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

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN    CCCCCCCC  NN      NN  TTTTTTTTTT  RRRRRRRR    LL
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN    CCCCCCCC  NN      NN  TTTTTTTTTT  RRRRRRRR    LL
DD      DD  BB      BB  GG          NN      NN  CC          NN      NN      TT      RR      RR  LL
DD      DD  BB      BB  GG          NN      NN  CC          NN      NN      TT      RR      RR  LL
DD      DD  BB      BB  GG          NNNN    NN  CC          NNNN    NN      TT      RR      RR  LL
DD      DD  BB      BB  GG          NNNN    NN  CC          NNNN    NN      TT      RR      RR  LL
DD      DD  BBBBBBBB    GG          NN  NN  NN  CC          NN  NN  NN      TT      RRRRRRRR    LL
DD      DD  BBBBBBBB    GG          NN  NN  NN  CC          NN  NN  NN      TT      RRRRRRRR    LL
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  CC          NN    NNNN      TT      RR  RR      LL
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  CC          NN    NNNN      TT      RR  RR      LL
DD      DD  BB      BB  GG      GG  NN      NN  CC          NN      NN      TT      RR    RR    LL
DD      DD  BB      BB  GG      GG  NN      NN  CC          NN      NN      TT      RR    RR    LL
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN    CCCCCCCC  NN      NN      TT      RR      RR  LLLLLLLLLL
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN    CCCCCCCC  NN      NN      TT      RR      RR  LLLLLLLLLL
```

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

```
;     1          0001  0 MODULE DBGNCNTRL (IDENT = 'V04-000') =
;     2          0002  1 BEGIN
;     3          0003  1
;     4          0004  1 !
;     5          0005  1 !**************************************************************************
;     6          0006  1 !*                                                                        *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                  *
;    10          0010  1 !*                                                                        *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                           *
;    17          0017  1 !*                                                                        *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                           *
;    21          0021  1 !*                                                                        *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !**************************************************************************
;    27          0027  1 !
;    28          0028  1 !
;    29          0029  1 ! MODULE FUNCTION
;    30          0030  1 !       This module contains DEBUG's top level parsing and execution routines.
;    31          0031  1 !       Routine DBG$NCONTROL is called after an input line has been collected
;    32          0032  1 !       from the user's terminal, or a record has been read from an indirect
;    33          0033  1 !       command file or DO action buffer. Other routines in this module break
;    34          0034  1 !       an input line into single commands, verify these commands if appropri-
;    35          0035  1 !       ate, and allocate and deallocate temporary DEBUG memory (i.e., memory
;    36          0036  1 !       that is automatically reclaimed at the end of each command).  End of
;    37          0037  1 !       command clean-up routines are also included.
;    38          0038  1 !
;    39          0039  1 !       Routine DBG$NCONTROL invokes the top level parsing and command execu-
;    40          0040  1 !       tion networks as needed.  Commands are parsed and executed one at a
;    41          0041  1 !       time.  Detection of errors in the parsing phase cause the command in
;    42          0042  1 !       question to not be executed.
;    43          0043  1 !
;    44          0044  1 !
;    45          0045  1 ! AUTHOR:       David Plummer, CREATION DATE:   4/15/80
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !       R. Title, Feb 1982,     Filled in CIS$A_WHILE_CLAUSE field from
;    50          0050  1 !                               within DBG$NGET_CMD; this was in order to
;    51          0051  1 !                               implement the WHILE command.
;    52          0052  1 !       R. Title, Apr 1982,     Fixed a bug in DBG$NSAVE_FILESP that was
;    53          0053  1 !                               preventing logical name translation from
;    54          0054  1 !                               occuring (a semicolon was always being
;    55          0055  1 !                               appended to the filename).
;    56          0056  1 !       R. Title, Jan 1983      Changed DBG$NGET_CMD so that when language
;    57          0057  1 !                               is set to C, the comment character is
```

```
;    58          0058  1 !                                    /*  and not  !
;    59          0059  1 !       R. Title, Sep 1983           Fixed DBG$NGET_CMD to treat a line of tabs
;    60          0060  1 !                                    the same as a line of blanks.
;    61          0061  1 !       B. Becker, Oct 1983          Fixed up the uppercasing so that when an Ada
;    62          0062  1 !                                    tick is found DBG$NGET_CMD will continue to
;    63          0063  1 !                                    uppercase the line.  Also created another
;    64          0064  1 !                                    routine to do the uppercasing work.
;    65          0065  1 !
;    66          0066  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    67          0200  1
;    68          0201  1 LIBRARY 'LIB$:DBGGEN.L32';
;    69          0202  1
;    70          0203  1 FORWARD ROUTINE
;    71          0204  1     DBG$NCONTROL: NOVALUE,              ! Controls parsing and command execution
;    72          0205  1     ADD_TO_BUFLIST: NOVALUE,
;    73          0206  1     DBG$NGET_CMD,                       ! Chops input string into command strings
;    74          0207  1     DBG$NKILL_CMD,                      ! Kills a command and frees up dynamic
;    75          0208  1                                         !     memory
;    76          0209  1     DBG$NEND_OF_INPUT,                  ! End of parse line clean-up
;    77          0210  1     DBG$NSAVE_FILESP,                   ! Saves a filespec in a dynamic buffer
;    78          0211  1     DBG$NVERIFY_OUT: NOVALUE,           ! VERIFIES indirect command file commands
;    79          0212  1     DBG$NCHANGE_TO_NEW: NOVALUE,        ! Aids transition to new debugger
;    80          0213  1     DBG$NSAVE_BREAK_BUFFER: NOVALUE,    ! Save a break action buffer
;    81          0214  1     GET_C_CMD_STRING,                   ! Uppercases a 'C' command string
;    82          0215  1     GET_ADA_CMD_STRING,                 ! Uppercases a Ada command string
;    83          0216  1     GET_NORMAL_CMD_STRING;              ! Uppercases a normal command string
```

```
;    85          0217  1 EXTERNAL ROUTINE
;    86          0218  1     DBG$EXPAND_DEFINE_NAME,          ! Expands a DEFINE name
;    87          0219  1     DBG$FAO_OUT: NOVALUE,            ! ???
;    88          0220  1     DBG$NINITIALIZE: NOVALUE,        ! Sets language specific context
;    89          0221  1     DBG$NMATCH,                      ! Matches counted string to input
;    90          0222  1     DBG$NOUT_INFO,                   ! Outputs an informational message
;    91          0223  1     DBG$NSYNTAX_ERROR,               ! Formats a syntax error
;    92          0224  1     DBG$END_OF_LINE: NOVALUE,        ! Version 2 end of line clean-up
;    93          0225  1     DBG$END_OF_CMD: NOVALUE,         ! Version 2 end of command clean-up
;    94          0226  1     DBG$NOUT_ARG_VECT: NOVALUE,      ! Outputs a message vector
;    95          0227  1     DBG$NMAKE_ARG_VECT,              ! Constructs an argument vector
;    96          0228  1     DBG$NNEXT_WORD,                  ! Isolates next word of input
;    97          0229  1     DBG$NCIS_REMOVE,                 ! Removes a link from the cis
;    98          0230  1     DBG$GET_MEMORY,                  ! Allocates a dynamic memory block
;    99          0231  1     DBG$GET_TEMPMEM,                 ! Allocates a temporary memory block
;   100          0232  1     DBG$REL_MEMORY,                  ! Release permanent memory
;   101          0233  1     DBG$NPARSE_CMD,                  ! The DEBUG command parser
;   102          0234  1     DBG$NEXECUTE_CMD;                ! The DEBUG command executor
;   103          0235  1
;   104          0236  1 EXTERNAL
;   105          0237  1     DBG$GB_LANGUAGE: BYTE,           ! Current language setting
;   106          0238  1     DBG$GL_GBLTYP,                   ! Override type
;   107          0239  1     DBG$GW_GBLLNGTH: WORD,           ! Override length
;   108          0240  1     DBG$GL_DFLTTYP,                  ! Default type
;   109          0241  1     DBG$GW_DFLTLENG: WORD,           ! Default length
;   110          0242  1     DBG$GL_CISHEAD: REF CIS$LINK,    ! Head of cis
;   111          0243  1     DBG$GB_DEF_OUT: VECTOR[,BYTE],   ! Output control vecter in old debugger
;   112          0244  1     DBG$GL_ORIG_COMMAND_PTR,         ! Pointer to original command string
;   113          0245  1     DBG$GL_UPCASE_COMMAND_PTR: VECTOR[2];
;   114          0246  1                                      ! Pointers to start and end
;   115          0247  1                                      !    of current command string
;   116          0248  1
;   117          0249  1 GLOBAL
;   118          0250  1     DBG$GL_ORIG_COMMAND_PTR,         ! Pointer to original command string
;   119          0251  1     DBG$GL_UPCASE_COMMAND_PTR:       ! Pointer to upcased command string
;   120          0252  1                         VECTOR[2, LONG];!
;   121          0253  1
;   122          0254  1 OWN
;   123          0255  1     MESSAGE_POINTER,                 ! Holds address of message argument vector
;   124          0256  1     CMD_STG_DESC: BLOCK[12,BYTE],    ! Command input string descriptor. Note
;   125          0257  1                                      !      the extra longword to contain
;   126          0258  1                                      !      the original DSC$A_POINTER.
;   127          0259  1     CMD_VERB_PTR,                    ! Start of executable parse tree
;   128          0260  1     SAVE_INPUT_DESC: REF DBG$STG_DESC,  ! Pointer to parse string descriptor
;   129          0261  1                                      !      used in gathering filespecs.
;   130          0262  1     START_VERIFY_POINTER;            ! Pointer to the start of the input to
;   131          0263  1                                      !      be verified
;   132          0264  1
;   133          0265  1 MACRO
;   134          0266  1         INITIAL_PTR = 8, 0, 32, 0 %;    ! Pointer to start of dynamic buffer
```

```
;  136     0267  1 GLOBAL ROUTINE DBG$NCONTROL(PARSE_STG_DESC): NOVALUE =
;  137     0268  1 !
;  138     0269  1 ! FUNCTION
;  139     0270  1 !       Routine DBG$NCONTROL oversees command parsing and execution. Only commands
;  140     0271  1 !       that are parsed without detection of errors are executed. Routines are invoked
;  141     0272  1 !       for end of command and input processing.
;  142     0273  1 !
;  143     0274  1 ! FORMAL PARAMETERS:
;  144     0275  1 !       PARSE_STG_DESC - A VAX standrd descriptor of the input string.
;  145     0276  1 !
;  146     0277  1 ! IMPLICIT INPUTS:
;  147     0278  1 !
;  148     0279  1 !       CMD_VERB_PTR - Pointer to the verb node (head node) of the
;  149     0280  1 !               executable command tree.
;  150     0281  1 !
;  151     0282  1 !       MESSAGE_POINTER - Pointer to message argument vector.
;  152     0283  1 !
;  153     0284  1 !
;  154     0285  1
;  155     0286  2     BEGIN
;  156     0287  2
;  157     0288  2     MAP
;  158     0289  2         PARSE_STG_DESC : REF BLOCK [,BYTE];
;  159     0290  2
;  160     0291  2     LOCAL
;  161     0292  2         STATUS;                     ! Retains return code
;  162     0293  2
;  163     0294  2
;  164     0295  2     ! Try to get another command from the present input buffer. Check for comments.
;  165     0296  2     !
;  166     0297  2     status = dbg$nget_cmd (.parse_stg_desc, cmd_stg_desc, message_pointer);
;  167     0298  2
;  168     0299  2     CASE .status FROM sts$k_warning TO sts$k_severe
;  169     0300  2         OF
;  170     0301  2         SET
;  171     0302  2
;  172     0303  2         [sts$k_warning] :               ! No more input from present buffer
;  173     0304  3             BEGIN
;  174     0305  3             IF NOT dbg$nend_of_input (message_pointer)
;  175     0306  3             THEN
;  176     0307  3                 dbg$nout_arg_vect (.message_pointer);
;  177     0308  2             END;
;  178     0309  2
;  179     0310  2         [sts$k_success] :       ! Parse and execute command
;  180     0311  3             BEGIN
;  181     0312  3
;  182     0313  3             dbg$ninitialize ();
;  183     0314  3
;  184     0315  3             IF dbg$nparse_cmd (cmd_stg_desc, cmd_verb_ptr, message_pointer)
;  185     0316  3             THEN
;  186     0317  4                 BEGIN
;  187     0318  4                 dbg$nverify_out (.parse_stg_desc [dsc$a_pointer]);
;  188     0319  4
;  189     0320  4                 IF NOT dbg$nexecute_cmd (cmd_verb_ptr, message_pointer)
;  190     0321  4                 THEN
;  191     0322  5                     BEGIN
;  192     0323  5                     dbg$nout_arg_vect (.message_pointer);
```

```
;  193          0324  5                IF NOT dbg$nkill_cmd (message_pointer)
;  194          0325  5                THEN
;  195          0326  5                    dbg$nout_arg_vect (.message_pointer);
;  196          0327  4                END;
;  197          0328  4            END
;  198          0329  3        ELSE
;  199          0330  4            BEGIN   ! Kill command - bad parse
;  200          0331  4            dbg$nout_arg_vect (.message_pointer);
;  201          0332  4            IF NOT dbg$nkill_cmd (message_pointer)
;  202          0333  4            THEN
;  203          0334  4                dbg$nout_arg_vect (.message_pointer);
;  204          0335  3            END;
;  205          0336  2        END;
;  206          0337  2
;  207          0338  2    [sts$k_error] : ! Not parsable. Just verify the comment.
;  208          0339  3        BEGIN
;  209          0340  3        dbg$nverify_out (.parse_stg_desc [dsc$a_pointer]);
;  210          0341  3        IF NOT dbg$nend_of_input (message_pointer)
;  211          0342  3        THEN
;  212          0343  3            dbg$nout_arg_vect (.message_pointer);
;  213          0344  2        END;
;  214          0345  2
;  215          0346  2    [sts$k_severe] :        ! Error in input
;  216          0347  3        BEGIN
;  217          0348  3        dbg$nout_arg_vect (.message_pointer);
;  218          0349  3        IF NOT dbg$nkill_cmd (message_pointer)
;  219          0350  3        THEN
;  220          0351  3            dbg$nout_arg_vect (.message_pointer);
;  221          0352  2        END;
;  222          0353  2
;  223          0354  2    [INRANGE,OUTRANGE] :
;  224          0355  3        BEGIN
;  225          0356  3        0;
;  226          0357  2        END;
;  227          0358  2
;  228          0359  2        TES;
;  229          0360  2
;  230          0361  2
;  231          0362  2    ! Perform end of command clean-up.  This involves resetting data structures
;  232          0363  2    ! shared between the old debugger and the new.  It also involves releasing
;  233          0364  2    ! all temporary memory allocated during the processing of the command and
;  234          0365  2    ! releasing all unreferenced RST entries on the Temporary RST Entry List.
;  235          0366  2    !
;  236          0367  2    DBG$END_OF_CMD();
;  237          0368  2    RETURN;
;  238          0369  2
;  239          0370  1    END;


                                .TITLE  DBGNCNTRL
                                .IDENT  \V04-000\

                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                            00000 MESSAGE_POINTER:
                                .BLKB   4
                            00004 CMD_STG_DESC:
```

```
                        .BLKB   12
00010 CMD_VERB_PTR:
                        .BLKB   4
00014 SAVE_INPUT_DESC:
                        .BLKB   4
00018 START_VERIFY_POINTER:
                        .BLKB   4

                        .PSECT  DBG$GLOBAL,NOEXE,  PIC,2

00000 DBG$GL_ORIG_COMMAND_PTR::
                        .BLKB   4
00004 DBG$GL_UPCASE_COMMAND_PTR::
                        .BLKB   8

                        .EXTRN  DBG$EXPAND_DEFINE_NAME
                        .EXTRN  DBG$FAO_OUT, DBG$NINITIALIZE
                        .EXTRN  DBG$NMATCH, DBG$NOUT_INFO
                        .EXTRN  DBG$NSYNTAX_ERROR
                        .EXTRN  DBG$END_OF_LINE
                        .EXTRN  DBG$END_OF_CMD, DBG$NOUT_ARG_VECT
                        .EXTRN  DBG$NMAKE_ARG_VECT
                        .EXTRN  DBG$NNEXT_WORD, DBG$NCIS_REMOVE
                        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
                        .EXTRN  DBG$REL_MEMORY, DBG$NPARSE_CMD
                        .EXTRN  DBG$NEXECUTE_CMD
                        .EXTRN  DBG$GB_LANGUAGE
                        .EXTRN  DBG$GL_GBLTYP, DBG$GW_GBLLNGTH
                        .EXTRN  DBG$GL_DFLTTYP, DBG$GW_DFLTLENG
                        .EXTRN  DBG$GL_CISHEAD, DBG$GB_DEF_OUT

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

                    001C 00000         .ENTRY  DBG$NCONTROL, Save R2,R3,R4     ; 0267
            54 00000000G   00    9E 00002 MOVAB   DBG$NOUT_ARG_VECT, R4       ;
            53 00000000'   EF    9E 00009 MOVAB   MESSAGE_POINTER, R3         ;
                           53    DD 00010 PUSHL   R3                          ; 0297
                     04    A3    9F 00012 PUSHAB  CMD_STG_DESC                ;
                52   04    AC    D0 00015 MOVL    PARSE_STG_DESC, R2          ;
                           52    DD 00019 PUSHL   R2                          ;
              0000V CF           03    FB 0001B CALLS   #3, DBG$NGET_CMD      ;
          04        00           50    CF 00020 CASEL   STATUS, #0, #4        ; 0299
0063          003E        000C          0046     00024 1$:    .WORD   4$-1$,-  ;
                                        004F     0002C                 2$-1$,-  ;
                                                               3$-1$,-  ;
                                                               7$-1$,-  ;
                                                               5$-1$    ;
                           57    11 0002E         BRB     7$                  ;
         00000000G   00          00    FB 00030 2$:    CALLS   #0, DBG$NINITIALIZE ; 0313
                           53    DD 00037         PUSHL   R3                  ; 0315
                     10    A3    9F 00039         PUSHAB  CMD_VERB_PTR        ;
                     04    A3    9F 0003C         PUSHAB  CMD_STG_DESC        ;
         00000000G   00          03    FB 0003F CALLS   #3, DBG$NPARSE_CMD    ;
                     2A          50    E9 00046 BLBC    R0, 5$                ;
                     04    A2    DD 00049         PUSHL   4(R2)               ; 0318
              0000V CF           01    FB 0004C CALLS   #1, DBG$NVERIFY_OUT    ;
                           53    DD 00051         PUSHL   R3                  ; 0320
```

```
                                10   A3  9F 00053          PUSHAB  CMD_VERB_PTR                          ;
                    00000000G   00       02  FB 00056          CALLS   #2, DBG$NEXECUTE_CMD             ;
                                27       50  E8 0005D          BLBS    R0, 7$                           ;
                                         11  11 00060          BRB     5$                               ; 0331
                                04       A2  DD 00062 3$:      PUSHL   4(R2)                            ; 0340
                          0000V CF           01  FB 00065          CALLS   #1, DBG$NVERIFY_OUT              ;
                                         53  DD 0006A 4$:      PUSHL   R3                               ; 0341
                          0000V CF           01  FB 0006C          CALLS   #1, DBG$NEND_OF_INPUT            ;
                                         0C  11 00071          BRB     6$                               ;
                                         63  DD 00073 5$:      PUSHL   MESSAGE_POINTER                  ; 0348
                                64           01  FB 00075          CALLS   #1, DBG$NOUT_ARG_VECT            ;
                                         53  DD 00078          PUSHL   R3                               ; 0349
                          0000V CF           01  FB 0007A          CALLS   #1, DBG$NKILL_CMD                ;
                                05       50  E8 0007F 6$:      BLBS    R0, 7$                           ;
                                         63  DD 00082          PUSHL   MESSAGE_POINTER                  ; 0351
                                64           01  FB 00084          CALLS   #1, DBG$NOUT_ARG_VECT            ;
                    00000000G   00       00  FB 00087 7$:      CALLS   #0, DBG$END_OF_CMD               ; 0367
                                         04  0008E            RET                                      ; 0370

