                                            01FA  31 0001C      BRW     24$                                                   ;
                                50 00000000'  EF  D0 0001F 2$:  MOVL    DBG$GL_PR_NEST_LEVEL, R0                              ; 2155
```

```
                00000000'EF40   D6 00026          INCL    DBG$GL_PARAM_COUNT[R0]                  ; 2156
                          59    DD 00029          PUSHL   R9                                      ; 2164
                          7E    D4 0002B          CLRL    -(SP)                                   ; 2160
                          08    AE 9F 0002D       PUSHAB  PARAM_NAME
                00000000'EF40   DD 00030          PUSHL   DBG$GL_PARAM_COUNT[R0]                  ; 2161
          FAC0  CF              04 FB 00037       CALLS   #4, DBG$CONSTRUCT_PARAM_NAME
                2C              50 E9 0003C       BLBC    R0, 4$
                                59 DD 0003F       PUSHL   R9                                      ; 2171
                          08    AE 9F 00041       PUSHAB  GLOBAL_FLAG                             ; 2170
                          10    AE 9F 00044       PUSHAB  RETURNED_VALUE
                          18    AF 9F 00047       PUSHAB  RETURNED_KIND
                          10    AE DD 0004A       PUSHL   PARAM_NAME
00000000G       00              05 FB 0004D       CALLS   #5, DBG$DEF_SYM_FIND
                17              50 E8 00054       BLBS    R0, 5$
                                68 DD 00057       PUSHL   (NOUN_NODE)                             ; 2180
                                01 DD 00059       PUSHL   #1                                      ; 2179
                00028E60        8F DD 0005B       PUSHL   #167520
00000000G       00              03 FB 00061       CALLS   #3, DBG$NMAKE_ARG_VECT
                69              50 D0 00064 3$:   MOVL    R0, (R9)
                          019C  31 00067 4$:      BRW     22$                                     ; 2181
                06        0C    AE D1 0006A 5$:   CMPL    RETURNED_KIND, #6                       ; 2188
                                11 13 0006E       BEQL    6$
                                5B DD 00070       PUSHL   R11                                     ; 2190
                                01 DD 00072       PUSHL   #1
                00028362        8F DD 00074       PUSHL   #164706
00000000G       00              03 FB 0007A       CALLS   #3, LIB$SIGNAL
                12        04    AE E9 00081 6$:   BLBC    GLOBAL_FLAG, 7$                         ; 2196
                          1D    AB 9F 00085       PUSHAB  P.ABG                                   ; 2198
                                01 DD 00088       PUSHL   #1
                00028362        8F DD 0008A       PUSHL   #164706
00000000G       00              03 FB 00090       CALLS   #3, LIB$SIGNAL
                          57    08 AE D0 00097 7$:   MOVL    RETURNED_VALUE, R7                     ; 2206
                          56        67 9A 0009B   MOVZBL  (R7), LEN
                50        56        04 C7 0009E   DIVL3   #4, LEN, R0                            ; 2207
                          01    A0 9F 000A2       PUSHAB  1(R0)
00000000G       00              01 FB 000A5       CALLS   #1, DBG$GET_TEMPMEM
                          5A        50 D0 000AC   MOVL    R0, NEW_STRING
                6A        01    A7    56 28 000AF MOVC3   LEN, 1(R7), (NEW_STRING)                ; 2208
                        664A        0D 90 000B4   MOVB    #13, (LEN)[NEW_STRING]                  ; 2209
          24    AE        56        01 A1 000B8   ADDW3   #1, LEN, INPUT_DESC                     ; 2210
                    28    AE        5A D0 000BD   MOVL    NEW_STRING, INPUT_DESC+4                ; 2211
                06              01    0C A8 CF 000C1   CASEL   12(NOUN_NODE), #1, #6              ; 2216
    005D        0081            005D      0022    000C6 8$:   .WORD  10$-8$,-
                000E            000E      00DE    000CE       13$-8$,-
                                                              15$-8$,-
                                                              13$-8$,-
                                                              19$-8$,-
                                                              9$-8$,-
                                                              9$-8$
                          3A    AB 9F 000D4 9$:   PUSHAB  P.ABH                                   ; 2311
                                01 DD 000D7       PUSHL   #1
                00028362        8F DD 000D9       PUSHL   #164706
00000000G       00              03 FB 000DF       CALLS   #3, LIB$SIGNAL
                                5C 11 000E6       BRB     14$
                                59 DD 000E8 10$:  PUSHL   R9                                      ; 2229
                                7E D4 000EA       CLRL    -(SP)                                   ; 2227
                          7E 00000000G  00  9A 000EC      MOVZBL  DBG$GB_RADIX, -(SP)             ; 2228
```