; Routine Size:  143 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  241          0371  1 ROUTINE ADD_TO_BUFLIST (BUFFER) : NOVALUE =
;  242          0372  1 !
;  243          0373  1 ! FUNCTION
;  244          0374  1 !       This routine builds a list of buffers to be freed at the end of
;  245          0375  1 !       processing a line of input.
;  246          0376  1 !
;  247          0377  1 !       The top link in the CIS list represents the current line of input.
;  248          0378  1 !       There is a field CIS$A_BUFLIST which points to a linked list
;  249          0379  1 !       of buffers to be freed up.
;  250          0380  1 !
;  251          0381  1 !       +----------+
;  252          0382  1 !       |   CIS    |
;  253          0383  1 !       |   ...    |
;  254          0384  1 !       |----------|      +-------- +          +---------+
;  255          0385  1 !       | BUFLIST  |----->|        |--------->|         |--->...
;  256          0386  1 !       +----------+      | bufptr |->buffer  | bufptr  |->buffer
;  257          0387  1 !                         +-------- +          +---------+
;  258          0388  1 !
;  259          0389  1 !       These bufffers get created as we expand symbols that were defined
;  260          0390  1 !       with DEFINE/COMMAND (we need to allocate new buffers to hold the
;  261          0391  1 !       expanded command). This happens in DBG$NGET_CMD. These buffers
;  262          0392  1 !       are freed in DBG$NCIS_REMOVE.
;  263          0393  1 !
;  264          0394  1 ! INPUTS
;  265          0395  1 !       BUFFER - address of a buffer. This address is to be added to the list.
;  266          0396  1 !       DBG$GL_CISHEAD - (implicit input) - current top CIS link
;  267          0397  1 !
;  268          0398  1 ! OUTPUTS
;  269          0399  1 !       The BUFLIST associated with DBG$GL_CISHEAD is added to.
;  270          0400  1 !
;  271          0401  2     BEGIN
;  272          0402  2     LOCAL
;  273          0403  2         NEWLINK: REF VECTOR[];
;  274          0404  2     NEWLINK = DBG$GET_MEMORY(2);
;  275          0405  2     NEWLINK[0] = .DBG$GL_CISHEAD[CIS$A_BUFLIST];
;  276          0406  2     NEWLINK[1] = .BUFFER;
;  277          0407  2     DBG$GL_CISHEAD[CIS$A_BUFLIST] = .NEWLINK;
;  278          0408  1     END;


                                        0000 00000 ADD_TO_BUFLIST:
                                                   .WORD   Save nothing                    ; 0371
                                    02  DD 00002             PUSHL   #2                      ; 0404
                00000000G     00    01  FB 00004             CALLS   #1, DBG$GET_MEMORY      ;
                          51 00000000G  00  D0 0000B         MOVL    DBG$GL_CISHEAD, R1      ; 0405
                          60       30   A1  D0 00012         MOVL    48(R1), (NEWLINK)       ;
                     04   A0       04   AC  D0 00016         MOVL    BUFFER, 4(NEWLINK)      ; 0406
                     30   A1            50  D0 0001B         MOVL    NEWLINK, 48(R1)         ; 0407
                                        04  0001F             RET                             ; 0408

; Routine Size:  32 bytes,    Routine Base:  DBG$CODE + 008F
```

```
;  280     0409  1 ROUTINE DBG$NGET_CMD ( INPUT_DESC, CMD_DESC, MESSAGE_VECT, P_EXPAN_FLAG) =
;  281     0410  1
;  282     0411  1 !++
;  283     0412  1 ! FUNCTIONAL DESCRIPTION:
;  284     0413  1 !
;  285     0414  1 !       This routine seperates the input line into one or more DEBUG commands. <cr>
;  286     0415  1 !       <ff>, and the null character (00) imply end of input line. Semi-colon (;)
;  287     0416  1 !       implies end of command.
;  288     0417  1 !
;  289     0418  1 !       This routine takes care of stripping the comments off the end of
;  290     0419  1 !       a DEBUG command. For all languages except C, the comment character is
;  291     0420  1 !       '!'. In C, '!' is an operator, so the pair of characters '/*' is
;  292     0421  1 !       the comment indicator (as in the language).  Since there are slight
;  293     0422  1 !       differences in the way a line is to be Uppercased and striped of comments
;  294     0423  1 !       we case on the language a call a specific routine to do these jobs.
;  295     0424  1 !
;  296     0425  1 ! FORMAL PARAMETERS:
;  297     0426  1 !
;  298     0427  1 !       input_desc -            a VAX standard descriptor of the entire input line
;  299     0428  1 !
;  300     0429  1 !       cmd_desc -              upon exit from this routine, a VAX standard descriptor
;  301     0430  1 !                               of a single potential DEBUG command
;  302     0431  1 !
;  303     0432  1 !       message_vect            - the address of a longword to contain the address
;  304     0433  1 !                                 of a message argument vector
;  305     0434  1 !       p_expand_flag -         optional fourth parameter which says whether to
;  306     0435  1 !                               expand defined names.
;  307     0436  1 !
;  308     0437  1 ! IMPLICIT INPUTS:
;  309     0438  1 !
;  310     0439  1 !       NONE
;  311     0440  1 !
;  312     0441  1 ! IMPLICIT OUTPUTS:
;  313     0442  1 !
;  314     0443  1 !       NONE
;  315     0444  1 !
;  316     0445  1 ! ROUTINE VALUE:
;  317     0446  1 !
;  318     0447  1 !       unsigned integer longword completion code
;  319     0448  1 !
;  320     0449  1 ! COMPLETION CODES:
;  321     0450  1 !
;  322     0451  1 !       sts$k_warning (0)       - the input line was found to be exhausted
;  323     0452  1 !
;  324     0453  1 !       sts$k_success (1)       - the comd_desc was updated to refer to a potential
;  325     0454  1 !                                 DEBUG command
;  326     0455  1 !
;  327     0456  1 !       sts$k_error (2)         - the input descriptor was found to contain nothing
;  328     0457  1 !                                 but a comment (! in first position)
;  329     0458  1 !
;  330     0459  1 !       sts$k_severe (4)        - error in input line
;  331     0460  1 !
;  332     0461  1 ! SIDE EFFECTS:
;  333     0462  1 !
;  334     0463  1 !       All lower case alphabetic characters are converted to upper case, except
;  335     0464  1 !       for strings enclosed withing single or double quote marks. A check
;  336     0465  1 !       is made for unprintable characters in the input line (error message generated
```

```
;  337          0466  1 !       and failure return).
;  338          0467  1 !
;  339          0468  1 !--
;  340          0469  1
;  341          0470  2     BEGIN
;  342          0471  2
;  343          0472  2     LOCAL
;  344          0473  2         CHAR_COUNT,                             ! Keeps count of characters
;  345          0474  2         CHAR_STRING     : REF VECTOR [,BYTE],   ! Vector of 1 byte characters
;  346          0475  2         CIS_DESC        : REF CIS$LINK,         ! Current command input stream.
;  347          0476  2         CMD_STRING      : REF VECTOR [, BYTE],  ! Counted string with the expansion of
;  348          0477  2                                                 ! the first token, if it has been
;  349          0478  2                                                 ! defined with DEFINE/COMMAND
;  350          0479  2         EXPAND_FLAG,                            ! Says whether or not to expand
;  351          0480  2                                                 !    defined names.
;  352          0481  2         STATUS,                                 ! status of calls
;  353          0482  2         QUOTE_CHAR,                             ! Contains quote char
;  354          0483  2         QUOTE_FLAG;                             ! Signifies quoted string
;  355          0484  2
;  356          0485  2     MAP
;  357          0486  2         INPUT_DESC      : REF dbo$stg_desc,
;  358          0487  2         CMD_DESC        : REF BLOCK [,BYTE];    ! We don't REF to dbg$stg_desc
;  359          0488  2                                                 ! because of the extra longword for
;  360          0489  2                                                 ! the initial dsc$a_pointer
;  361          0490  2     BUILTIN
;  362          0491  2         ACTUALCOUNT;
;  363          0492  2
;  364          0493  2
;  365          0494  2     ! Set the flag saying whether we want to expand defined names. The case
;  366          0495  2     ! where we do not want to is when we are called from DBG$NSAVE_BREAK_BUFFER
;  367          0496  2     ! to pick up the rest of a command.
;  368          0497  2     !
;  369          0498  2     IF ACTUALCOUNT() LSS 4
;  370          0499  2     THEN
;  371          0500  2         EXPAND_FLAG = TRUE
;  372          0501  2     ELSE
;  373          0502  2         EXPAND_FLAG = .P_EXPAND_FLAG;
;  374          0503  2
;  375          0504  2
;  376          0505  2     ! Initialize the command descriptor
;  377          0506  2     !
;  378          0507  2     cmd_desc [dsc$b_dtype] = dsc$k_dtype_t;
;  379          0508  2     cmd_desc [dsc$b_class] = dsc$k_class_s;
;  380          0509  2     cmd_desc [dsc$w_length] = 0;
;  381          0510  2     cmd_desc [dsc$a_pointer] = 0;
;  382          0511  2     cmd_desc [initial_ptr] = 0;
;  383          0512  2
;  384          0513  2
;  385          0514  2     ! Find a significant character
;  386          0515  2     !
;  387          0516  2     char_string = .input_desc [dsc$a_pointer];
;  388          0517  2     char_count = 0;
;  389          0518  2     WHILE .input_desc [dsc$w_length] GTR 0 DO
;  390          0519  3         BEGIN
;  391          0520  3         IF .char_string [.char_count] NEQ dbg$k_car_return
;  392          0521  3                                   AND
;  393          0522  3            .char_string [.char_count] NEQ dbg$k_line_feed
```

```
;  394     0523  3                                    AND
;  395     0524  3            .char_string [.char_count] NEQ dbg$k_null
;  396     0525  3                                    AND
;  397     0526  3            .char_string [.char_count] NEQ dbg$k_semicolon
;  398     0527  3                                    AND
;  399     0528  3            .char_string [.char_count] NEQ dbg$k_blank
;  400     0529  3                                    AND
;  401     0530  3            .char_string [.char_count] NEQ dbg$k_tab
;  402     0531  3        THEN
;  403     0532  3            EXITLOOP
;  404     0533  3        ELSE
;  405     0534  4            BEGIN
;  406     0535  4            char_count = .char_count + 1;
;  407     0536  4            input_desc [dsc$w_length] = .input_desc [dsc$w_length] - 1;
;  408     0537  3            END;
;  409     0538  2        END;
;  410     0539  2
;  411     0540  2    ! Return warning if there was no significant input on the line.
;  412     0541  2    !
;  413     0542  2    IF .input_desc [dsc$w_length] EQL 0
;  414     0543  2    THEN
;  415     0544  2        RETURN sts$k_warning;
;  416     0545  2
;  417     0546  2
;  418     0547  2    ! Set up the start verify pointer. This is done before stripping
;  419     0548  2    ! non-significant input to preserve the indentation.
;  420     0549  2    !
;  421     0550  2    start_verify_pointer = .input_desc [dsc$a_pointer];
;  422     0551  2
;  423     0552  2
;  424     0553  2    ! Update pointer to rest of string
;  425     0554  2    !
;  426     0555  2    input_desc [dsc$a_pointer] = char_string [.char_count];
;  427     0556  2
;  428     0557  2
;  429     0558  2    ! The next thing we do is check for the first token in the command line
;  430     0559  2    ! being a symbol defined with DEFINE/COMMAND.
;  431     0560  2    !
;  432     0561  2    IF .EXPAND_FLAG
;  433     0562  2    THEN
;  434     0563  3        BEGIN
;  435     0564  3        IF DBG$EXPAND_DEFINE_NAME (.INPUT_DESC, DEFINE_COMMAND, CMD_STRING)
;  436     0565  3        THEN
;  437     0566  4            BEGIN
;  438     0567  4            LOCAL
;  439     0568  4                BUFPTR,                 ! A pointer into the command buffer
;  440     0569  4                LENGTH,                 ! The length of the new command buffer
;  441     0570  4                NEW_BUFFER;                     ! Will point to the new command buffer.
;  442     0571  4
;  443     0572  4
;  444     0573  4            ! We need to allocate a new command buffer to hold the expanded
;  445     0574  4            ! token concatenated with the rest of the command.
;  446     0575  4            !
;  447     0576  4            LENGTH = .INPUT_DESC [DSC$W_LENGTH] + .CMD_STRING [0];
;  448     0577  4            NEW_BUFFER = DBG$GET_MEMORY((.LENGTH+3)/4);
;  449     0578  4
;  450     0579  4
```

```
;  451          0580  4          ! Copy the concatenated strings into the new buffer.
;  452          0581  4          !
;  453          0582  4          BUFPTR = CH$MOVE (.CMD_STRING [0], CMD_STRING [1], .NEW_BUFFER);
;  454          0583  4          BUFPTR = CH$MOVE (.INPUT_DESC[DSC$W_LENGTH], .INPUT_DESC[DSC$A_POINTER], .BUFPTR);
;  455          0584  4
;  456          0585  4
;  457          0586  4          ! Fill in the input descriptor to point to the new buffer.
;  458          0587  4          !
;  459          0588  4          INPUT_DESC[DSC$A_POINTER] = .NEW_BUFFER;
;  460          0589  4          INPUT_DESC[DSC$W_LENGTH] = .INPUT_DESC[DSC$W_LENGTH]+.CMD_STRING[0];
;  461          0590  4
;  462          0591  4
;  463          0592  4          ! We need to remember to free up the space occupied by NEW_BUFFER
;  464          0593  4          ! when we are done processing this CIS link. So, we
;  465          0594  4          ! put NEW_BUFFER onto the linked list of all the buffers allocated
;  466          0595  4          ! in this fashion. These will get freed up in CIS_REMOVE.
;  467          0596  4          !
;  468          0597  4          ADD_TO_BUFLIST (.NEW_BUFFER);
;  469          0598  4
;  470          0599  4
;  471          0600  4          ! Now re-do the code where we find a significant character
;  472          0601  4          !
;  473          0602  4          char_string = .input_desc [dsc$a_pointer];
;  474          0603  4          char_count = 0;
;  475          0604  4          WHILE .input_desc [dsc$w_length] GTR 0 DO
;  476          0605  5              BEGIN
;  477          0606  5              IF .char_string [.char_count] NEQ dbg$k_car_return
;  478          0607  5                                 AND
;  479          0608  5                 .char_string [.char_count] NEQ dbg$k_line_feed
;  480          0609  5                                 AND
;  481          0610  5                 .char_string [.char_count] NEQ dbg$k_null
;  482          0611  5                                 AND
;  483          0612  5                 .char_string [.char_count] NEQ dbg$k_semicolon
;  484          0613  5                                 AND
;  485          0614  5                 .char_string [.char_count] NEQ dbg$k_blank
;  486          0615  5                                 AND
;  487          0616  5                 .char_string [.char_count] NEQ dbg$k_tab
;  488          0617  5              THEN
;  489          0618  5                  EXITLOOP
;  490          0619  5              ELSE
;  491          0620  6                  BEGIN
;  492          0621  6                  char_count = .char_count + 1;
;  493          0622  6                  input_desc [dsc$w_length] = .input_desc [dsc$w_length] - 1;
;  494          0623  5                  END;
;  495          0624  4              END;
;  496          0625  4
;  497          0626  4          ! Again, if we have no significant input on the line then
;  498          0627  4          ! return warning.
;  499          0628  4          !
;  500          0629  4          IF .input_desc [dsc$w_length] EQL 0
;  501          0630  4          THEN
;  502          0631  4              RETURN sts$k_warning;
;  503          0632  4
;  504          0633  4          ! Set up the start verify pointer
;  505          0634  4          !
;  506          0635  4          start_verify_pointer = .input_desc [dsc$a_pointer];
;  507          0636  4
```

```
;  508     0637  4              ! Update pointer to rest of string
;  509     0638  4              !
;  510     0639  4              input_desc [dsc$a_pointer] = char_string [.char_count];
;  511     0640  3              END;
;  512     0641  2          END;
;  513     0642  2
;  514     0643  2
;  515     0644  2      ! Now case on the language and get the command and uppercase the characters.
;  516     0645  2      !
;  517     0646  2      CASE .dbg$gb_language FROM dbg$k_min_language TO dbg$k_max_language OF
;  518     0647  2          SET
;  519     0648  2
;  520     0649  2          [dbg$k_macro, dbg$k_fortran, dbg$k_bliss,
;  521     0650  2           dbg$k_cobol, dbg$k_basic, dbg$k_pli,
;  522     0651  2           dbg$k_pascal, dbg$k_rpg, dbg$k_unknown,
;  523     0652  2           INRANGE, OUTRANGE]:
;  524     0653  2              status = get_normal_cmd_string(.input_desc, .cmd_desc, .cis_desc, .message_vect);
;  525     0654  2
;  526     0655  2          [dbg$k_c]:
;  527     0656  2              status = get_c_cmd_string(.input_desc, .cmd_desc, .cis_desc, .message_vect);
;  528     0657  2
;  529     0658  2          [dbg$k_ada]:
;  530     0659  2              status = get_ada_cmd_string(.input_desc, .cmd_desc, .cis_desc, .message_vect);
;  531     0660  2
;  532     0661  2          TES;
;  533     0662  2
;  534     0663  2      ! If an error occured return with status.
;  535     0664  2      !
;  536     0665  2      IF NOT .status
;  537     0666  2      THEN
;  538     0667  2          RETURN .status;
;  539     0668  2
;  540     0669  2      ! Delete all leading end of command signifiers from the input string
;  541     0670  2      !
;  542     0671  2      char_string = .input_desc [dsc$a_pointer];
;  543     0672  2
;  544     0673  2      WHILE .input_desc [dsc$w_length] GTR 0
;  545     0674  2      DO
;  546     0675  3          BEGIN
;  547     0676  3
;  548     0677  3              IF .char_string [0] NEQ dbg$k_car_return
;  549     0678  3                              AND
;  550     0679  3                 .char_string [0] NEQ dbg$k_line_feed
;  551     0680  3                              AND
;  552     0681  3                 .char_string [0] NEQ dbg$k_null
;  553     0682  3                              AND
;  554     0683  3                 .char_string [0] NEQ dbg$k_semicolon
;  555     0684  3              THEN
;  556     0685  3                  EXITLOOP;
;  557     0686  3
;  558     0687  3              input_desc [dsc$w_length] = .input_desc [dsc$w_length] - 1;
;  559     0688  3              input_desc [dsc$a_pointer] = char_string [1];
;  560     0689  3              char_string = .input_desc [dsc$a_pointer];
;  561     0690  3
;  562     0691  2          END;
;  563     0692  2
;  564     0693  2
```

```
;  565          0694  2     ! Save a pointer to the new input descriptor so that dbg$nsave_filespec
;  566          0695  2     ! can use it.
;  567          0696  2     !
;  568          0697  2     save_input_desc = .input_desc;
;  569          0698  2
;  570          0699  2     RETURN sts$k_success;
;  571          0700  1     END;                              !End of dbg$nget_cmd
; INFO#250                L1:0653
; Referenced LOCAL symbol CIS_DESC is probably not initialized
```

```
                        0FFC 00000 DBG$NGET_CMD:
                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                ; 0409
5E                  08  C2 00002        SUBL2   #8, SP                                       ;
04                  6C  91 00005        CMPB    (AP), #4                                     ; 0498
                    05  1E 00008        BGEQU   1$                                           ;
51                  01  D0 0000A        MOVL    #1, EXPAND_FLAG                              ; 0500
                    04  11 0000D        BRB     2$                                           ;
51          10      AC  D0 0000F 1$:    MOVL    P_EXPAND_FLAG, EXPAND_FLAG                   ; 0502
59          08      AC  D0 00013 2$:    MOVL    CMD_DESC, R9                                 ; 0507
69 010E0000         8F  D0 00017        MOVL    #17694720, (R9)                              ; 0509
            04      A9  7C 0001E        CLRQ    4(R9)                                        ; 0510
56          04      AC  D0 00021        MOVL    INPUT_DESC, R6                               ; 0516
5A          04      A6  9E 00025        MOVAB   4(R6), R10                                   ;
58                  6A  D0 00029        MOVL    (R10), CHAR_STRING                           ;
                    6E  D4 0002C        CLRL    CHAR_COUNT                                   ; 0517
                    66  B5 0002E 3$:    TSTW    (R6)                                         ; 0518
                    28  13 00030        BEQL    5$                                           ;
50          00 BE48  9A 00032        MOVZBL  @CHAR_COUNT[CHAR_STRING], R0                 ; 0520
0D                  50  91 00037        CMPB    R0, #T3                                      ;
                    18  13 0003A        BEQL    4$                                           ;
0A                  50  91 0003C        CMPB    R0, #10                                      ; 0522
                    13  13 0003F        BEQL    4$                                           ;
                    50  D5 00041        TSTL    R0                                           ; 0524
                    0F  13 00043        BEQL    4$                                           ;
3B                  50  91 00045        CMPB    R0, #59                                      ; 0526
                    0A  13 00048        BEQL    4$                                           ;
20                  50  91 0004A        CMPB    R0, #32                                      ; 0528
                    05  13 0004D        BEQL    4$                                           ;
09                  50  91 0004F        CMPB    R0, #9                                       ; 0530
                    06  12 00052        BNEQ    5$                                           ;
                    6E  D6 00054 4$:    INCL    CHAR_COUNT                                   ; 0535
                    66  B7 00056        DECW    (R6)                                         ; 0536
                    D4  11 00058        BRB     3$                                           ; 0518
                    66  B5 0005A 5$:    TSTW    (R6)                                         ; 0542
                    03  12 0005C        BNEQ    6$                                           ;
                011C  31 0005E        BRW     21$                                          ;
   00000000'        EF  6A  D0 00061 6$:    MOVL    (R10), START_VERIFY_POINTER              ; 0550
6A                  58  6E  C1 00068        ADDL3   CHAR_COUNT, CHAR_STRING, (R10)           ; 0555
                    03  51  E8 0006C        BLBS    EXPAND_FLAG, 7$                          ; 0561
                    008F  31 0006F        BRW     12$                                          ;
                    04  AE  9F 00072 7$:    PUSHAB  CMD_STRING                               ; 0564
                        02  DD 00075        PUSHL   #2                                       ;
                        56  DD 00077        PUSHL   R6                                       ;
```

```
                00000000G   00              03  FB 00079          CALLS   #3, DBG$EXPAND_DEFINE_NAME                      ;
                            7E              50  E9 00080          BLBC    R0, 12$                                         ;
                            57        04    AE  D0 00083          MOVL    CMD_STRING, R7                                  ; 0576
                            50              66  3C 00087          MOVZWL  (R6)+, LENGTH                                   ;
                            51              67  9A 0008A          MOVZBL  (R7), R1                                        ;
                            50              51  C0 0008D          ADDL2   R1, LENGTH                                      ;
                            50              03  C0 00090          ADDL2   #3, R0                                          ; 0577
                      7E                    50  C4 00093          MULL3   #4, R0, -(SP)                                   ;
                00000000G   00              01  FB 00097          CALLS   #1, DBG$GET_MEMORY                              ;
                            5B              50  D0 0009E          MOVL    R0, NEW_BUFFER                                  ;
                            50              67  9A 000A1          MOVZBL  (R7), R0                                        ; 0582
                6B          01              50  28 000A4          MOVC3   R0, 1(R7), (NEW_BUFFER)                         ;
                63          00              66  28 000A9          MOVC3   (R6), @0(R10), (BUFPTR)                         ; 0583
                            6A              5B  D0 000AE          MOVL    NEW_BUFFER, (R10)                               ; 0588
                            50              67  9A 000B1          MOVZBL  (R7)+, R0                                       ; 0589
                            66              50  A0 000B4          ADDW2   R0, (R6)                                        ;
                            5B              DD 000B7              PUSHL   NEW_BUFFER                                      ; 0597
                      FF22      CF          01  FB 000B9          CALLS   #1, ADD_TO_BUFLIST                              ;
                            58              6A  D0 000BE          MOVL    (R10), CHAR_STRING                              ; 0602
                                            6E  D4 000C1          CLRL    CHAR_COUNT                                      ; 0603
                                            66  B5 000C3  8$:     TSTW    (R6)                                            ; 0604
                                            28  13 000C5          BEQL    10$                                             ;
                            50        00 BE48   9A 000C7          MOVZBL  @CHAR_COUNT[CHAR_STRING], R0                    ; 0606
                            0D              50  91 000CC          CMPB    R0, #13                                         ;
                                            18  13 000CF          BEQL    9$                                              ;
                            0A              50  91 000D1          CMPB    R0, #10                                         ; 0608
                                            13  13 000D4          BEQL    9$                                              ;
                                            50  D5 000D6          TSTL    R0                                              ; 0610
                                            0F  13 000D8          BEQL    9$                                              ;
                            3B              50  91 000DA          CMPB    R0, #59                                         ; 0612
                                            0A  13 000DD          BEQL    9$                                              ;
                            20              50  91 000DF          CMPB    R0, #32                                         ; 0614
                                            05  13 000E2          BEQL    9$                                              ;
                            09              50  91 000E4          CMPB    R0, #9                                          ; 0616
                                            06  12 000E7          BNEQ    10$                                             ;
                                            6E  D6 000E9  9$:     INCL    CHAR_COUNT                                      ; 0621
                                            66  B7 000EB          DECW    (R6)                                            ; 0622
                                            D4  11 000ED          BRB     8$                                              ; 0604
                                            66  B5 000EF  10$:     TSTW    (R6)                                            ; 0629
                                            03  12 000F1          BNEQ    11$                                             ;
                                          0087  31 000F3          BRW     21$                                             ;
                      00000000'             EF  6A D0 000F6 11$:  MOVL    (R10), START_VERIFY_POINTER                     ; 0635
                            6A              58  6E C1 000FD      ADDL3   CHAR_COUNT, CHAR_STRING, (R10)                  ; 0639
                            0A              00  00000000G 00  8F 00101 12$:  CASEB   DBG$GB_LANGUAGE, #0, #10             ; 0646
0016                0016            0016        0016      00109 13$:  .WORD   14$-13$,-                                   ;
0026                0016            0016        0016      00111             14$-13$,-                                     ;
                    0016            0036        0016      00119             14$-13$,-                                     ;
                                                                            14$-13$,-                                     ;
                                                                            14$-13$,-                                     ;
                                                                            14$-13$,-                                     ;
                                                                            14$-13$,-                                     ;
                                                                            14$-13$,-                                     ;
                                                                            15$-13$,-                                     ;
                                                                            14$-13$,-                                     ;
                                                                            16$-13$,-                                     ;
                                                                            14$-13$                                       ;
                                                  0C    AC  DD 0011F 14$:   PUSHL   MESSAGE_VECT                          ; 0653
                                                        50  DD 00122        PUSHL   CIS_DESC                              ;
```

```
                        0240   8F  BB 00124          PUSHR   #^M<R6,R9>                                ;
              0000V     CF         04 FB 00128       CALLS   #4, GET_NORMAL_CMD_STRING                 ;
                                   1E 11 0012D       BRB     17$                                       ;
                          0C   AC  DD 0012F 15$:     PUSHL   MESSAGE_VECT                              ; 0656
                               50  DD 00132          PUSHL   CIS_DESC                                  ;
                        0240   8F  BB 00134          PUSHR   #^M<R6,R9>                                ;
              0000V     CF         04 FB 00138       CALLS   #4, GET_C_CMD_STRING                      ;
                                   0E 11 0013D       BRB     17$                                       ;
                          0C   AC  DD 0013F 16$:     PUSHL   MESSAGE_VECT                              ; 0659
                               50  DD 00142          PUSHL   CIS_DESC                                  ;
                        0240   8F  BB 00144          PUSHR   #^M<R6,R9>                                ;
              0000V     CF         04 FB 00148       CALLS   #4, GET_ADA_CMD_STRING                    ;
                        2F         50 E9 0014D 17$:  BLBC    STATUS, 22$                               ; 0665
                        58         6A D0 00150 18$:  MOVL    (R10), CHAR_STRING                        ; 0671
                                   66 B5 00153       TSTW    (R6)                                      ; 0673
                                   1B 13 00155       BEQL    20$                                       ;
                        0D         68 91 00157       CMPB    (CHAR_STRING), #13                        ; 0677
                                   0E 13 0015A       BEQL    19$                                       ;
                        0A         68 91 0015C       CMPB    (CHAR_STRING), #10                        ; 0679
                                   09 13 0015F       BEQL    19$                                       ;
                                   68 95 00161       TSTB    (CHAR_STRING)                             ; 0681
                                   05 13 00163       BEQL    19$                                       ;
                        3B         68 91 00165       CMPB    (CHAR_STRING), #59                        ; 0683
                                   08 12 00168       BNEQ    20$                                       ;
                                   66 B7 0016A 19$:  DECW    (R6)                                      ; 0687
                        6A   01    A8 9E 0016C       MOVAB   1(R8), (R10)                              ; 0688
                                   DE 11 00170       BRB     18$                                       ; 0689
    00000000'           EF         56 D0 00172 20$:  MOVL    R6, SAVE_INPUT_DESC                       ; 0697
                        50         01 D0 00179       MOVL    #1, R0                                    ; 0699
                                      04 0017C       RET                                               ;
                                   50 D4 0017D 21$:  CLRL    R0                                        ; 0700
                                      04 0017F 22$:  RET                                               ;

; Routine Size:  384 bytes,    Routine Base:  DBG$CODE + 00AF
```

```
;  573          0701  1 GLOBAL ROUTINE DBG$NKILL_CMD (MESSAGE_VECT) =
;  574          0702  1
;  575          0703  1 !++
;  576          0704  1 ! FUNCTIONAL DESCRIPTION:
;  577          0705  1 !
;  578          0706  1 !       Invocation ot this routine takes place when an invalid debug command is
;  579          0707  1 !       encountered during parsing.
;  580          0708  1 !
;  581          0709  1 ! FORMAL PARAMETERS:
;  582          0710  1 !
;  583          0711  1 !       message_vect    - the address of a longword to contain the address of a
;  584          0712  1 !                         message argument vector
;  585          0713  1 !
;  586          0714  1 ! IMPLICIT INPUTS:
;  587          0715  1 !
;  588          0716  1 !       NONE
;  589          0717  1 !
;  590          0718  1 ! IMPLICIT OUTPUTS:
;  591          0719  1 !
;  592          0720  1 !       On error return, a message argument vector is constructed
;  593          0721  1 !
;  594          0722  1 ! ROUTINE VALUE:
;  595          0723  1 !
;  596          0724  1 !       An unsigned integer longword completion code
;  597          0725  1 !
;  598          0726  1 ! COMPLETION CODES:
;  599          0727  1 !
;  600          0728  1 !       sts$k_success (1) - success
;  601          0729  1 !
;  602          0730  1 !       sts$k_severe  (4) - failure. message returned.
;  603          0731  1 !
;  604          0732  1 ! SIDE EFFECTS:
;  605          0733  1 !
;  606          0734  1 !       The present input buffer is discarded
;  607          0735  1 !
;  608          0736  1 !--
;  609          0737  1
;  610          0738  2     BEGIN
;  611          0739  2
;  612          0740  2     ! Simply blow away the rest of the input line
;  613          0741  2     !
;  614          0742  2     IF NOT dbg$nend_of_input (.message_vect)
;  615          0743  2     THEN
;  616          0744  2         RETURN sts$k_severe;
;  617          0745  2
;  618          0746  2     RETURN sts$k_success;
;  619          0747  2
;  620          0748  1     END;                        ! End of dbg$n_kill_cmd


                                    0000 00000          .ENTRY  DBG$NKILL_CMD, Save nothing                 ; 0701
                              04    AC  DD 00002        PUSHL   MESSAGE_VECT                                ; 0742
                    0000V     CF        01  FB 00005    CALLS   #1, DBG$NEND_OF_INPUT                       ;
                              04        50  E8 0000A    BLBS    R0, 1$                                      ;
```

```
                          50        04  D0 0000D          MOVL    #4, R0                          ; 0744
                                        04 00010          RET                                     ;
                          50        01  D0 00011 1$:      MOVL    #1, R0                          ; 0746
                                        04 00014          RET                                     ; 0748

; Routine Size:  21 bytes,    Routine Base:  DBG$CODE + 022F

;  621          0749  1
```

```
;  623          0750  1 GLOBAL ROUTINE DBG$NEND_OF_INPUT (MESSAGE_VECT) =
;  624          0751  1
;  625          0752  1 !++
;  626          0753  1 ! FUNCTIONAL DESCRIPTION:
;  627          0754  1 !
;  628          0755  1 !       This routine is invoked when the present input line is exhausted.
;  629          0756  1 !
;  630          0757  1 ! FORMAL PARAMETERS:
;  631          0758  1 !
;  632          0759  1 !       message_vect    - the address of a longword to contain the address of a
;  633          0760  1 !                         message argument vector
;  634          0761  1 !
;  635          0762  1 ! IMPLICIT INPUTS:
;  636          0763  1 !
;  637          0764  1 !       NONE
;  638          0765  1 !
;  639          0766  1 ! IMPLICIT OUTPUTS:
;  640          0767  1 !
;  641          0768  1 !       On error return, a message argument vector is returned.
;  642          0769  1 !
;  643          0770  1 ! ROUTINE VALUE:
;  644          0771  1 !
;  645          0772  1 !       An unsigned integer longword completion code
;  646          0773  1 !
;  647          0774  1 ! COMPLETION CODES:
;  648          0775  1 !
;  649          0776  1 !       sts$k_success (1)       - success. input removed
;  650          0777  1 !
;  651          0778  1 !       sts$k_severe  (4)       - failure. message argument vector returned.
;  652          0779  1 !
;  653          0780  1 ! SIDE EFFECTS:
;  654          0781  1 !
;  655          0782  1 !       A link is removed from the head of the command input stream.
;  656          0783  1 !
;  657          0784  1 !--
;  658          0785  1
;  659          0786  2     BEGIN
;  660          0787  2
;  661          0788  2     ! Remove a node from the cis
;  662          0789  2     !
;  663          0790  2     IF NOT dbg$ncis_remove (FALSE, .message_vec*)
;  664          0791  2     THEN
;  665          0792  2         RETURN sts$k_severe;
;  666          0793  2
;  667          0794  2     RETURN sts$k_success;
;  668          0795  2
;  669          0796  1     END;                        ! End of dbg$nend_of_input


                                    0000 00000          .ENTRY  DBG$NEND_OF_INPUT, Save nothing                 ; 0750
                              04    AC   DD 00002       PUSHL   MESSAGE_VECT                                    ; 0790
                                    7E   D4 00005       CLRL    -(SP)                                           ;
00000000G   00                      02   FB 00007       CALLS   #2, DBG$NCIS_REMOVE                             ;
                  04                50   E8 0000E       BLBS    R0, 1$                                          ;
```

```
                50              04  D0 00011          MOVL    #4, R0                          ; 0792
                                    04 00014          RET                                     ;
                50              01  D0 00015 1$:      MOVL    #1, R0                          ; 0794
                                    04 00018          RET                                     ; 0796

; Routine Size:  25 bytes,    Routine Base:  DBG$CODE + 0244
```

```
;  671          0797  1 GLOBAL ROUTINE DBG$NSAVE_FILESP (INPUT_DESC, FILE, MESSAGE_VECT) =
;  672          0798  1
;  673          0799  1 !++
;  674          0800  1 ! FUNCTIONAL DESCRIPTION:
;  675          0801  1 !
;  676          0802  1 !       This routine gathers a file spec from the command line. Since filespecs
;  677          0803  1 !       may be in the form a.b;12, the version number will not be contained in the
;  678          0804  1 !       command descriptor string as dbg$nget_command regards a ';' as end of command.
;  679          0805  1 !       Consequently, look-ahead must be performed on the entire input line string
;  680          0806  1 !       to locate the version number of a file spec. Quoted filespec strings are
;  681          0807  1 !       also allowed as this construction is necessary to specify filespecs that
;  682          0808  1 !       contain disk specifiers or sub-directories.
;  683          0809  1 !
;  684          0810  1 !       A filespec is returned in the form of a counted string, the storage for which
;  685          0811  1 !       is allocated from non-listed storage.
;  686          0812  1 !
;  687          0813  1 ! FORMAL PARAMETERS:
;  688          0814  1 !
;  689          0815  1 !       input_desc -            the present command VAX standard string descriptor
;  690          0816  1 !
;  691          0817  1 !       file    -               the address of a longword to contain the filespec
;  692          0818  1 !
;  693          0819  1 !       message_vect -          the address of a longword to contain the address
;  694          0820  1 !                               of a message argument vector
;  695          0821  1 !
;  696          0822  1 ! IMPLICIT INPUTS:
;  697          0823  1 !
;  698          0824  1 !       save_input_desc -       VAX standard string descriptor of the rest of the
;  699          0825  1 !                               complete input line.
;  700          0826  1 !
;  701          0827  1 ! IMPLICIT OUTPUTS:
;  702          0828  1 !
;  703          0829  1 !       A counted string representing the filspec on success, or a message argument
;  704          0830  1 !       vector on failure.
;  705          0831  1 !
;  706          0832  1 ! ROUTINE VALUE:
;  707          0833  1 !
;  708          0834  1 !       An unsigned integer longword completion code
;  709          0835  1 !
;  710          0836  1 ! COMPLETION CODES:
;  711          0837  1 !
;  712          0838  1 !       sts$k_success (1) -     filespec collected.
;  713          0839  1 !
;  714          0840  1 !       sts$k_severe  (4) -     the filespec was not collected. message vector returned.
;  715          0841  1 !
;  716          0842  1 ! SIDE EFFECTS:
;  717          0843  1 !
;  718          0844  1 !       Both the command descriptor and the line input descriptor may be updated.
;  719          0845  1 !       The command descriptor (input_desc) is always updated to reflect exhaused
;  720          0846  1 !       input. That is, the filespec is taken to be everything left in the command
;  721          0847  1 !       string. The input line descriptor (save_input_desc) will be updated to
;  722          0848  1 !       point past an explicit version number string.
;  723          0849  1 !
;  724          0850  1 !--
;  725          0851  2     BEGIN
;  726          0852  2
;  727          0853  2     FORWARD ROUTINE
```

```
;  728          0854  2          NEXT_CHAR,                          ! Returns the next character from input
;  729          0855  2          LOOKAHEAD_CHAR,                     ! Returns next char but does not
;  730          0856  2                                              !    advance pointer.
;  731          0857  2          FILENAME,                           ! Extracts a file spec name
;  732          0858  2          FILETYPE,                           ! Extracts a file spec type
;  733          0859  2          VERSION_NUMBER,                     ! Extracts a file spec version number
;  734          0860  2          QUOTED_FILESPEC;                    ! Extracts a quoted filespec
;  735          0861  2
;  736          0862  2      LOCAL
;  737          0863  2          NEXT_PTR,                           ! Pointer to position in filespec
;  738          0864  2          FILESPEC        : REF VECTOR [,BYTE], ! Contains the file spec
;  739          0865  2          NAME            : REF VECTOR [,BYTE], ! Contains the file name
;  740          0866  2          TYPE            : REF VECTOR [,BYTE], ! Contains the file type
;  741          0867  2          VERSION         : REF VECTOR [,BYTE], ! Contains the file version number
;  742          0868  2          QUOTED_STRING;                      ! Pointer to a quoted file spec
;  743          0869  2
;  744          0870  2      OWN
;  745          0871  2          ERROR_VECTOR,                       ! Pointer to message argument vector
;  746          0872  2          CHAR            : BYTE,             ! Character of input buffer. This
;  747          0873  2                                              ! variable is 'own'ed to make it a
;  748          0874  2                                              ! 'local global'.
;  749          0875  2          ERROR_STG_DESC  : dbg$stg_desc,     ! For error reporting
;  750          0876  2          CMD_DESC        : REF dbg$stg_desc; ! Descriptor for command input
;  751          0877  2
;  752          0878  2      BIND
;  753          0879  2          ONE_QUOTE = UPLIT BYTE (dbg$k_quote),   ! For error reporting
;  754          0880  2          TWO_QUOTE = UPLIT BYTE (dbg$k_dblquote); ! For error reporting
;  755          0881  2
;  756          0882  2      !
;  757          0883  2      !
;  758          0884  2      !
;  759          0885  2      !
;  760          0886  2      !
;  761          0887  2
;  762          0888  2      ROUTINE NEXT_CHAR =
;  763          0889  2
;  764          0890  2      !++
;  765          0891  2      !   This routine returns the next character of input. This character
;  766          0892  2      !   may come from the command descriptor or the input line descriptor strings.
;  767          0893  2      !--
;  768          0894  2
;  769          0895  3          BEGIN
;  770          0896  3
;  771          0897  3          ! Take the character from the command input descriptor or the line
;  772          0898  3          ! input descriptor.
;  773          0899  3          !
;  774          0900  3          IF .cmd_desc [dsc$w_length] GTR 0
;  775          0901  3          THEN
;  776          0902  4              BEGIN
;  777          0903  4
;  778          0904  4                  ! Take the char from the command input buffer
;  779          0905  4                  !
;  780          0906  4                  cmd_desc [dsc$w_length] = .cmd_desc [dsc$w_length] - 1;
;  781          0907  4                  char = .(.cmd_desc [dsc$a_pointer]) <0, 8, 0>;
;  782          0908  4                  cmd_desc [dsc$a_pointer] = .cmd_desc [dsc$a_pointer] + 1;
;  783          0909  4
;  784          0910  4
```