```
                                1C   AE   9F 000FB          PUSHAB  PRIMPTR                                 ; 2227
                                34   AE   9F 000FE          PUSHAB  INPUT_DESC
                         00000000G   00    05 FB 00101      CALLS   #5, DBG$NPARSE_ADDRESS
                                     0F    50 E9 00108      BLBC    R0, 11$
                                           59 DD 0010B      PUSHL   R9                                      ; 2233
                                1C   AE   9F 0010D          PUSHAB  SYMID_LIST
                                18   AE   DD 00110          PUSHL   PRIMPTR
                         00000000G   00    03 FB 00113      CALLS   #3, DBG$NGET_SYMID
                                     03    50 E8 0011A 11$: BLBS    R0, 12$
                                        00EE 31 0011D       BRW     22$
                                           59 DD 00120 12$: PUSHL   R9                                      ; 2236
                                20   AE   9F 00122          PUSHAB  VALUE
                                18   AE   DD 00125          PUSHL   PRIMPTR
                                        00B6 31 00128       BRW     20$
                                     50    67 9A 0012B 13$: MOVZBL  (R7), R0                                ; 2253
                                     50    04 C0 0012E      ADDL2   #4, R0
                                7E   50    04 C7 00131      DIVL3   #4, R0, -(SP)
                         00000000G   00    01 FB 00135      CALLS   #1, DBG$GET_MEMORY
                                1C   AE    50 D0 0013C      MOVL    R0, VALUE
                                     60    67 90 00140      MOVB    (R7), (R0)                              ; 2254
                                     51    67 9A 00143      MOVZBL  (R7), R1                                ; 2255
                             01  A0   01    A7 51 28 00146  MOVC3   R1, 1(R7), 1(R0)
                                        00A6 31 0014C 14$:   BRW     21$                                     ; 2216
                                           01 DD 0014F 15$:  PUSHL   #1                                      ; 2265
                                FE   AB   9F 00151          PUSHAB  DBG$CS_LEFT_PAREN
                                2C   AE   9F 00154          PUSHAB  INPUT_DESC
                         00000000G   00    03 FB 00157      CALLS   #3, DBG$NMATCH
                                     37    50 E8 0015E      BLBS    R0, 18$
                                           01 DD 00161      PUSHL   #1                                      ; 2266
                                   FC   AB   9F 00163       PUSHAB  DBG$CS_CR
                                   2C   AE   DD 00166       PUSHL   INPUT_DESC
                         00000000G   00    03 FB 00169      CALLS   #3, DBG$NMATCH
                                     0F    50 E9 00170      BLBC    R0, 16$
                                 000280D0  8F DD 00173      PUSHL   #164048
                         00000000G   00    01 FB 00179      CALLS   #1, DBG$NMAKE_ARG_VECT
                                           13 11 00180      BRB     17$
                                   24   AE   DD 00182 16$:  PUSHL   INPUT_DESC
                         00000000G   00    01 FB 00185      CALLS   #1, DBG$NNEXT_WORD
                                           50 DD 0018C      PUSHL   R0
                         00000000G   00    01 FB 0018E      CALLS   #1, DBG$NSYNTAX_ERROR
                                        FED4 31 00195 17$:  BRW     3$
                                           59 DD 00198 18$: PUSHL   R9                                      ; 2274
                                20   AE   9F 0019A          PUSHAB  VALUE                                   ; 2272
                                2C   AE   9F 0019D          PUSHAB  INPUT_DESC
                         00000000G   00    03 FB 001A0      CALLS   #3, DBG$NSAVE_BREAK_BUFFER
                                     4B    50 E8 001A7      BLBS    R0, 21$
                                           62 11 001AA      BRB     22$                                     ; 2276
                                           59 DD 001AC 19$: PUSHL   R9                                      ; 2290
                                           7E D4 001AE      CLRL    -(SP)                                   ; 2288
                                   1C   AE   9F 001B0       PUSHAB  TEMP_DESC
                                7E 00000000G 00 9A 001B3    MOVZBL  DBG$GB_RADIX, -(SP)                     ; 2289
                                   34   AE   9F 001BA       PUSHAB  INPUT_DESC                              ; 2288
                         00000000G   00    05 FB 001BD      CALLS   #5, DBG$NPARSE_EXPRESSION
                                     47    50 E9 001C4      BLBC    R0, 22$
                                           59 DD 001C7      PUSHL   R9                                      ; 2296
                                   1C   AE   9F 001C9       PUSHAB  SYMID_LIST
                                   1C   AE   DD 001CC       PUSHL   TEMP_DESC
```