```
;  785          0911  4                ! Map a <cr> into a semicolon
;  786          0912  4                !
;  787          0913  4                IF .char EQL dbg$k_car_return
;  788          0914  4                THEN
;  789          0915  4                    char = dbg$k_semicolon;
;  790          0916  4            END
;  791          0917  4
;  792          0918  3        ELSE
;  793          0919  4            BEGIN
;  794          0920  4
;  795          0921  4                ! Take the character from the line input buffer.
;  796          0922  4                ! Check for exhausted input.
;  797          0923  4                !
;  798          0924  4                IF .save_input_desc [dsc$w_length] LEQ 0
;  799          0925  4                THEN
;  800          0926  4                    RETURN sts$k_error;
;  801          0927  4
;  802          0928  4
;  803          0929  4                ! We map a <cr> from the cmd buffer to a semicolon. Make sure
;  804          0930  4                ! that we do not return the semicolon twice.
;  805          0931  4                !
;  806          0932  4                IF .char EQL dbg$k_semicolon
;  807          0933  4                        AND
;  808          0934  4                    .(.save_input_desc [dsc$a_pointer]) <0, 8, 0> EQL dbg$k_semicolon
;  809          0935  4                THEN
;  810          0936  5                    BEGIN
;  811          0937  5                        char = 0;
;  812          0938  5                        save_input_desc [dsc$w_length] = .save_input_desc [dsc$w_length] - 1;
;  813          0939  5                        save_input_desc [dsc$a_pointer] = .save_input_desc [dsc$a_pointer] + 1;
;  814          0940  5                        RETURN next_char ();
;  815          0941  4                    END;
;  816          0942  4
;  817          0943  4                save_input_desc [dsc$w_length] = .save_input_desc [dsc$w_length] - 1;
;  818          0944  4                char = .(.save_input_desc [dsc$a_pointer]) <0, 8, 0>;
;  819          0945  4                save_input_desc [dsc$a_pointer] = .save_input_desc [dsc$a_pointer] + 1;
;  820          0946  3            END;
;  821          0947  3
;  822          0948  3        RETURN sts$k_success;
;  823          0949  3
;  824          0950  2        END;            ! End of next_chars


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

                              27  00000 P.AAA:  .BYTE   39                                                    ;
                              22  00001 P.AAB:  .BYTE   34                                                    ;

                                        .PSECT  DBG$OWN,NOEXE,  PIC,2

                                  0001C ERROR_VECTOR:
                                        .BLKB   4
                                  00020 CHAR:   .BLKB   1
                                  00021         .BLKB   3
                                  00024 ERROR_STG_DESC:
                                        .BLKB   12
                                  00030 CMD_DESC:
```

```
                                .BLKB   4

                        ONE_QUOTE=          P.AAA
                        TWO_QUOTE=          P.AAB


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

                            0004 00000 NEXT_CHAR:
                                        .WORD   Save R2                         ; 0888
52 00000000'    EF  9E 00002            MOVAB   CHAR, R2                        ;
50      10      A2  D0 00009            MOVL    CMD_DESC, R0                    ; 0900
                60  B5 0000D            TSTW    (R0)                            ;
                13  13 0000F            BEQL    1$                              ;
                60  B7 00011            DECW    (R0)                            ; 0906
62      04      B0  90 00013            MOVB    @4(R0), CHAR                    ; 0907
        04      A0  D6 00017            INCL    4(R0)                           ; 0908
0D              62  91 0001A            CMPB    CHAR, #13                       ; 0913
                35  12 0001D            BNEQ    4$                              ;
62              3B  90 0001F            MOVB    #59, CHAR                       ; 0915
                30  11 00022            BRB     4$                              ; 0900
50      F4      A2  D0 00024 1$:        MOVL    SAVE_INPUT_DESC, R0             ; 0924
                60  B5 00028            TSTW    (R0)                            ;
                07  12 0002A            BNEQ    2$                              ;
50              02  D0 0002C            MOVL    #2, R0                          ; 0926
                    04 0002F            RET                                     ;
3B              62  91 00030 2$:        CMPB    CHAR, #59                       ; 0932
                12  12 00033            BNEQ    3$                              ;
3B      04      B0  91 00035            CMPB    @4(R0), #59                     ; 0934
                0C  12 00039            BNEQ    3$                              ;
                62  94 0003B            CLRB    CHAR                            ; 0937
                60  B7 0003D            DECW    (R0)                            ; 0938
        04      A0  D6 0003F            INCL    4(R0)                           ; 0939
    BA  AF          00  FB 00042        CALLS   #0, NEXT_CHAR                   ; 0940
                    04 00046            RET                                     ;
50      F4      A2  D0 00047 3$:        MOVL    SAVE_INPUT_DESC, R0             ; 0943
                60  B7 0004B            DECW    (R0)                            ;
62      04      B0  90 0004D            MOVB    @4(R0), CHAR                    ; 0944
        04      A0  D6 00051            INCL    4(R0)                           ; 0945
50              01  D0 00054 4$:        MOVL    #1, R0                          ; 0948
                    04 00057            RET                                     ; 0950

; Routine Size:  88 bytes,    Routine Base:  DBG$CODE + 025D

;  825          0951  2
;  826          0952  2     !
;  827          0953  2     !
;  828          0954  2     !
;  829          0955  2     !
;  830          0956  2     !
;  831          0957  2
;  832          0958  2     ROUTINE LOOKAHEAD_CHAR =
;  833          0959  2     !++
;  834          0960  2     ! This routine is like NEXT_CHAR in that it returns the next character,
;  835          0961  2     ! but it does not advance the pointer. It can this be used for lookahead.
;  836          0962  2     !--
```

```
;  837          0963  3          BEGIN
;  838          0964  3
;  839          0965  3          ! Take the character from the command input descriptor or the line
;  840          0966  3          ! input descriptor.
;  841          0967  3          !
;  842          0968  3          IF .cmd_desc [dsc$w_length] GTR 0
;  843          0969  3          THEN
;  844          0970  4              BEGIN
;  845          0971  4              char = .(.cmd_desc [dsc$a_pointer]) <0, 8, 0>;
;  846          0972  4              ! Map a <cr> into a semicolon
;  847          0973  4              IF .char EQL dbg$k_car_return
;  848          0974  4              THEN
;  849          0975  4                  char = dbg$k_semicolon;
;  850          0976  4              END
;  851          0977  3          ELSE
;  852          0978  4              BEGIN
;  853          0979  4              ! Take the character from the line input buffer.
;  854          0980  4              ! Check for exhausted input.
;  855          0981  4              IF .save_input_desc [dsc$w_length] LEQ 0
;  856          0982  4              THEN
;  857          0983  4                  RETURN sts$k_error;
;  858          0984  4
;  859          0985  4              ! We map a <cr> from the cmd buffer to a semicolon. Make sure
;  860          0986  4              ! that we do not return the semicolon twice.
;  861          0987  4              !
;  862          0988  4              IF .char EQL dbg$k_semicolon
;  863          0989  4                  AND
;  864          0990  4                  .(.save_input_desc [dsc$a_pointer]) <0, 8, 0> EQL dbg$k_semicolon
;  865          0991  4              THEN
;  866          0992  5                  BEGIN
;  867          0993  5                  char = );
;  868          0994  5                  RETURN lookahead_char();
;  869          0995  4                  END;
;  870          0996  4
;  871          0997  4              char = .(.save_input_desc [dsc$a_pointer]) <0, 8, 0>;
;  872          0998  3              END;
;  873          0999  3
;  874          1000  3          RETURN sts$k_success;
;  875          1001  2          END, ! lookahead_char


                                 0004 00000 LOOKAHEAD_CHAR:
                                                .WORD   Save R2                                 ; 0958
    52 00000000'   EF   9E 00002                MOVAB   CHAR, R2                                ;
    50         10  A2   D0 00009                MOVL    CMD_DESC, R0                            ; 0968
               60  B5   0000D                   TSTW    (R0)                                    ;
               0E  13   0000F                   BEQL    1$                                      ;
    62         04  B0   90 00011                MOVB    @4(R0), CHAR                            ; 0971
    0D             62   91 00015                CMPB    CHAR, #13                               ; 0973
                   2B   12 00018                BNEQ    4$                                      ;
    62             3B   90 0001A                MOVB    #59, CHAR                               ; 0975
                   26   11 0001D                BRB     4$                                      ; 0968
    50         F4  A2   D0 0001F 1$:            MOVL    SAVE_INPUT_DESC, R0                     ; 0981
               60  B5   00023                   TSTW    (R0)                                    ;
```

```
                                     04  12 00025          BNEQ    2$                                      ;
                               50    02  D0 00027          MOVL    #2, R0                                  ; 0983
                                     04  0002A            RET                                             ;
                               3B    62  91 0002B 2$:     CMPB    CHAR, #59                               ; 0988
                                     0D  12 0002E          BNEQ    3$                                      ;
                               3B  04  B0  91 00030        CMPB    @4(R0), #59                             ; 0990
                                     07  12 00034          BNEQ    3$                                      ;
                                     62  94 00036          CLRB    CHAR                                    ; 0993
                           C4  AF    00  FB 00038          CALLS   #0, LOOKAHEAD_CHAR                      ; 0994
                                     04  0003C            RET                                             ;
                               50  F4  A2  D0 0003D 3$:     MOVL    SAVE_INPUT_DESC, R0                     ; 0997
                               62  04  B0  90 00041        MOVB    @4(R0), CHAR                            ;
                               50      01  D0 00045 4$:     MOVL    #1, R0                                  ; 1000
                                     04  00048            RET                                             ; 1001

; Routine Size:  73 bytes,    Routine Base:  DBG$CODE + 02B5

;  876          1002  2     !
;  877          1003  2     !
;  878          1004  2     !
;  879          1005  2     !
;  880          1006  2     !
;  881          1007  2
;  882          1008  2     ROUTINE FILENAME =
;  883          1009  2
;  884          1010  2     !++
;  885          1011  2     !   This routine collects the filespec file name string. That is, all characters
;  886          1012  2     !   up to a '.' or end of line.
;  887          1013  2     !--
;  888          1014  2
;  889          1015  3         BEGIN
;  890          1016  3
;  891          1017  3         LOCAL
;  892          1018  3             NAME_BUF : REF VECTOR [,BYTE],      ! Contains the filename string
;  893          1019  3             I;                                  ! Counter
;  894          1020  3
;  895          1021  3         ! The filename cannot be longer than the command input buffer. Get storage
;  896          1022  3         ! to hold the name string.
;  897          1023  3         !
;  898          1024  3         name_buf = dbg$get_tempmem (\      desc [dsc$w_length] / %UPVAL ) + 1);
;  899          1025  3
;  900          1026  3
;  901          1027  3         ! Take characters up to a dot,      olon, or blank
;  902          1028  3         !
;  903          1029  3         name_buf [0] = 0;
;  904          1030  3         i = 1;
;  905          1031  3
;  906          1032  3         WHILE .char NEQ dbg$k_dot
;  907          1033  3                 AND
;  908          1034  3               .char NEQ dbg$k_semicolon
;  909          1035  3         DO
;  910          1036  4             BEGIN
;  911          1037  4                 name_buf [0] = .i;
;  912          1038  4                 name_buf [.i] = .char;
;  913          1039  4                 i = .i + 1;
;  914          1040  4
```

```
;  915          1041  4                ! Check for left paren or blank. This could be the case
;  916          1042  4                ! @F00(param1,param2,....)   or  @F00 param1, param2, ...
;  917          1043  4                !
;  918          1044  4                lookahead_char();
;  919          1045  4                IF .char EQL dbg$k_left_parenthesis OR .char EQL dbg$k_blank
;  920          1046  4                THEN
;  921          1047  4                    EXITLOOP;
;  922          1048  4
;  923          1049  4                next_char ();
;  924          1050  3            END;
;  925          1051  3
;  926          1052  3        RETURN name_buf [0]
;  927          1053  3
;  928          1054  2        END;            ! End of filename


                                001C 00000 FILENAME:
                                                .WORD   Save R2,R3,R4                   ; 1008
                        54 00000000'    EF  9E 00002            MOVAB   CHAR, R4        ;
                        50        10    B4  3C 00009            MOVZWL  @CMD_DESC, R0   ; 1024
                        50              04  C6 0000D            DIVL2   #4, R0          ;
                                  01    A0  9F 00010            PUSHAB  1(R0)           ;
                00000000G   00          01  FB 00013            CALLS   #1, DBG$GET_TEMPMEM     ;
                        52              50  D0 0001A            MOVL    R0, NAME_BUF    ;
                                        62  94 0001D            CLRB    (NAME_BUF)      ; 1029
                        53              01  D0 0001F            MOVL    #1, I           ; 1030
                        50              64  9A 00022 1$:        MOVZBL  CHAR, R0        ; 1032
                        2E              50  91 00025            CMPB    R0, #46         ;
                                        22  13 00028            BEQL    2$              ;
                        3B              50  91 0002A            CMPB    R0, #59         ; 1034
                                        1D  13 0002D            BEQL    2$              ;
                        62              53  90 0002F            MOVB    I, (NAME_BUF)   ; 1037
                      8342              50  90 00032            MOVB    R0, (I)+[NAME_BUF]      ; 1038
                FF7C    CF              00  FB 00036            CALLS   #0, LOOKAHEAD_CHAR      ; 1044
                        28              64  91 0003B            CMPB    CHAR, #40       ; 1045
                                        0C  13 0003E            BEQL    2$              ;
                        20              64  91 00040            CMPB    CHAR, #32       ;
                                        07  13 00043            BEQL    2$              ;
                FF15    CF              00  FB 00045            CALLS   #0, NEXT_CHAR   ; 1049
                                        D6  11 0004A            BRB     1$              ; 1032
                        50              52  D0 0004C 2$:        MOVL    NAME_BUF, R0    ; 1052
                                            04 00 4F            RET                     ; 1054

; Routine Size:  80 bytes,    Routine Base:  DBG$CODE + 02FE

;  929          1055  2
;  930          1056  2    !
;  931          1057  2    !
;  932          1058  2    !
;  933          1059  2    !
;  934          1060  2    !
;  935          1061  2
;  936          1062  2    ROUTINE FILETYPE =
```

```
;  937          1063  2
;  938          1064  2      !++
;  939          1065  2      !   This routine collects the filespec file type string. The file type
;  940          1066  2      !   consists of all characters between '.' and ';'.
;  941          1067  2      !--
;  942          1068  2
;  943          1069  3          BEGIN
;  944          1070  3
;  945          1071  3          LOCAL
;  946          1072  3              TYPE_BUF : REF VECTOR [,BYTE],      ! Buffer for file type
;  947          1073  3              I;                                  ! Counter
;  948          1074  3
;  949          1075  3          ! The file name cannot be longer than the command buffer. Get storage.
;  950          1076  3          !
;  951          1077  3          type_buf = dbg$get_tempmem (( .cmd_desc [dsc$w_length] / %UPVAL) + 1);
;  952          1078  3
;  953          1079  3
;  954          1080  3          ! Take chars up to a semicolon, left paren, or blank.
;  955          1081  3          !
;  956          1082  3          type_buf [0] = 0;
;  957          1083  3          i = 1;
;  958          1084  3
;  959          1085  3          WHILE .char NEQ dbg$k_semicolon
;  960          1086  3          DO
;  961          1087  4              BEGIN
;  962          1088  4                  type_buf [0] = .i;
;  963          1089  4                  type_buf [.i] = .char;
;  964          1090  4                  i = .i + 1;
;  965          1091  4
;  966          1092  4                  ! Check for left paren or blank. This could be
;  967          1093  4                  ! @FOO.COM(param1,param2...) or @FOO.COM param1, param2, ...
;  968          1094  4                  !
;  969          1095  4                  lookahead_char();
;  970          1096  4                  IF .char EQL dbg$k_left_parenthesis OR .char EQL dbg$k_blank
;  971          1097  4                  THEN
;  972          1098  4                      EXITLOOP;
;  973          1099  4
;  974          1100  4                  next_char ();
;  975          1101  3              END;
;  976          1102  3
;  977          1103  3          RETURN type_buf [0];
;  978          1104  3
;  979          1105  2          END;              ! End of filetype


                                 001C 00000 FILETYPE:
                                                .WORD   Save R2,R3,R4                           ; 1062
                    54 00000000'  EF  9E 00002  MOVAB   CHAR, R4                                ;
                    50        10  B4  3C 00009  MOVZWL  @CMD_DESC, R0                         ; 1077
                    50            04  C6 0000D  DIVL2   #4, R0                                ;
                              01  A0  9F 00010  PUSHAB  1(R0)                                  ;
          00000000G 00            01  FB 00013  CALLS   #1, DBG$GET_TEMPMEM                   ;
                    52            50  D0 0001A  MOVL    R0, TYPE_BUF                          ;
                                  62  94 0001D  CLRB    (TYPE_BUF)                            ; 1082
```

```
                     53        01  D0 0001F          MOVL    #1, I                                   ; 1083
                     3B        64  91 00022 1$:      CMPB    CHAR, #59                               ; 1085
                               1D  13 00025          BEQL    2$                                      ;
                     62        53  90 00027          MOVB    I, (TYPE_BUF)                           ; 1088
                   8342        64  90 0002A          MOVB    CHAR, (I)+[TYPE_BUF]                    ; 1089
               FF34    CF      00  FB 0002E          CALLS   #0, LOOKAHEAD_CHAR                      ; 1095
                     28        64  91 00033          CMPB    CHAR, #40                               ; 1096
                               0C  13 00036          BEQL    2$                                      ;
                     20        64  91 00038          CMPB    CHAR, #32                               ;
                               07  13 0003B          BEQL    2$                                      ;
               FECD    CF      00  FB 0003D          CALLS   #0, NEXT_CHAR                           ; 1100
                               DE  11 00042          BRB     1$                                      ; 1085
                     50        52  D0 00044 2$:      MOVL    TYPE_BUF, R0                            ; 1103
                                   04 00047          RET                                             ; 1105

; Routine Size:  72 bytes,    Routine Base:  DBG$CODE + 034E

;  980          1106  2
;  981          1107  2      !
;  982          1108  2      !
;  983          1109  2      !
;  984          1110  2      !
;  985          1111  2      !
;  986          1112  2
;  987          1113  2      ROUTINE VERSION_NUMBER =
;  988          1114  2
;  989          1115  2      !++
;  990          1116  2      !   This routine collects a filespec version number string. That is, all numeric
;  991          1117  2      !   characters following a ';' are taken to be the version number characters.
;  992          1118  2      !--
;  993          1119  2
;  994          1120  3          BEGIN
;  995          1121  3
;  996          1122  3          LOCAL
;  997          1123  3              VERSION_BUF : REF VECTOR [,BYTE],   ! Holds the version string
;  998          1124  3              I,                                  ! Counter
;  999          1125  3              FLAG;                               ! Indicates end of input
; 1000          1126  3
; 1001          1127  3          ! Allocate storage to hold the string.
; 1002          1128  3          ! The version number can be no longer than the save input buffer ( the rest
; 1003          1129  3          ! of the parse line buffer), plus one for the count and one for the
; 1004          1130  3          ! semicolon.
; 1005          1131  3          !
; 1006          1132  3          version_buf = dbg$get_tempmem
; 1007          1133  3                        ((.save_input_desc [dsc$w_length]+5) / %UPVAL);
; 1008          1134  3
; 1009          1135  3
; 1010          1136  3          ! The first character will always be a semicolon. Store this character.
; 1011          1137  3          !
; 1012          1138  3          version_buf [1] = .char;
; 1013          1139  3          version_buf [0] = 1;
; 1014          1140  3
; 1015          1141  3
; 1016          1142  3          ! Aquire the version number chars. Take characters as long as they are alphanumeric
; 1017          1143  3          !
; 1018          1144  3          i = 2;
```

```
;  1019          1145  3          flag = false;
;  1020          1146  3          next_char ();
;  1021          1147  3
;  1022          1148  3          WHILE .char GEQ '0'
;  1023          1149  3                AND
;  1024          1150  3              .char LEQ '9'
;  1025          1151  3          DO
;  1026          1152  4              BEGIN
;  1027          1153  4                  version_buf [0] = .i;
;  1028          1154  4                  version_buf [.i] = .char;
;  1029          1155  4                  i = .i + 1;
;  1030          1156  4
;  1031          1157  4
;  1032          1158  4                  ! Check for exhausted input
;  1033          1159  4                  !
;  1034          1160  4                  IF NOT next_char ()
;  1035          1161  4                  THEN
;  1036          1162  5                      BEGIN
;  1037          1163  5                          flag = true;
;  1038          1164  5                          EXITLOOP;
;  1039          1165  4                      END;
;  1040          1166  4
;  1041          1167  3              END;
;  1042          1168  3
;  1043          1169  3          ! If no numerics were read, a version number was not present. In
;  1044          1170  3          ! that case, remove the semicolon from the buffer.
;  1045          1171  3          !
;  1046          1172  3          IF .version_buf[0] EQL 1
;  1047          1173  3          THEN
;  1048          1174  3              version_buf[0] = 0;
;  1049          1175  3
;  1050          1176  3          ! Return the last character to the input buffer, if it is not a semicolon
;  1051          1177  3          !
;  1052          1178  3          IF NOT .flag
;  1053          1179  3                AND
;  1054          1180  3              .char NEQ dbg$k_semicolon
;  1055          1181  3          THEN
;  1056          1182  4              BEGIN
;  1057          1183  4                  save_input_desc [dsc$w_length] = .save_input_desc [dsc$w_length] + 1;
;  1058          1184  4                  save_input_desc [dsc$a_pointer] = .save_input_desc [dsc$a_pointer] - 1;
;  1059          1185  4                  (.save_input_desc [dsc$a_pointer]) <0, 8, 0> = .char;
;  1060          1186  3              END;
;  1061          1187  3
;  1062          1188  3          RETURN version_buf [0];
;  1063          1189  3
;  1064          1190  2          END;              ! End of version_number


                                 003C 00000 VERSION_NUMBER:
                                                    .WORD   Save R2,R3,R4,R5                        ; 1113
                      55 00000000'   EF  9E 00002   MOVAB   CHAR, R5                                ;
                      50         F4  B5  3C 00009   MOVZWL  @SAVE_INPUT_DESC, R0                    ; 1133
                      50             05  C0 0000D   ADDL2   #5, R0                                 ;
                   7E 50             04  C7 00010   DIVL3   #4, R0, -(SP)                          ;
```

```
                00000000G  00          01 FB 00014          CALLS   #1, DBG$GET_TEMPMEM                      ;
                           52          50 D0 0001B          MOVL    R0, VERSION_BUF                         ;
                        01 A2          65 90 0001E          MOVB    CHAR, 1(VERSION_BUF)                    ; 1138
                           62          01 90 00022          MOVB    #1, (VERSION_BUF)                       ; 1139
                           53          02 7D 00025          MOVQ    #2, I                                   ; 1144
                      FE9A CF          00 FB 00028          CALLS   #0, NEXT_CHAR                           ; 1146
                           50          65 9A 0002D 1$:      MOVZBL  CHAR, R0                                ; 1148
                           30          50 91 00030          CMPB    R0, #48                                 ;
                                       17 1F 00033          BLSSU   2$                                      ;
                           39          50 91 00035          CMPB    R0, #57                                 ; 1150
                                       12 1A 00038          BGTRU   2$                                      ;
                           62          53 90 0003A          MOVB    I, (VERSION_BUF)                        ; 1153
                         8342          50 90 0003D          MOVB    R0, (I)+[VERSION_BUF]                   ; 1154
                      FE81 CF          00 FB 00041          CALLS   #0, NEXT_CHAR                           ; 1160
                           E4          50 E8 00046          BLBS    R0, 1$                                  ;
                           54          01 D0 00049          MOVL    #1, FLAG                                ; 1163
                           01          62 91 0004C 2$:      CMPB    (VERSION_BUF), #1                       ; 1172
                                       02 12 0004F          BNEQ    3$                                      ;
                                       62 94 00051          CLRB    (VERSION_BUF)                           ; 1174
                           12          54 E8 00053 3$:      BLBS    FLAG, 4$                                ; 1178
                           3B          65 91 00056          CMPB    CHAR, #59                               ; 1180
                                       0D 13 00059          BEQL    4$                                      ;
                           50      F4  A5 D0 0005B          MOVL    SAVE_INPUT_DESC, R0                     ; 1183
                                       60 B6 0005F          INCW    (R0)                                    ;
                               04      A0 D7 00061          DECL    4(R0)                                   ; 1184
                        04 B0          65 90 00064          MOVB    CHAR, @4(R0)                            ; 1185
                           50          52 D0 00068 4$:      MOVL    VERSION_BUF, R0                         ; 1188
                                          04 0006B          RET                                             ; 1190

; Routine Size:  108 bytes,    Routine Base:  DBG$CODE + 0396

;  1065          1191  2
;  1066          1192  2     !
;  1067          1193  2     !
;  1068          1194  2     !
;  1069          1195  2     !
;  1070          1196  2     !
;  1071          1197  2
;  1072          1198  2     ROUTINE QUOTED_FILESPEC =
;  1073          1199  2
;  1074          1200  2     !++
;  1075          1201  2     !   This routine collects a quoted filespec string. That is, all characters
;  1076          1202  2     !   coming between ' and ' or " and " are taken to be filespec string characters.
;  1077          1203  2     !   If a termnal ' or " is not encountered, an error message is produced.
;  1078          1204  2     !--
;  1079          1205  2
;  1080          1206  3         BEGIN
;  1081          1207  3
;  1082          1208  3         LOCAL
;  1083          1209  3             I,                                      ! Counter
;  1084          1210  3             QUOTE_CHAR          : BYTE,             ! Holds ' or " for error message
;  1085          1211  3             TEMP_FILESPEC_BUF   : REF VECTOR [,BYTE], ! TEMP buffer for filespec
;  1086          1212  3             FILESPEC_BUF        : REF VECTOR [,BYTE]; ! Buffer for spec string
;  1087          1213  3
;  1088          1214  3         ! The first non-blank character must be a quote or we report failure.
;  1089          1215  3         !
```

```
;  1090          1216  3          IF .char NEQ dbg$k_quote
;  1091          1217  3                AND
;  1092          1218  3             .char NEQ dbg$k_dblquote
;  1093          1219  3          THEN
;  1094          1220  3              RETURN sts$k_error;
;  1095          1221  3
;  1096          1222  3          quote_char = .char;
;  1097          1223  3
;  1098          1224  3
;  1099          1225  3          ! We must allocate non-listed storage to contain the quoted filespec
;  1100          1226  3          ! since we don't want it to disappear at the end of command clean-up.
;  1101          1227  3          ! First we must allocate listed storage to hold the filespec while we
;  1102          1228  3          ! get the characters since the maximum possible length of the buffer
;  1103          1229  3          ! is the length of the command buffer + the length of the input buffer.
;  1104          1230  3          ! Allocating a non-listed buffer of this size would be a waste.
;  1105          1231  3          !
;  1106          1232  3          temp_filespec_buf = dbg$get_tempmem (((.cmd_desc [dsc$w_length] +
;  1107          1233  3                  .save_input_desc [dsc$w_length]) / %UPVAL) + 1);
;  1108          1234  3          temp_filespec_buf [0] = 0;
;  1109          1235  3          next_char ();
;  1110          1236  3
;  1111          1237  3
;  1112          1238  3          ! Get characters until encountering a second quote. If no second quote
;  1113          1239  3          ! is found, produce an error message.
;  1114          1240  3          !
;  1115          1241  3          i = 1;
;  1116          1242  3
;  1117          1243  3          WHILE .char NEQ dbg$k_quote
;  1118          1244  3                AND
;  1119          1245  3              .char NEQ dbg$k_dblquote
;  1120          1246  3          DO
;  1121          1247  4              BEGIN
;  1122          1248  4                  temp_filespec_buf [0] = .i;
;  1123          1249  4                  temp_filespec_buf [.i] = .char;
;  1124          1250  4                  i = .i + 1;
;  1125          1251  4
;  1126          1252  4                  IF NOT next_char ()
;  1127          1253  4                  THEN
;  1128          1254  5                      BEGIN       ! No terminating quote mark - error
;  1129          1255  5
;  1130          1256  5                          ! Don't print the last char which may be a spurious <cr> or semicolon
;  1131          1257  5                          !
;  1132          1258  5                          temp_filespec_buf [0] =
;  1133          1259  6                                  ( IF .temp_filespec_buf [0] GTR 0
;  1134          1260  6                                    THEN
;  1135          1261  6                                          .temp_filespec_buf [0] - 1
;  1136          1262  6                                    ELSE
;  1137          1263  5                                          0);
;  1138          1264  5
;  1139          1265  5                          error_stg_desc [dsc$a_pointer] = temp_filespec_buf [1];
;  1140          1266  5                          error_stg_desc [dsc$w_length] = .temp_filespec_buf [0];
;  1141          1267  5
;  1142          1268  5                          .error_vector = dbg$nmake_arg_vect (dbg$_noend, 3, error_stg_desc,
;  1143          1269  6                                                              1, (IF .quote_char EQL dbg$k_quote
;  1144          1270  6                                                                  THEN one_quote
;  1145          1271  5                                                                  ELSE two_quote));
;  1146          1272  5
```

```
;  1147          1273  5                     RETURN sts$k_severe;
;  1148          1274  5
;  1149          1275  4                 END;
;  1150          1276  3             END;
;  1151          1277  3
;  1152          1278  3
;  1153          1279  3             ! Now allocate the semi-permanant dynamic buffer and copy the chars
;  1154          1280  3             ! from the temporary buffer.
;  1155          1281  3             !
;  1156          1282  3
;  1157          1283  3             filespec_buf = dbg$get_memory((.temp_filespec_buf [0] / %UPVAL) + 1);
;  1158          1284  3             filespec_buf [0] = .temp_filespec_buf [0];
;  1159          1285  3             ch$move (.filespec_buf [0], temp_filespec_buf [1], filespec_buf [1]);
;  1160          1286  3
;  1161          1287  3             RETURN filespec_buf [0];
;  1162          1288  3
;  1163          1289  2             END;            ! End of quoted_filespec


                                00FC 00000 QUOTED_FILESPEC:
                                           .WORD   Save R2,R3,R4,R5,R6,R7                  ;  1198
                57 00000000'    EF  9E 00002       MOVAB   CHAR, R7                        ;
                50              67  9A 00009       MOVZBL  CHAR, R0                        ;  1216
                27              50  91 0000C       CMPB    R0, #39                         ;
                                09  13 0000F       BEQL    1$                              ;
                22              50  91 00011       CMPB    R0, #34                         ;  1218
                                04  13 00014       BEQL    1$                              ;
                50              02  D0 00016       MOVL    #2, R0                          ;  1220
                                04  00019         RET                                     ;
                54              50  90 0001A 1$:    MOVB    R0, QUOTE_CHAR                  ;  1222
                50          10  B7  3C 0001D       MOVZWL  @CMD_DESC, R0                   ;  1233
                51          F4  B7  3C 00021       MOVZWL  @SAVE_INPUT_DESC, R1            ;
                50              51  C0 00025       ADDL2   R1, R0                          ;
                50              04  C6 00028       DIVL2   #4, R0                          ;
                            01  A0  9F 0002B       PUSHAB  1(R0)                           ;
        00000000G   00          01  FB 0002E       CALLS   #1, DBG$GET_TEMPMEM             ;
                52              50  D0 00035       MOVL    R0, TEMP_FILESPEC_BUF           ;
                                62  94 00038       CLRB    (TEMP_FILESPEC_BUF)             ;  1234
            FE1C    CF          00  FB 0003A       CALLS   #0, NEXT_CHAR                   ;  1235
                53              01  D0 0003F       MOVL    #1, I                           ;  1241
                50              67  9A 00042 2$:    MOVZBL  CHAR, R0                        ;  1243
                27              50  91 00045       CMPB    R0, #39                         ;
                                60  13 00048       BEQL    7$                              ;
                22              50  91 0004A       CMPB    R0, #34                         ;  1245
                                5B  13 0004D       BEQL    7$                              ;
                62              53  90 0004F       MOVB    I, (TEMP_FILESPEC_BUF)          ;  1248
              8342              50  90 00052       MOVB    R0, (I)+[TEMP_FILESPEC_BUF]     ;  1249
            FE00    CF          00  FB 00056       CALLS   #0, NEXT_CHAR                   ;  1252
                E4              50  E8 0005B       BLBS    R0, 2$                          ;
                                62  95 0005E       TSTB    (TEMP_FILESPEC_BUF)             ;  1259
                                07  13 00060       BEQL    3$                              ;
                50              62  9A 00062       MOVZBL  (TEMP_FILESPEC_BUF), R0         ;  1261
                                50  D7 00065       DECL    R0                              ;
                                02  11 00067       BRB     4$                              ;
```

```
                                                50  D4 00069 3$:      CLRL    R0                                      ; 1259
                                             62 50  90 0006B 4$:      MOVB    R0, (TEMP_FILESPEC_BUF)                 ;
                                          08 A7     01   A2 9E 0006E         MOVAB   1(R2), ERROR_STG_DESC+4         ; 1265
                                          04 A7          62 9B 00073         MOVZBW  (TEMP_FILESPEC_BUF), ERROR_STG_DESC ; 1266
                                             27          54 91 00077         CMPB    QUOTE_CHAR, #39                 ; 1269
                                                         09 12 0007A         BNEQ    5$                              ;
                                             50 00000000' EF 9E 0007C         MOVAB   ONE_QUOTE, R0                   ;
                                                         07 11 00083         BRB     6$                              ;
                                             50 00000000' EF 9E 00085 5$:     MOVAB   TWO_QUOTE, R0                   ;
                                                         50 DD 0008C 6$:     PUSHL   R0                              ;
                                                         01 DD 0008E         PUSHL   #1                              ; 1268
                                                      04 A7 9F 00090         PUSHAB  ERROR_STG_DESC                  ;
                                                         03 DD 00093         PUSHL   #3                              ;
                                                000281D0    8F DD 00095         PUSHL   #164304                         ;
                                          00000000G      00 05 FB 0009B         CALLS   #5, DBG$NMAKE_ARG_VECT          ;
                                                FC       B7 50 D0 000A2         MOVL    R0, @ERROR_VECTOR               ;
                                                         50 04 D0 000A6         MOVL    #4, R0                          ; 1273
                                                            04 000A9         RET                                     ;
                                                         50 62 9A 000AA 7$:     MOVZBL  (TEMP_FILESPEC_BUF), R0         ; 1283
                                                         50 04 C6 000AD         DIVL2   #4, R0                          ;
                                                      01 A0 9F 000B0         PUSHAB  1(R0)                           ;
                                          00000000G      00 01 FB 000B3         CALLS   #1, DBG$GET_MEMORY              ;
                                                         56 50 D0 000BA         MOVL    R0, FILESPEC_BUF                ;
                                                         66 62 90 000BD         MOVB    (TEMP_FILESPEC_BUF), (FILESPEC_BUF) ; 1284
                                                         50 66 9A 000C0         MOVZBL  (FILESPEC_BUF), R0              ; 1285
                                    01 A6     01   A2    50 28 000C3         MOVC3   R0, 1(TEMP_FILESPEC_BUF), 1(FILESPEC_BUF)
                                                         50 56 D0 000C9         MOVL    FILESPEC_BUF, R0                ; 1287
                                                            04 000CC         RET                                     ; 1289

; Routine Size:  205 bytes,    Routine Base:  DBG$CODE + 0402
```

```
;  1164          1290  2
;  1165          1291  2
;  1166          1292  2
;  1167          1293  2      ! Start of executable code for dbg$nsave_filesp
;  1168          1294  2      !
;  1169          1295  2      cmd_desc = .input_desc;
;  1170          1296  2      error_vector = .message_vect;
;  1171          1297  2
;  1172          1298  2
;  1173          1299  2      ! Obtain the first non-blank character
;  1174          1300  2      !
;  1175          1301  2      next_char ();
;  1176          1302  2      WHILE .char EQL dbg$k_blank
;  1177          1303  2      DO
;  1178          1304  2          next_char ();
;  1179          1305  2
;  1180          1306  2
;  1181          1307  2      ! Check for a quoted file spec
;  1182          1308  2      !
;  1183          1309  2      IF ( quoted_string = quoted_filespec () ) NEQ sts$k_error    ! sts$k_error means
;  1184          1310  2                                                                  ! no quotes
;  1185          1311  2      THEN
;  1186          1312  2
;  1187          1313  2          ! Check for an error
;  1188          1314  2          !
```

```
;  1189          1315  2          IF .quoted_string EQL sts$k_severe
;  1190          1316  2          THEN
;  1191          1317  2              RETURN sts$k_severe
;  1192          1318  2          ELSE
;  1193          1319  3              BEGIN
;  1194          1320  3                  .file = .quoted_string;
;  1195          1321  3                  RETURN sts$k_success;
;  1196          1322  2              END;
;  1197          1323  2
;  1198          1324  2      ! File spec wasn't quoted. Get the file name.
;  1199          1325  2      !
;  1200          1326  2      IF ( name = filename () ) EQL sts$k_severe
;  1201          1327  2      THEN
;  1202          1328  2          RETURN sts$k_severe;
;  1203          1329  2
;  1204          1330  2      ! Get the file type
;  1205          1331  2      !
;  1206          1332  2      IF ( type = filetype () ) EQL sts$k_severe
;  1207          1333  2      THEN
;  1208          1334  2          RETURN sts$k_severe;
;  1209          1335  2
;  1210          1336  2      ! Get the version number
;  1211          1337  2      !
;  1212          1338  2      IF ( version = version_number () ) EQL sts$k_severe
;  1213          1339  2      THEN
;  1214          1340  2          RETURN sts$k_severe;
;  1215          1341  2
;  1216          1342  2      ! Now put the filespec together
;  1217          1343  2      !
;  1218          1344  2      filespec = dbg$get_memory(
;  1219          1345  2          ((.name [0] + .type[0] + .version[0]) / %UPVAL) + 1);
;  1220          1346  2      next_ptr = ch$move (.name [0], name [1], filespec[1]);
;  1221          1347  2      next_ptr = ch$move (.type [0], type [1], .next_ptr);
;  1222          1348  2      ch$move (.version [0], version [1], .next_ptr);
;  1223          1349  2      filespec [0] = .name [0] + .type [0] + .version [0];
;  1224          1350  2
;  1225          1351  2      .file = filespec [0];
;  1226          1352  2
;  1227          1353  2      RETURN sts$k_success;
;  1228          1354  2
;  1229          1355  1      END;                ! End of dbg$nsave_filesp


                                07FC 00000          .ENTRY  DBG$NSAVE_FILESP, Save R2,R3,R4,R5,R6,R7,-      ; 0797
                                                            R8,R9,R10                                           ;
                5A 00000000'    EF   9E 00002       MOVAB   CMD_DESC, R10                                       ;
                6A        04    AC   D0 00009       MOVL    INPUT_DESC, CMD_DESC                                ; 1295
             EC AA        0C    AC   D0 0000D       MOVL    MESSAGE_VECT, ERROR_VECTOR                          ; 1296
          FD77 CF               00   FB 00012 1$:   CALLS   #0, NEXT_CHAR                                       ; 1301
                20        F0    AA   91 00017       CMPB    CHAR, #32                                           ; 1302
```