```
                00000000G  00        03  FB 001CF          CALLS   #3, DBG$NGET_SYMID                       ;
                           35        50  E9 001D6          BLBC    R0, 22$                                 ;
                                     59  DD 001D9          PUSHL   R9                                      ; 2300
                                 20  AE  9F 001DB          PUSHAB  VALUE                                   ; 2299
                                 1C  AE  DD 001DE          PUSHL   TEMP_DESC                               ;
                00000000G  00        03  FB 001E1 20$:     CALLS   #3, DBG$NCOPY_DESC                      ;
                           23        50  E9 001E8          BLBC    R0, 22$                                 ;
                                 18  AE  DD 001EB          PUSHL   SYMID_LIST                              ; 2303
                00000000G  00        01  FB 001EE          CALLS   #1, DBG$STA_LOCK_SYMID                  ;
                                     59  DD 001F5 21$:     PUSHL   R9                                      ; 2320
                                 24  AE  9F 001F7          PUSHAB  REPLACED_FLAG                           ; 2318
                                     7E  D4 001FA          CLRL    -(SP)                                   ;
                                 28  AE  DD 001FC          PUSHL   VALUE                                   ; 2319
                                 0C  A8  DD 001FF          PUSHL   12(NOUN_NODE)                           ;
                                     68  DD 00202          PUSHL   (NOUN_NODE)                             ; 2318
                00000000G  00        06  FB 00204          CALLS   #6, DBG$DEF_SYM_ADD                     ;
                           04        50  E8 0020B          BLBS    R0, 23$                                 ;
                           50        04  D0 0020E 22$:     MOVL    #4, R0                                  ; 2322
                                     04  00211             RET                                             ;
                           58  08    A8  D0 00212 23$:     MOVL    8(NOUN_NODE), NOUN_NODE                 ; 2326
                               FDFF  31  00216             BRW     1$                                      ; 2150
                           50        01  D0 00219 24$:     MOVL    #1, R0                                  ; 2331
                                     04  0021C             RET                                             ; 2333

; Routine Size:  541 bytes,    Routine Base:  DBG$CODE + 0A3E

; 2217          2334  1 END
; 2218          2335  0 ELUDOM


                                                                .EXTRN  LIB$SIGNAL

                                PSECT SUMMARY

     Name                       Bytes                          Attributes

DBG$GLOBAL                        48  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
DBG$PLIT                         407  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
DBG$CODE                        3163  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
.  ABS  .                          0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)


                              Library Statistics

                                        -------- Symbols --------      Pages       Processing
     File                               Total    Loaded   Percent      Mapped      Time

_$255$DUA28:[SYSLIB]LIB.L32;1           18619        78         0        1000        00:01.8
_$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1      32         1         3           7        00:00.1
_$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1      1545       122         7          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       13        3        22      00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1        150        1        0        12      00:00.3



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


;                                       COMMAND QUALIFIERS

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

; Size:          3163 code + 455 data bytes
; Run Time:        00:58.6
; Elapsed Time:    03:15.9
; Lines/CPU Min:    2391
; Lexemes/CPU-Min: 14814
; Memory Used:  260 pages
; Compilation Complete
```

STRUCDEF
REQ

TEMPREQ
REQ

DBGADDEXP
LIS

DBGATSIGN
LIS

SSIDEF
REQ

DBGCALL
LIS

DBGCVTDX
LIS