```
                                 F5  13 0001B          BEQL    1$                                                      ;
                         FF11    CF     00  FB 0001D          CALLS   #0, QUOTED_FILESPEC                             ; 1309
                                 02     50  D1 00022          CMPL    QUOTED_STRING, #2                               ;
                                        0B  13 00025          BEQL    2$                                              ;
                                 04     50  D1 00027          CMPL    QUOTED_STRING, #4                               ; 1315
                                        2D  13 0002A          BEQL    3$                                              ;
                           08    BC     50  D0 0002C          MOVL    QUOTED_STRING, @FILE                            ; 1320
                                        6B  11 00030          BRB     5$                                              ; 1321
                         FDF8    CF     00  FB 00032 2$:      CALLS   #0, FILENAME                                    ; 1327
                                 52     50  D0 00037          MOVL    R0, NAME                                        ;
                                 04     52  D1 0003A          CMPL    NAME, #4                                        ;
                                        1A  13 0003D          BEQL    3$                                              ;
                         FE3B    CF     00  FB 0003F          CALLS   #0, FILETYPE                                    ; 1334
                                 59     50  D0 00044          MOVL    R0, TYPE                                        ;
                                 04     59  D1 00047          CMPL    TYPE, #4                                        ;
                                        0D  13 0004A          BEQL    3$                                              ;
                         FE76    CF     00  FB 0004C          CALLS   #0, VERSION_NUMBER                              ; 1341
                                 58     50  D0 00051          MOVL    R0, VERSION                                     ;
                                 04     58  D1 00054          CMPL    VERSION, #4                                     ;
                                        04  12 00057          BNEQ    4$                                              ;
                                 50     04  D0 00059 3$:      MOVL    #4, R0                                          ; 1343
                                            04 0005C          RET                                                     ;
                                 57     62  9A 0005D 4$:      MOVZBL  (NAME), R7                                      ; 1349
                                 50     69  9A 00060          MOVZBL  (TYPE), R0                                      ;
                                 57     50  C0 00063          ADDL2   R0, R7                                          ;
                                 51     68  9A 00066          MOVZBL  (VERSION), R1                                   ;
                                 57     51  C0 00069          ADDL2   R1, R7                                          ;
                           50    57     04  C7 0006C          DIVL3   #4, R7, R0                                      ;
                                    01  A0  9F 00070          PUSHAB  1(R0)                                           ;
                         00000000G  00     01  FB 00073          CALLS   #1, DBG$GET_MEMORY                              ;
                                 56     50  D0 0007A          MOVL    R0, FILESPEC                                    ;
                                 50     62  9A 0007D          MOVZBL  (NAME), R0                                      ; 1350
                           01    A6     01  A2 00080          MOVC3   R0, 1(NAME), 1(FILESPEC)                        ;
                                 50     69  9A 00086          MOVZBL  (TYPE), R0                                      ; 1351
                           63    01     A9  50 28 00089          MOVC3   R0, 1(TYPE), (NEXT_PTR)                         ;
                                 50     68  9A 0008E          MOVZBL  (VERSION), R0                                   ; 1352
                           63    01     A8  50 28 00091          MOVC3   R0, 1(VERSION), (NEXT_PTR)                      ;
                                 66     57  90 00096          MOVB    R7, (FILESPEC)                                  ; 1353
                                 08     BC  56 D0 00099          MOVL    FILESPEC, @FILE                                 ; 1355
                                 50     01  D0 0009D 5$:      MOVL    #1, R0                                          ; 1357
                                            04 000A0          RET                                                     ; 1359

; Routine Size:  161 bytes,    Routine Base:  DBG$CODE + 04CF

; 1234          1360  1
```

```
;  1236          1361  1 GLOBAL ROUTINE DBG$NVERIFY_OUT (END_VERIFY_POINTER) : NOVALUE =
;  1237          1362  1
;  1238          1363  1 !++
;  1239          1364  1 ! FUNCTIONAL DESCRIPTION:
;  1240          1365  1 !
;  1241          1366  1 !       The function of this routine is to verify commands read from an indirect
;  1242          1367  1 !       command file. That is, the command or comment in question is displayed
;  1243          1368  1 !       at the user's terminal.
;  1244          1369  1 !
;  1245          1370  1 ! FORMAL PARAMETERS:
;  1246          1371  1 !
;  1247          1372  1 !       end_verify_pointer -            pointer to the last character of the input
;  1248          1373  1 !                                       to be verified
;  1249          1374  1 !
;  1250          1375  1 ! IMPLICIT INPUTS:
;  1251          1376  1 !
;  1252          1377  1 !       start_verify_pointer -          pointer to the first character of the input string
;  1253          1378  1 !                                       to be verified
;  1254          1379  1 !
;  1255          1380  1 !       dbg$gb_def_out [out_verify] -   if this byte of dbg$gb_out_verify is set
;  1256          1381  1 !                                       to a non-zero value (1), then a SET OUTPUT VERIFY
;  1257          1382  1 !                                       command is in effect and the VERIFY should
;  1258          1383  1 !                                       take place
;  1259          1384  1 !
;  1260          1385  1 ! IMPLICIT OUTPUTS:
;  1261          1386  1 !
;  1262          1387  1 !       NONE
;  1263          1388  1 !
;  1264          1389  1 ! ROUTINE VALUE:
;  1265          1390  1 !
;  1266          1391  1 !       NOVALUE
;  1267          1392  1 !
;  1268          1393  1 ! COMPLETION CODES:
;  1269          1394  1 !
;  1270          1395  1 !       NONE
;  1271          1396  1 !
;  1272          1397  1 ! SIDE EFFECTS:
;  1273          1398  1 !
;  1274          1399  1 !       The character string lying between the start and end pointers (usually a
;  1275          1400  1 !       single command or comment) is displayed at the user's terminal, if appropriate.
;  1276          1401  1 !
;  1277          1402  1 !--
;  1278          1403  1
;  1279          1404  2     BEGIN
;  1280          1405  2
;  1281          1406  2     LOCAL
;  1282          1407  2         PREV_LINK       : REF cis$link,
;  1283          1408  2         OK_TO_VERIFY;
;  1284          1409  2
;  1285          1410  2     ! Get address of previous link in cis
;  1286          1411  2     !
;  1287          1412  2     prev_link = .dbg$gl_cishead [cis$a_next_link];
;  1288          1413  2
;  1289          1414  2
;  1290          1415  2     ! Check the type of the cis node
;  1291          1416  2     !
;  1292          1417  2     CASE .dbg$gl_cishead [cis$b_input_type] FROM cis_dbg$input TO cis_if
```

```
;  1293          1418  2          OF
;  1294          1419  2          SET
;  1295          1420  2
;  1296          1421  2          [cis_dbg$input] :
;  1297          1422  3              BEGIN
;  1298          1423  3                  ok_to_verify = false;
;  1299          1424  2              END;
;  1300          1425  2
;  1301          1426  2          [cis_rab] :
;  1302          1427  3              BEGIN
;  1303          1428  3                  IF .prev_link [cis$b_input_type] NEQ cis_inpbuf
;  1304          1429  3                  THEN
;  1305          1430  3                      ok_to_verify = true
;  1306          1431  3                  ELSE
;  1307          1432  4                      BEGIN
;  1308          1433  4                          LOCAL
;  1309          1434  4                              pre_prev : REF cis$link;
;  1310          1435  4
;  1311          1436  4                          pre_prev = .prev_link [cis$a_next_link];
;  1312          1437  4                          IF .pre_prev [cis$b_input_type] NEQ cis_dbg$input
;  1313          1438  4                          THEN
;  1314          1439  4                              ok_to_verify = true
;  1315          1440  4                          ELSE
;  1316          1441  4                              ok_to_verify = false;
;  1317          1442  3                      END;
;  1318          1443  2              END;
;  1319          1444  2
;  1320          1445  2          [cis_inpbuf] :
;  1321          1446  3              BEGIN
;  1322          1447  3                  IF .prev_link [cis$b_input_type] EQL cis_dbg$input
;  1323          1448  3                  THEN
;  1324          1449  3                      ok_to_verify = false
;  1325          1450  3                  ELSE
;  1326          1451  3                      ok_to_verify = true;
;  1327          1452  2              END;
;  1328          1453  2
;  1329          1454  2          [cis_acbuf] :
;  1330          1455  3              BEGIN
;  1331          1456  3                  ok_to_verify = true;
;  1332          1457  2              END;
;  1333          1458  2
;  1334          1459  2          [cis_while] :
;  1335          1460  2                  ok_to_verify = true;
;  1336          1461  2
;  1337          1462  2          [cis_repeat] :
;  1338          1463  2                  ok_to_verify = true;
;  1339          1464  2
;  1340          1465  2          [cis_if] :
;  1341          1466  2                  ok_to_verify = true;
;  1342          1467  2
;  1343          1468  2          TES;
;  1344          1469  2
;  1345          1470  2
;  1346          1471  2      ! Delete leading semicolons
;  1347          1472  2      !
;  1348          1473  2      WHILE .(.start_verify_pointer) <0, 8, 0> EQL dbg$k_semicolon
;  1349          1474  2      DO
```

```
; 1350          1475  2         start_verify_pointer = .start_verify_pointer + 1;
; 1351          1476  2
; 1352          1477  2
; 1353          1478  2     ! Now check whether the command should be verified
; 1354          1479  2     !
; 1355          1480  2     IF .dbg$gb_def_out [out_verify]
; 1356          1481  2             AND
; 1357          1482  2        .ok_to_verify
; 1358          1483  2     THEN
; 1359          1484  2         $fao_tt_out (' !AD', .end_verify_pointer - .start_verify_pointer, .start_verify_pointer);
; 1360          1485  2
; 1361          1486  2     RETURN
; 1362          1487  2
; 1363          1488  1     END;                    ! End of dbg$nverify_out


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

                                                04  00002 P.AAC:  .BYTE   4                                       ;
                                    44  41  21  20  00003         .ASCII  \ !AD\                                  ;


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

                                              0004  00000         .ENTRY  DBG$NVERIFY_OUT, Save R2                ; 1361
                          52 00000000'   EF    9E  00002          MOVAB   START_VERIFY_POINTER, R2                ;
                          50 00000000G   00    D0  00009          MOVL    DBG$GL_CISHEAD, R0                      ; 1412
                          51        08   A0    D0  00010          MOVL    8(R0), PREV_LINK                        ;
                 06       00        02   A0    8F  00014          CASEB   2(R0), #0, #6                           ; 1417
  002A         0021     0010      0026             00019 1$:      .WORD   4$-1$,-                                 ;
               002A     002A      002A             00021                  2$-1$,-                                 ;
                                                                          3$-1$,-                                 ;
                                                                          5$-1$,-                                 ;
                                                                          5$-1$,-                                 ;
                                                                          5$-1$,-                                 ;
                                                                          5$-1$                                   ;
                                         16    11  00027          BRB     4$                                      ; 1423
                          02        02   A1    91  00029 2$:      CMPB    2(PREV_LINK), #2                        ; 1428
                                         14    12  0002D          BNEQ    5$                                      ;
                          51        08   A1    D0  0002F          MOVL    8(PREV_LINK), PRE_PREV                  ; 1436
                                    02   A1    95  00033          TSTB    2(PRE_PREV)                             ; 1437
                                         07    13  00036          BEQL    4$                                      ;
                                         09    11  00038          BRB     5$                                      ; 1439
                                    02   A1    95  0003A 3$:      TSTB    2(PREV_LINK)                            ; 1447
                                         04    12  0003D          BNEQ    5$                                      ;
                                         50    D4  0003F 4$:      CLRL    OK_TO_VERIFY                            ; 1449
                                         03    11  00041          BRB     6$                                      ;
                          50             01    D0  00043 5$:      MOVL    #1, OK_TO_VERIFY                        ; 1466
                          3B        00   B2    91  00046 6$:      CMPB    @START_VERIFY_POINTER, #59              ; 1473
                                         04    12  0004A          BNEQ    7$                                      ;
                                         62    D6  0004C          INCL    START_VERIFY_POINTER                    ; 1475
                                         F6    11  0004E          BRB     6$                                      ;
                          17 00000000G   00    E9  00050 7$:      BLBC    DBG$GB_DEF_OUT+2, 8$                    ; 1480
                          14             50    E9  00057          BLBC    OK_TO_VERIFY, 8$                        ; 1482
                                         62    DD  0005A          PUSHL   START_VERIFY_POINTER                    ; 1484
```

```
                     7E      04    AC            62  C3 0005C        SUBL3   START_VERIFY_POINTER, END_VERIFY_POINTER, - ;
                                                                            -(SP)                                       ;
                                        00000000'  EF  9F 00061        PUSHAB  P.AAC                                       ;
                         00000000G  00             03  FB 00067        CALLS   #3, DBG$FAO_OUT                             ;
                                                     04  0006E 8$:     RET                                                 ; 1488

; Routine Size:  111 bytes,    Routine Base:  DBG$CODE + 0570

; 1364          1489  1
```

```
;1366     1490  1 GLOBAL ROUTINE DBG$NCHANGE_TO_NEW : NOVALUE =
;1367     1491  1
;1368     1492  1 !++
;1369     1493  1 ! FUNCTIONAL DESCRIPTION:
;1370     1494  1 !
;1371     1495  1 !       Performs actions associated with switching from old debugger to new
;1372     1496  1 !       debugger. These include initializing data structures, etc.
;1373     1497  1 !
;1374     1498  1 ! FORMAL PARAMETERS:
;1375     1499  1 !
;1376     1500  1 !       NONE
;1377     1501  1 !
;1378     1502  1 ! IMPLICIT INPUTS:
;1379     1503  1 !
;1380     1504  1 !       NONE
;1381     1505  1 !
;1382     1506  1 ! IMPLICIT OUTPUTS:
;1383     1507  1 !
;1384     1508  1 !       dbg$gw_gbllngth - override length
;1385     1509  1 !
;1386     1510  1 !       dbg$gl_gbltyp   - override type
;1387     1511  1 !
;1388     1512  1 !       dbg$gw_dfltleng - default length
;1389     1513  1 !
;1390     1514  1 !       dbg$gl_dflttyp  - default type
;1391     1515  1 !
;1392     1516  1 ! ROUTINE VALUE:
;1393     1517  1 !
;1394     1518  1 !       NONE
;1395     1519  1 !
;1396     1520  1 ! COMPLETION CODES:
;1397     1521  1 !
;1398     1522  1 !       NONE
;1399     1523  1 !
;1400     1524  1 ! SIDE EFFECTS:
;1401     1525  1 !
;1402     1526  1 !       The state of the world is changed to the way the new debugger expects it.
;1403     1527  1 !
;1404     1528  1 !--
;1405     1529  2     BEGIN
;1406     1530  2
;1407     1531  2     ! The old debugger doesn't care about default and override lengths unless
;1408     1532  2     ! the d/o type is asci. However, the new debugger does.
;1409     1533  2     !
;1410     1534  2     dbg$gw_dfltleng =
;1411     1535  3         ( CASE .dbg$gl_dflttyp FROM dsc$k_dtype_bu TO dsc$k_dtype_l
;1412     1536  3             OF
;1413     1537  3             SET
;1414     1538  3
;1415     1539  3             [dsc$k_dtype_bu, dsc$k_dtype_b] : 1;         ! One byte
;1416     1540  3
;1417     1541  3             [dsc$k_dtype_wu, dsc$k_dtype_w] : 2;         ! Two bytes
;1418     1542  3
;1419     1543  3             [dsc$k_dtype_lu, dsc$k_dtype_l] : 4;         ! Four bytes
;1420     1544  3
;1421     1545  3             [INRANGE, OUTRANGE] : .dbg$gw_dfltleng;      ! No change
;1422     1546  3
```

```
;  1423          1547  2              TES);
;  1424          1548  2
;  1425          1549  2             $gl_gbltyp NEQ -1
;  1426          1550  2
;  1427          1551  2            ,$gw_gbllngth =
;  1428          1552  3                 ( CASE .dbg$gl_gbltyp FROM dsc$k_dtype_bu TO dsc$k_dtype_l
;  1429          1553  3                       OF
;  1430          1554  3                       SET
;  1431          1555  3
;  1432          1556  3                       [dsc$k_dtype_bu, dsc$k_dtype_b] : 1;
;  1433          1557  3
;  1434          1558  3                       [dsc$k_dtype_wu, dsc$k_dtype_w] : 2;
;  1435          1559  3
;  1436          1560  3                       [dsc$k_dtype_lu, dsc$k_dtype_l] : 4;
;  1437          1561  3
;  1438          1562  3                       [INRANGE, OUTRANGE] : .dbg$gw_gbllngth;
;  1439          1563  3
;  1440          1564  3                       TES);
;  1441          1565  2
;  1442          1566  2      RETURN;
;  1443          1567  2
;  1444          1568  1      END;        ! End of dbg$nchange_to_new


                                                000C 00000          .ENTRY  DBG$NCHANGE_TO_NEW, Save R2,R3                  ; 1490
                                      53 00000000G   00   9E 00002  MOVAB   DBG$GW_GBLLNGTR, R3                              ;
                                      52 00000000G   00   9E 00009  MOVAB   DBG$GW_DFLTLENG, R2                             ;
                                    06               02   CF 00010  CASEL   DBG$GL_DFLTTYP, #2, #6                        ; 1535
                              000E      001D 0018          0013     00018 1$:  .WORD   3$-1$,-                              ;
                                        001D 0018          0013     00020              4$-1$,-                              ;
                                                                                     5$-1$,-                              ;
                                                                                     2$-1$,-                              ;
                                                                                     3$-1$,-                              ;
                                                                                     4$-1$,-                              ;
                                                                                     5$-1$                                ;
                                      50               62   3C 00026 2$:  MOVZWL  DBG$GW_DFLTLENG, R0                     ; 1545
                                                       0D   11 00029      BRB    6$                                       ;
                                      50               01   D0 0002B 3$:  MOVL   #1, R0                                  ; 1535
                                                       08   11 0002E      BRB    6$                                       ;
                                      50               02   D0 00030 4$:  MOVL   #2, R0                                  ;
                                                       03   11 00033      BRB    6$                                       ;
                                      50               04   D0 00035 5$:  MOVL   #4, R0                                  ;
                                      62               50   B0 00038 6$:  MOVW   R0, DBG$GW_DFLTLENG                      ;
                                      50 00000000G     00   D0 0003B      MOVL   DBG$GL_GBLTYP, R0                         ; 1549
                                FFFFFFFF   8F          50   D1 00042      CMPL   R0, #-1                                   ;
                                                       27   13 00049      BEQL   13$                                       ;
                                    06               02   50 CF 0004B    CASEL  R0, #2, #6                                ; 1552
                              000E      001D 0018          0013     0004F 7$:  .WORD   9$-7$,-                              ;
                                        001D 0018          0013     00057              10$-7$,-                             ;
                                                                                     11$-7$,-                             ;
                                                                                     8$-7$,-                              ;
                                                                                     9$-7$,-                              ;
                                                                                     10$-7$,-                             ;
                                                                                     11$-7$                               ;
```

```
                                50              63  3C 0005D 8$:        MOVZWL  DBG$GW_GBLLNGTH, R0             ; 1562
                                                0D  11 00060            BRB     12$                             ;
                                50              01  D0 00062 9$:        MOVL    #1, R0                          ; 1552
                                                08  11 00065            BRB     12$                             ;
                                50              02  D0 00067 10$:       MOVL    #2, R0                          ;
                                                03  11 0006A            BRB     12$                             ;
                                50              04  D0 0006C 11$:       MOVL    #4, R0                          ;
                                63              50  B0 0006F 12$:       MOVW    R0, DBG$GW_GBLLNGTH             ;
                                                    04 00072 13$:       RET                                     ; 1568

; Routine Size:  115 bytes,    Routine Base:  DBG$CODE + 05DF
```

```
;1446    1569  1 GLOBAL ROUTINE DBG$NSAVE_BREAK_BUFFER(INPUT_DESC, BUFFER) : NOVALUE =
;1447    1570  1 !
;1448    1571  1 ! FUNCTION
;1449    1572  1 !       This routine is essentially like DBG$EXTRACT_STR()
;1450    1573  1 !       except that the bounding characters are "(" and ")" instead
;1451    1574  1 !       of """, and nesting of parentheses is allowed.  This routine
;1452    1575  1 !       is called when the opening parenthesis of a list of breakpoint
;1453    1576  1 !       actions is encountered.  The breakpoint actions are collected
;1454    1577  1 !       but not lexically or semantically scanned.  Storage is reserved
;1455    1578  1 !       for the new string, and a pointer to this storage is returned.
;1456    1579  1 !
;1457    1580  1 !       This routine is also used to collect the action clause for
;1458    1581  1 !       the commands IF, WHILE, and DO. Here, we collect a string as
;1459    1582  1 !       in the processing for SET BREAK DO. Since we may have input after
;1460    1583  1 !       the string is collected, this routine also copies over the rest
;1461    1584  1 !       of the command line from save_input_desc to cmd_stg_desc.
;1462    1585  1 !
;1463    1586  1 !       Another use for this routine is in commands like
;1464    1587  1 !       SET DISPLAY X DO (EXAMINE Y)
;1465    1588  1 !       Here also you can have input after the break buffer, since there
;1466    1589  1 !       may be a comma list of displays.
;1467    1590  1 !
;1468    1591  1 !       A further complication for this routine but not for
;1469    1592  1 !       exact_string, is that we can't just go blindly charging on
;1470    1593  1 !       looking for matching parenthesis.  i.e. we can't get
;1471    1594  1 !       fooled by:
;1472    1595  1 !
;1473    1596  1 !               DBG>SET BREAK x DO (D/AS .=')'; etc)
;1474    1597  1 !
;1475    1598  1 !       We resolve this problem by NOT paying any attention to characters
;1476    1599  1 !       inside quoted strings within the DO action string.
;1477    1600  1 !
;1478    1601  1 ! INPUTS
;1479    1602  1 !       INPUT_DESC - A longword containing the address of an ASCII string
;1480    1603  1 !                    descriptor describing the present input command.
;1481    1604  1 !
;1482    1605  1 !       BUFFER - The address of a longword to contain the address of the
;1483    1606  1 !                stored action buffer. This action buffer is stored as
;1484    1607  1 !                a counted string with a word count at the beginning.
;1485    1608  1 !                (I.e., an ASCIW string).
;1486    1609  1 !
;1487    1610  1 ! OUTPUTS
;1488    1611  1 !       BUFFER  - The address of the saved DEBUG command list action buffer
;1489    1612  1 !                 is returned to the BUFFER longword.
;1490    1613  1 !
;1491    1614  1
;1492    1615  2     BEGIN
;1493    1616  2
;1494    1617  2     MAP
;1495    1618  2         INPUT_DESC: REF DBG$STG_DESC,    ! The input string descriptor
;1496    1619  2         BUFFER: REF VECTOR[1,LONG];      ! Pointer to buffer address return loc.
;1497    1620  2
;1498    1621  2     LOCAL
;1499    1622  2         DELIMITER,                       ! Current delimiter character
;1500    1623  2         ERROR_LENGTH,                    ! Used for error messages
;1501    1624  2         ERROR_PTR,                       ! Used for error messages
;1502    1625  2         NEW_POINTER,                     ! Temporary pointer
```

```
;  1503          1626  2          USE_COUNT,                      ! Number of characters used from input
;  1504          1627  2          PARSE_STG_DESC: REF DBG$STG_DESC,! Parse input string descriptor
;  1505          1628  2          POINTER: REF VECTOR[,WORD],     ! Holds address of dynamic storage for
;  1506          1629  2                                          !      action string when collected
;  1507          1630  2          PAREN_COUNT,                    ! Count of paren levels
;  1508          1631  2          PTR: REF VECTOR[,BYTE],
;  1509          1632  2          CHAR,                           ! Holds a single character
;  1510          1633  2          COUNT,                          ! Character count
;  1511          1634  2          INPUT_PTR,                      ! Current pointer to input string
;  1512          1635  2          IN_STRING,                      ! 0 => we are not currently within an
;  1513          1636  2                                          !      embedded quoted string.  Other-
;  1514          1637  2                                          !      wise we are, and .in_string is
;  1515          1638  2                                          !      the string delimiter (' or ").
;  1516          1639  2          LEN,
;  1517          1640  2          TEMP_PTR: BLOCK[8,BYTE];        ! String descriptor for embedded quote strings
;  1518          1641  2
;  1519          1642  2
;  1520          1643  2
;  1521          1644  2      ! The present input descriptor describes the input command lineup to the first
;  1522          1645  2      ! semicolon in the entire input line. Since we may have semicolons embedded
;  1523          1646  2      ! in a break action sequence, we must construct a buffer which contains the
;  1524          1647  2      ! present command line plus the rest of the input line. The remaining input
;  1525          1648  2      ! line is described by save_input_desc. Later, we must update the save_input_string
;  1526          1649  2      ! to reflect any input that we have used.
;  1527          1650  2      !
;  1528          1651  2      ! Obtain storage for the descriptor.
;  1529          1652  2      !
;  1530          1653  2      PARSE_STG_DESC = DBG$GET_TEMPMEM (2);
;  1531          1654  2
;  1532          1655  2
;  1533          1656  2      ! Allocate a new buffer to hold all the input.
;  1534          1657  2      !
;  1535          1658  4      PARSE_STG_DESC [DSC$A_POINTER] = DBG$GET_TEMPMEM(((.INPUT_DESC [DSC$W_LENGTH] +
;  1536          1659  2              .SAVE_INPUT_DESC [DSC$W_LENGTH] ) / %UPVAL) + 1);
;  1537          1660  2
;  1538          1661  2
;  1539          1662  2      ! Copy the portion of the string from INPUT_DESC into the new descriptor.
;  1540          1663  2      ! One complication is that for C, we want to copy from the original
;  1541          1664  2      ! input buffer, not the upcased one.
;  1542          1665  2      !
;  1543          1666  2      INPUT_PTR = .INPUT_DESC[DSC$A_POINTER];
;  1544          1667  2      IF .DBG$GB_LANGUAGE EQL DBG$K_C
;  1545          1668  2      THEN
;  1546          1669  3          BEGIN
;  1547          1670  3          IF (.INPUT_PTR LSS .DBG$GL_UPCASE_COMMAND_PTR[0]) OR
;  1548          1671  4             (.INPUT_PTR GTR .DBG$GL_UPCASE_COMMAND_PTR[1])
;  1549          1672  3          THEN
;  1550          1673  3              $DBG_ERROR('DBGNCNTRL\DBG$NSAVE_BREAK_BUFFER 10');
;  1551          1674  3
;  1552          1675  3          INPUT_PTR = (.INPUT_PTR - .DBG$GL_UPCASE_COMMAND_PTR[0]) +
;  1553          1676  3                      .DBG$GL_ORIG_COMMAND_PTR;
;  1554          1677  2          END;
;  1555          1678  2      NEW_POINTER = CH$MOVE (.INPUT_DESC [DSC$W_LENGTH], .INPUT_PTR,
;  1556          1679  2                             .PARSE_STG_DESC [DSC$A_POINTER]);
;  1557          1680  2
;  1558          1681  2
;  1559          1682  2      ! There is a <CR> at the end of the input descriptor.  Change this to a
```

```
;  1560          1683  2      ! semicolon.
;  1561          1684  2      !
;  1562          1685  2      CH$WCHAR (';', .NEW_POINTER - 1);
;  1563          1686  2
;  1564          1687  2
;  1565          1688  2      ! Now copy the rest of the input line.
;  1566          1689  2      !
;  1567          1690  2      CH$MOVE (.SAVE_INPUT_DESC [DSC$W_LENGTH], .SAVE_INPUT_DESC [DSC$A_POINTER],
;  1568          1691  2              .NEW_POINTER);
;  1569          1692  2
;  1570          1693  2
;  1571          1694  2      ! Set the count.
;  1572          1695  2      !
;  1573          1696  2      PARSE_STG_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] + .SAVE_INPUT_DESC [DSC$W_LENGTH];
;  1574          1697  2
;  1575          1698  2
;  1576          1699  2      ! Set the variables used for error reporting.
;  1577          1700  2      !
;  1578          1701  2      ERROR_LENGTH = .PARSE_STG_DESC [DSC$W_LENGTH] - 1;
;  1579          1702  2      ERROR_PTR = .PARSE_STG_DESC [DSC$A_POINTER];
;  1580          1703  2
;  1581          1704  2
;  1582          1705  2      ! Do the real work.
;  1583          1706  2      !
;  1584          1707  2      INPUT_PTR = CH$PTR (.PARSE_STG_DESC [DSC$A_POINTER]);
;  1585          1708  2      COUNT = 0;
;  1586          1709  2      IN_STRING = 0;
;  1587          1710  2      TEMP_PTR[DSC$A_POINTER] = 0;
;  1588          1711  2      PAREN_COUNT = 1;
;  1589          1712  2      WHILE TRUE DO
;  1590          1713  3          BEGIN
;  1591          1714  3
;  1592          1715  3
;  1593          1716  3          ! Pick up the next character and see if we
;  1594          1717  3          ! have run off the end of the string.
;  1595          1718  3          !
;  1596          1719  3          CHAR = CH$RCHAR (.INPUT_PTR);
;  1597          1720  3          IF .CHAR EQL 0
;  1598          1721  3          THEN
;  1599          1722  4              BEGIN
;  1600          1723  4
;  1601          1724  4
;  1602          1725  4              ! The string we complain about not begin delimited
;  1603          1726  4              ! is either the supposed break action string, or
;  1604          1727  4              ! a non-terminated embedded quoted string.
;  1605          1728  4              !
;  1606          1729  4              IF .TEMP_PTR[DSC$A_POINTER] NEQ 0
;  1607          1730  4              THEN
;  1608          1731  5                  BEGIN
;  1609          1732  5
;  1610          1733  5
;  1611          1734  5                  ! We didn't find the ending ')' for the break
;  1612          1735  5                  ! action string because an embedded ascii
;  1613          1736  5                  ! string was not properly terminated.
;  1614          1737  5                  !
;  1615          1738  5                  PARSE_STG_DESC[DSC$A_POINTER] = .TEMP_PTR[DSC$A_POINTER];
;  1616          1739  5                  PARSE_STG_DESC[DSC$W_LENGTH] = .TEMP_PTR[DSC$W_LENGTH];
```

```
;  1617          1740  5              DELIMITER = .IN_STRING;
;  1618          1741  5              END
;  1619          1742  5
;  1620          1743  4          ELSE
;  1621          1744  5              BEGIN
;  1622          1745  5
;  1623          1746  5              ! The action string itself was not terminated.
;  1624          1747  5              !
;  1625          1748  5              PARSE_STG_DESC [DSC$W_LENGTH] = .ERROR_LENGTH;
;  1626          1749  5              PARSE_STG_DESC [DSC$A_POINTER] = .ERROR_PTR;
;  1627          1750  5              DELIMITER = %C')';
;  1628          1751  4              END;
;  1629          1752  4
;  1630          1753  4
;  1631          1754  4              ! Truncate the string to 10 characters unless it is already
;  1632          1755  4              ! smaller than that.
;  1633          1756  4              !
;  1634          1757  4              IF .PARSE_STG_DESC[DSC$W_LENGTH] GTR 10
;  1635          1758  4              THEN
;  1636          1759  4                  PARSE_STG_DESC[DSC$W_LENGTH] = 10;
;  1637          1760  4
;  1638          1761  4              SIGNAL(DBG$_NOEND, 3, .PARSE_STG_DESC, 1, DELIMITER);
;  1639          1762  3              END;
;  1640          1763  3
;  1641          1764  3
;  1642          1765  3          ! If we are not already in an embedded quoted string, then this may be
;  1643          1766  3          ! the beginning of one.  If we are, then this may be the end of it.
;  1644          1767  3          !
;  1645          1768  3          IF .CHAR EQL %C'''' OR .CHAR EQL %C'"'
;  1646          1769  3          THEN
;  1647          1770  4          BEGIN
;  1648          1771  4
;  1649          1772  4
;  1650          1773  4          ! IN_STRING tells not only whether or not we are in a quoted
;  1651          1774  4          ! string, but what that string is delimited by.
;  1652          1775  4          !
;  1653          1776  4          IF .IN_STRING EQL 0
;  1654          1777  4          THEN
;  1655          1778  5              BEGIN
;  1656          1779  5
;  1657          1780  5
;  1658          1781  5              ! Now we are within a string.  Save the delimiter so we can
;  1659          1782  5              ! find the end of it.
;  1660          1783  5              !
;  1661          1784  5              IN_STRING = .CHAR;
;  1662          1785  5
;  1663          1786  5
;  1664          1787  5              ! Also save a string descriptor for this string as we may need
;  1665          1788  5              ! it for later error processing.  This string descriptor
;  1666          1789  5              ! includes the supposed delimiting character.
;  1667          1790  5              !
;  1668          1791  5              TEMP_PTR[DSC$A_POINTER] = .INPUT_PTR;
;  1669          1792  5              TEMP_PTR[DSC$W_LENGTH] = .ERROR_LENGTH - .COUNT;
;  1670          1793  5              END
;  1671          1794  5
;  1672          1795  4          ELSE
;  1673          1796  5              BEGIN
```

```
;  1674          1797  5
;  1675          1798  5
;  1676          1799  5                ! See if this quote ends the string we were already in.
;  1677          1800  5                !
;  1678          1801  5                IF .IN_STRING EQL .CHAR
;  1679          1802  5                THEN
;  1680          1803  6                    BEGIN
;  1681          1804  6                    IN_STRING = 0;
;  1682          1805  6                    TEMP_PTR[DSC$A_POINTER] = 0;
;  1683          1806  5                    END;
;  1684          1807  5                !
;  1685          1808  4                END;
;  1686          1809  4                !
;  1687          1810  4            END
;  1688          1811  4            !
;  1689          1812  3        ELSE
;  1690          1813  4            BEGIN
;  1691          1814  4            !
;  1692          1815  4            !
;  1693          1816  4            ! If we are already in an embedded string, and there is no chance
;  1694          1817  4            ! that it is ending, then we don't care what the current character
;  1695          1818  4            ! is.  Otherwise we have to look for parenthesis.
;  1696          1819  4            !
;  1697          1820  4            IF .IN_STRING EQL 0
;  1698          1821  4            THEN
;  1699          1822  5                BEGIN
;  1700          1823  5                !
;  1701          1824  5                !
;  1702          1825  5                ! We are not in an embedded string.  Now we sort out the
;  1703          1826  5                ! parenthesis matching.
;  1704          1827  5                !
;  1705          1828  5                IF .CHAR EQL %C')'
;  1706          1829  5                THEN
;  1707          1830  6                    BEGIN
;  1708          1831  6                    !
;  1709          1832  6                    !
;  1710          1833  6                    ! Found a closing parenthesis. See whether this one matches
;  1711          1834  6                    ! the opening breakpoint action parenthesis, and if it
;  1712          1835  6                    ! does, then exit from this loop, and thus from the macro.
;  1713          1836  6                    !
;  1714          1837  6                    PAREN_COUNT = .PAREN_COUNT - 1;
;  1715          1838  6                    IF .PAREN_COUNT LEQ 0 THEN EXITLOOP;
;  1716          1839  6                    END
;  1717          1840  6                !
;  1718          1841  5                ELSE IF .CHAR EQL %C'('
;  1719          1842  5                THEN
;  1720          1843  5                    PAREN_COUNT = .PAREN_COUNT + 1;
;  1721          1844  5                !
;  1722          1845  4                END;
;  1723          1846  4                !
;  1724          1847  3            END;
;  1725          1848  3            !
;  1726          1849  3            !
;  1727          1850  3        ! Increment the character counter, update the pointer so that we are
;  1728          1851  3        ! looking at the next character, and loop back to do so.
;  1729          1852  3        !
;  1730          1853  3        COUNT = .COUNT + 1;
```

```
;  1731          1854  3          INPUT_PTR = CH$PLUS (.INPUT_PTR, 1);
;  1732          1855  3          PARSE_STG_DESC [DSC$W_LENGTH] = .PARSE_STG_DESC [DSC$W_LENGTH] - 1;
;  1733          1856  3
;  1734          1857  2          END;                                ! End of loop
;  1735          1858  2
;  1736          1859  2
;  1737          1860  2      ! The breakpoint action string has been isolated.  We now allocate a buffer
;  1738          1861  2      ! in dynamic memory for it.  The number of bytes we need to allocate is the
;  1739          1862  2      ! action string size + 3 because we need 2 more bytes to hold the count
;  1740          1863  2      ! and one byte to hold a trailing zero. We then apply the standard formula
;  1741          1864  2      ! num_longwords = (num_bytes+3)/%UPVAL
;                1865  2      !
;  1742          1866  2      POINTER = DBG$GET_MEMORY(((.COUNT + 3) + 3)/%UPVAL);
;  1744          1867  2
;  1745          1868  2
;  1746          1869  2      ! Copy the action string from the input buffer, and then overwrite the
;  1747          1870  2      ! character before the string begins so that is becomes a counted string
;  1748          1871  2      ! action buffer.  We also ensure that there is a 0 character at the end
;  1749          1872  2      ! of the buffer to ensure proper termination of parsing it when the break
;  1750          1873  2      ! happens.
;  1751          1874  2      !
;  1752          1875  2      POINTER[0] = .COUNT + 1;    ! "+ 1" is for trailing zero
;  1753          1876  2      PTR = CH$MOVE(.COUNT, .PARSE_STG_DESC[DSC$A_POINTER], POINTER[1]);
;  1754          1877  2      PTR[0] = 0;                 ! Add trailing zero
;  1755          1878  2
;  1756          1879  2
;  1757          1880  2      ! Now update the parse string descriptor to address the character after
;  1758          1881  2      ! the closing parenthesis.
;  1759          1882  2      !
;  1760          1883  2      PARSE_STG_DESC[DSC$A_POINTER] = CH$PLUS(.INPUT_PTR, 1);
;  1761          1884  2      PARSE_STG_DESC[DSC$W_LENGTH] = .PARSE_STG_DESC[DSC$W_LENGTH] - 1;
;  1762          1885  2
;  1763          1886  2
;  1764          1887  2      ! Now comes the time to figure out what we have and have not eaten so
;  1765          1888  2      ! that the SAVE_INPUT_DESC may be updated properly.  We must also update
;  1766          1889  2      ! the original input descriptor.  Check to see if there is still more
;  1767          1890  2      ! stuff from INPUT_DESC.
;  1768          1891  2      !
;  1769          1892  2      ! If the INPUT_DESC buffer has not been used completely, we set up the
;  1770          1893  2      ! INPUT_DESC string descriptor to point to what remains.
;  1771          1894  2      !
;  1772          1895  2      IF .PARSE_STG_DESC[DSC$A_POINTER] LSSA .NEW_POINTER
;  1773          1896  2      THEN
;  1774          1897  3          BEGIN
;  1775          1898  3          INPUT_DESC[DSC$A_POINTER] = .INPUT_DESC[DSC$A_POINTER] +
;  1776          1899  3                                     .INPUT_DESC[DSC$W_LENGTH] -
;  1777          1900  3                                     (.NEW_POINTER - .PARSE_STG_DESC[DSC$A_POINTER]);
;  1778          1901  3          INPUT_DESC[DSC$W_LENGTH] = .NEW_POINTER - .PARSE_STG_DESC[DSC$A_POINTER];
;  1779          1902  3          END
;  1780          1903  3
;  1781          1904  3
;  1782          1905  3      ! Otherwise, the INPUT_DESC buffer has been exhausted and we are into the
;  1783          1906  3      ! SAVE_INPUT_DESC buffer.  Update it and show exhaustion of the INPUT_DESC
;  1784          1907  3      ! buffer.
;  1785          1908  3      !
;  1786          1909  2      ELSE
;  1787          1910  3          BEGIN
```

```
;  1788          1911  3
;  1789          1912  3
;  1790          1913  3          ! Show INPUT_DESC to be empty and update SAVE_INPUT_DESC.
;  1791          1914  3          !
;  1792          1915  3          INPUT_DESC[DSC$A_POINTER] = 0;
;  1793          1916  3          INPUT_DESC[DSC$W_LENGTH] = 0;
;  1794          1917  3          SAVE_INPUT_DESC[DSC$A_POINTER] = .SAVE_INPUT_DESC[DSC$A_POINTER] +
;  1795          1918  3                          .PARSE_STG_DESC[DSC$A_POINTER] - .NEW_POINTER;
;  1796          1919  3          SAVE_INPUT_DESC [DSC$W_LENGTH] = .SAVE_INPUT_DESC[DSC$W_LENGTH] -
;  1797          1920  3                          (.PARSE_STG_DESC[DSC$A_POINTER] - .NEW_POINTER);
;  1798          1921  3
;  1799          1922  3
;  1800          1923  3          ! We may need to move more of the command buffer into INPUT_DESC.
;  1801          1924  3          ! For example, in "IF TRUE THEN (E X;E Y) ELSE (E Z)" then the
;  1802          1925  3          ! "ELSE (E Z)" now sits in the SAVE_INPUT_DESC buffer and we
;  1803          1926  3          ! must move it into INPUT_DESC so that the parsing of the IF command
;  1804          1927  3          ! can be continued. Similarly with
;  1805          1928  3          ! "SET DISP X DO (E X;E Y), Y DO (E Z)"
;  1806          1929  3          ! We call DBG$NGET_CMD to do this for us.
;  1807          1930  3          ! On the other hand, if a semicolon followed the command buffer, as in
;  1808          1931  3          ! "IF TRUE THEN (EX X;EX Y);E Z"
;  1809          1932  3          ! then we do not want to charge ahead and collect the "E Z".
;  1810          1933  3          ! So we check for semicolon.
;  1811          1934  3          !
;  1812          1935  3          LEN = .SAVE_INPUT_DESC[DSC$W_LENGTH];
;  1813          1936  3          PTR = .SAVE_INPUT_DESC[DSC$A_POINTER];
;  1814          1937  3          WHILE (.PTR[0] EQL DBG$K_BLANK) AND (.LEN NEQ 0) DO
;  1815          1938  4              BEGIN
;  1816          1939  4              PTR = .PTR + 1;
;  1817          1940  4              LEN = .LEN - 1;
;  1818          1941  3              END;
;  1819          1942  4          IF (.PTR[0] NEQ DBG$K_SEMICOLON) AND (.LEN NEQ 0)
;  1820          1943  3          THEN
;  1821          1944  3              DBG$NGET_CMD(.SAVE_INPUT_DESC, .INPUT_DESC, MESSAGE_POINTER, FALSE);
;  1822          1945  3
;  1823          1946  2          END;
;  1824          1947  2
;  1825          1948  2
;  1826          1949  2      ! Return a pointer to the saved-away action buffer.
;  1827          1950  2      !
;  1828          1951  2      BUFFER[0] = .POINTER;
;  1829          1952  1      END;


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

24 47 42 44 5C 4C 52 54 4E 43 4E 47 42 44 23  00007 P.AAD:   .ASCII  \#DBGNCNTRL\<92>\DBG$NSAVE_BREAK_BUFFER \   ;
46 55 42 5F 4B 41 45 52 42 5F 45 56 41 53 4E  00016                                                             ;
                                  20 52 45 46  00025                                                             ;
                                  30 31        00029          .ASCII  \10\                                       ;


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

                                              0FFC 00000        .ENTRY  DBG$NSAVE_BREAK_BUFFER, Save R2,R3,R4,R5,-  ; 1569
```

```
                                                                 R6,R7,R8,R9,R10,R11              :
                   5E             10  C2 00002          SUBL2    #16, SP                          :
                                  02  DD 00005          PUSHL    #2                               : 1653
        00000000G  00             01  FB 00007          CALLS    #1, DBG$GET_TEMPMEM              :
                   59             50  D0 0000E          MOVL     R0, PARSE_STG_DESC               :
                   5A       04    A9  9E 00011          MOVAB    4(PARSE_STG_DESC), R10           : 1658
                   58       04    AC  D0 00015          MOVL     INPUT_DESC, R8                   :
                   50             68  3C 00019          MOVZWL   (R8), R0                         : 1659
                   51 00000000'   FF  3C 0001C          MOVZWL   @SAVE_INPUT_DESC, R1             :
                   50             51  C0 00023          ADDL2    R1, R0                           :
                   50             04  C6 00026          DIVL2    #4, R0                           :
                            01    A0  9F 00029          PUSHAB   1(R0)                            :
        00000000G  00             01  FB 0002C          CALLS    #1, DBG$GET_TEMPMEM              :
                   6A             50  D0 00033          MOVL     R0, (R10)                        :
                   56       04    A8  D0 00036          MOVL     4(R8), INPUT_PTR                 : 1666
                   07 00000000G   00  91 0003A          CMPB     DBG$GB_LANGUAGE, #7              : 1667
                                  37  12 00041          BNEQ     3$                               :
        00000000'  EF             56  D1 00043          CMPL     INPUT_PTR, DBG$GL_UPCASE_COMMAND_PTR   : 1670
                                  09  19 0004A          BLSS     1$                               :
        00000000'  EF             56  D1 0004C          CMPL     INPUT_PTR, DBG$GL_UPCASE_COMMAND_PTR+4 : 1671
                                  15  15 00053          BLEQ     2$                               :
                      00000000'   EF  9F 00055 1$:      PUSHAB   P.AAD                            : 1673
                                  01  DD 0005B          PUSHL    #1                               :
                      00028362    8F  DD 0005D          PUSHL    #164706                          :
        00000000G  00             03  FB 00063          CALLS    #3, LIB$SIGNAL                   :
      50           56 00000000'   EF  C3 0006A 2$:      SUBL3    DBG$GL_UPCASE_COMMAND_PTR, INPUT_PTR, R0 : 1675
      56           50 00000000'   EF  C1 00072          ADDL3    DBG$GL_ORIG_COMMAND_PTR, R0, INPUT_PTR   : 1676
  00  BA           66             68  28 0007A 3$:      MOVC3    (R8), (INPUT_PTR), @0(R10)       : 1679
                   5B             53  D0 0007F          MOVL     R3, NEW_POINTER                  :
            FF     AB             3B  90 00082          MOVB     #59, -1(NEW_POINTER)             : 1685
                   57 00000000'   EF  D0 00086          MOVL     SAVE_INPUT_DESC, R7              : 1690
      6B    04     B7             67  28 0008D          MOVC3    (R7), @4(R7), (NEW_POINTER)      : 1691
      69           68             67  A1 00092          ADDW3    (R7), (R8), (PARSE_STG_DESC)     : 1696
                   57             69  3C 00096          MOVZWL   (PARSE_STG_DESC), ERROR_LENGTH   : 1701
                                  57  D7 00099          DECL     ERROR_LENGTH                     :
                   6E             6A  D0 0009B          MOVL     (R10), ERROR_PTR                 : 1702
                   56             6A  D0 0009E          MOVL     (R10), INPUT_PTR                 : 1707
                                  52  D4 000A1          CLRL     COUNT                            : 1708
                                  54  D4 000A3          CLRL     IN_STRING                        : 1709
                            0C    AE  D4 000A5          CLRL     TEMP_PTR+4                       : 1710
                   55             01  D0 000A8          MOVL     #1, PAREN_COUNT                  : 1711
                   53             66  9A 000AB 4$:      MOVZBL   (INPUT_PTR), CHAR                : 1719
                                  3B  12 000AE          BNEQ     8$                               : 1720
                            0C    AE  D5 000B0          TSTL     TEMP_PTR+4                       : 1729
                                  0E  13 000B3          BEQL     5$                               :
                   6A       0C    AE  D0 000B5          MOVL     TEMP_PTR+4, (R10)                : 1738
                   69       08    AE  B0 000B9          MOVW     TEMP_PTR, (PARSE_STG_DESC)       : 1739
            04     AE             54  D0 000BD          MOVL     IN_STRING, DELIMITER             : 1740
                                  0A  11 000C1          BRB      6$                               : 1729
                   69             57  B0 000C3 5$:      MOVW     ERROR_LENGTH, (PARSE_STG_DESC)   : 1748
                   6A             6E  D0 000C6          MOVL     ERROR_PTR, (R10)                 : 1749
            04     AE             29  D0 000C9          MOVL     #41, DELIMITER                   : 1750
                   0A             69  B1 000CD 6$:      CMPW     (PARSE_STG_DESC), #10            : 1757
                                  03  1B 000D0          BLEQU    7$                               :
                   69             0A  B0 000D2          MOVW     #10, (PARSE_STG_DESC)            : 1759
                            04    AE  9F 000D5 7$:      PUSHAB   DELIMITER                        : 1761
                                  01  DD 000D8          PUSHL    #1                               :
```

```
                                                59  DD 000DA          PUSHL   PARSE_STG_DESC                                  ;
                                                03  DD 000DC          PUSHL   #3                                              ;
                                    00281D0     8F  DD 000DE          PUSHL   #164304                                         ;
                          00000000G   00        05  FB 000E4          CALLS   #5, LIB$SIGNAL                                  ;
                                      27        53  D1 000EB 8$:      CMPL    CHAR, #39                                       ; 1768
                                                05  13 000EE          BEQL    9$                                              ;
                                      22        53  D1 000F0          CMPL    CHAR, #34                                       ;
                                                1E  12 000F3          BNEQ    11$                                             ;
                                                54  D5 000F5 9$:      TSTL    IN_STRING                                       ; 1776
                                                0E  12 000F7          BNEQ    10$                                             ;
                                      54        53  D0 000F9          MOVL    CHAR, IN_STRING                                 ; 1784
                                0C    AE        56  D0 000FC          MOVL    INPUT_PTR, TEMP_PTR+4                           ; 1791
                    08    AE                  52  A3 00100          SUBW3   COUNT, ERROR_LENGTH, TEMP_PTR                   ; 1792
                                                21  11 00105          BRB     13$                                             ; 1776
                                      53        54  D1 00107 10$:     CMPL    IN_STRING, CHAR                                 ; 1801
                                                1C  12 0010A          BNEQ    13$                                             ;
                                                54  D4 0010C          CLRL    IN_STRING                                       ; 1804
                                          0C    AE  D4 0010E          CLRL    TEMP_PTR+4                                      ; 1805
                                                15  11 00111          BRB     13$                                             ; 1768
                                                54  D5 00113 11$:     TSTL    IN_STRING                                       ; 1820
                                                11  12 00115          BNEQ    12$                                             ;
                                      29        53  D1 00117          CMPL    CHAR, #41                                       ; 1828
                                                05  12 0011A          BNEQ    12$                                             ;
                                      09        55  F5 0011C          SOBGTR  PAREN_COUNT, 13$                                ; 1837
                                                10  11 0011F          BRB     14$                                             ; 1838
                                      28        53  D1 00121 12$:     CMPL    CHAR, #40                                       ; 1841
                                                02  12 00124          BNEQ    13$                                             ;
                                                55  D6 00126          INCL    PAREN_COUNT                                     ; 1843
                                                52  D6 00128 13$:     INCL    COUNT                                           ; 1853
                                                56  D6 0012A          INCL    INPUT_PTR                                       ; 1854
                                                69  B7 0012C          DECW    (PARSE_STG_DESC)                                ; 1855
                                            FF7A    31 0012E          BRW     4$                                              ; 1712
                                      50        06  A2 9E 00131 14$:  MOVAB   6(R2), R0                                      ; 1866
                              7E                50  C7 00135          DIVL3   #4, R0, -(SP)                                   ;
                                  00000000G     00  FB 00139          CALLS   #1, DBG$GET_MEMORY                              ;
                                      57        50  D0 00140          MOVL    R0, POINTER                                     ;
                                  67            52  A1 00143          ADDW3   #1, COUNT, (POINTER)                            ; 1875
                    02    A7          00      BA  52 28 00147          MOVC3   COUNT, @0(R10), 2(POINTER)                      ; 1876
                                      52        53  D0 0014D          MOVL    R3, PTR                                         ;
                                                62  94 00150          CLRB    (PTR)                                           ; 1877
                                      6A        01  A6 9E 00152          MOVAB   1(R6), (R10)                                    ; 1883
                                                69  B7 00156          DECW    (PARSE_STG_DESC)                                ; 1884
                                      5B        6A  D1 00158          CMPL    (R10), NEW_POINTER                              ; 1895
                                                15  1E 0015B          BGEQU   15$                                             ;
                                      50        68  3C 0015D          MOVZWL  (R8), R0                                        ; 1899
                                      50        04  A8 C0 00160          ADDL2   4(R8), R0                                       ;
                                  51            6A  5B C3 00164          SUBL3   NEW_POINTER, (R10), R1                          ; 1900
                    04    A8              50      51  C1 00168          ADDL3   R1, R0, 4(R8)                                   ;
                                      68            51  AE 0016D          MNEGW   R1, (R8)                                        ; 1901
                                                4D  11 00170          BRB     18$                                             ; 1895
                                          04    A8  D4 00172 15$:     CLRL    4(R8)                                           ; 1915
                                                68  B4 00175          CLRW    (R8)                                            ; 1916
                                      50 00000000'  EF  D0 00177          MOVL    SAVE_INPUT_DESC, R0                             ; 1917
                                  51        04  A0      6A  C1 0017E          ADDL3   (R10), 4(R0), R1                                ; 1918
                    04    A0              51      5B  C3 00183          SUBL3   NEW_POINTER, R1, 4(R0)                          ;
                                  53            5B      6A  C3 00188          SUBL3   (R10), NEW_POINTER, R3                          ; 1920
                                      60            53  A0 0018C          ADDW2   R3, (R0)                                        ;
```

```
                        51          60  3C 0018F          MOVZWL  (R0), LEN                       ; 1935
                        52    04    A0  D0 00192          MOVL    4(R0), PTR                      ; 1936
                        20          62  91 00196 16$:     CMPB    (PTR), #32                      ; 1937
                                    0A  12 00199          BNEQ    17$                             ;
                                    51  D5 0019B          TSTL    LEN                             ;
                                    06  13 0019D          BEQL    17$                             ;
                                    52  D6 0019F          INCL    PTR                             ; 1939
                                    51  D7 001A1          DECL    LEN                             ; 1940
                                    F1  11 001A3          BRB     16$                             ; 1937
                        3B          62  91 001A5 17$:     CMPB    (PTR), #59                      ; 1942
                                    15  13 001A8          BEQL    18$                             ;
                                    51  D5 001AA          TSTL    LEN                             ;
                                    11  13 001AC          BEQL    18$                             ;
                                    7E  D4 001AE          CLRL    -(SP)                           ; 1944
                00000000'           EF  9F 001B0          PUSHAB  MESSAGE_POINTER                 ;
                    0101            8F  BB 001B6          PUSHR   #^M<R0,R8>                      ;
                F89E    CF          04  FB 001BA          CALLS   #4, DBG$NGET_CMD                ;
                  08    BC          57  D0 001BF 18$:     MOVL    POINTER, @BUFFER                ; 1951
                                    04  001C3           RET                                     ; 1952

; Routine Size:  452 bytes,    Routine Base:  DBG$CODE + 0652
```

```
;1831     1953  1 ROUTINE GET_C_CMD_STRING(INPUT_DESC, CMD_DESC, CIS_DESC, MESSAGE_VECT) =
;1832     1954  1
;1833     1955  1 !++
;1834     1956  1 ! FUNCTIONAL DESCRIPTION:
;1835     1957  1 !
;1836     1958  1 !       This routine gets the first command from the input line.  Also,
;1837     1959  1 !       uppercases the line except for what is in quotes.  This routine
;1838     1960  1 !       takes care of stripping the comments off the end of a DEBUG
;1839     1961  1 !       command. For the language C, the comment characters are '/*'.
;1840     1962  1 !
;1841     1963  1 ! FORMAL PARAMETERS:
;1842     1964  1 !
;1843     1965  1 !       input_desc -            a VAX standard descriptor of the input line
;1844     1966  1 !
;1845     1967  1 !       cmd_desc -              a descriptor that will hold the next command
;1846     1968  1 !                               line.
;1847     1969  1 !       cis_desc -              a descriptor for the current command input
;1848     1970  1 !                               stream.  Just another copy of the above in
;1849     1971  1 !                               case the command is a WHILE-DO.
;1850     1972  1 !       message_vect -          the address of a longword to contain the address
;1851     1973  1 !                               of a message argument vector.
;1852     1974  1 !
;1853     1975  1 ! ROUTINE VALUE:
;1854     1976  1 !
;1855     1977  1 !       A status of the routine.
;1856     1978  1 !
;1857     1979  1 !--
;1858     1980  1
;1859     1981  2     BEGIN
;1860     1982  2
;1861     1983  2     MAP
;1862     1984  2         INPUT_DESC      : REF dbg$stg_desc,     ! Command line
;1863     1985  2         CIS_DESC        : REF CIS$LINK,         ! Current command input stream
;1864     1986  2         CMD_DESC        : REF BLOCK [,BYTE];    ! We don't REF to dbg$stg_desc
;1865     1987  2                                                 ! because of the extra longword
;1866     1988  2                                                 ! for the initial dsc$a_pointer
;1867     1989  2     LOCAL
;1868     1990  2         CHAR_COUNT,
;1869     1991  2         CHAR_STRING     : REF VECTOR [,BYTE],   ! Vector of characters
;1870     1992  2         QUOTE_FLAG,
;1871     1993  2         QUOTE_CHAR;
;1872     1994  2
;1873     1995  2
;1874     1996  2     char_string = .input_desc[dsc$a_pointer];
;1875     1997  2     char_count = 0;
;1876     1998  2
;1877     1999  2     ! Check for a comment line. For C the comment character is '/*',
;1878     2000  2     ! but we also treat a beginning-of-line '!' as a comment line.
;1879     2001  2     ! The reason for this is so that the output of DEBUG log files
;1880     2002  2     ! can still be used as DEBUG input even if language is set to C.
;1881     2003  2     !
;1882     2004  2     IF .char_string [.char_count] EQL '!'
;1883     2005  3     OR (.char_string [.char_count] EQL '/'
;1884     2006  3     AND .char_string [.char_count+1] EQL '*')
;1885     2007  2     THEN
;1886     2008  3         BEGIN
;1887     2009  3         input_desc [dsc$a_pointer] = .input_desc [dsc$a_pointer] +
```

```
;1888        2010  3                                   .input_desc [dsc$w_length];
;1889        2011  3          input_desc [dsc$w_length] = 0;
;1890        2012  3          RETURN sts$k_error;
;1891        2013  2          END;
;1892        2014  2
;1893        2015  2
;1894        2016  2      ! Before proceeding, we fill in the CIS$A_WHILE_CLAUSE field
;1895        2017  2      ! to point to the beginning of the command. This is in case the command
;1896        2018  2      ! is a WHILE; then we are able to iterate by backing up to the
;1897        2019  2      ! beginning of the command.
;1898        2020  2      ! The following code relies on the fact that INPUT_DESC is superimposed
;1899        2021  2      ! on the top link pointed to by DBG$GL_CISHEAD.
;1900        2022  2      !
;1901        2023  2      cis_desc = .input_desc;
;1902        2024  2      cis_desc [cis$a_while_clause] = .input_desc [dsc$a_pointer];
;1903        2025  2      cis_desc [cis$w_while_length] = .input_desc [dsc$w_length];
;1904        2026  2
;1905        2027  2
;1906        2028  2      ! Now count the characters in the command
;1907        2029  2      !
;1908        2030  2      char_string = .input_desc [dsc$a_pointer];
;1909        2031  2      char_count = 0;
;1910        2032  2      quote_flag = false;
;1911        2033  2      WHILE .input_desc [dsc$w_length] GTR 0
;1912        2034  2      DO
;1913        2035  3          BEGIN
;1914        2036  3              IF .char_string [.char_count] EQL dbg$k_car_return
;1915        2037  3                                        OR
;1916        2038  3                 .char_string [.char_count] EQL dbg$k_line_feed
;1917        2039  3                                        OR
;1918        2040  3                 .char_string [.char_count] EQL dbg$k_null
;1919        2041  3                                        OR
;1920        2042  4                 ((NOT .quote_flag) AND .char_string [.char_count] EQL ';')
;1921        2043  3                                        OR
;1922        2044  4                 ((NOT .quote_flag) AND .char_string [.char_count] EQL '/'
;1923        2045  4                                        AND .char_string [.char_count+1] EQL '*')
;1924        2046  3              THEN
;1925        2047  3                  EXITLOOP
;1926        2048  3              ELSE
;1927        2049  4                  BEGIN
;1928        2050  4                  IF .char_string [.char_count] EQL dbg$k_quote
;1929        2051  4                                              OR
;1930        2052  4                     .char_string [.char_count] EQL dbg$k_dblquote
;1931        2053  4                  THEN
;1932        2054  5                      BEGIN
;1933        2055  5                      IF NOT .quote_flag
;1934        2056  5                      THEN
;1935        2057  6                          BEGIN
;1936        2058  6                          quote_char = .char_string [.char_count];
;1937        2059  6                          quote_flag = true;
;1938        2060  6                          END
;1939        2061  5                      ELSE
;1940        2062  6                          BEGIN
;1941        2063  6                          IF .char_string [.char_count] EQL .quote_char
;1942        2064  6                          THEN
;1943        2065  6                              quote_flag = false;
;1944        2066  5                          END;
```

```
;  1945     2067  4                        END;
;  1946     2068  4
;  1947     2069  4                    char_count = .char_count + 1;
;  1948     2070  4                    input_desc [dsc$w_length] = .input_desc [dsc$w_length] - 1;
;  1949     2071  3                END;
;  1950     2072  2            END;
;  1951     2073  2
;  1952     2074  2
;  1953     2075  2        ! Now try to get storage for the command string
;  1954     2076  2        !
;  1955     2077  2        cmd_desc [dsc$a_pointer] = dbg$get_tempmem((.char_count / %UPVAL) + 1);
;  1956     2078  2
;  1957     2079  2
;  1958     2080  2        ! Save away pointers both to the original input string, and to
;  1959     2081  2        ! the copied string in cmd_desc. These are used later as follows:
;  1960     2082  2        ! In the language C, a lower case name represents a distinct object
;  1961     2083  2        ! from its upper-case counterpart. Since we upper-case commands in
;  1962     2084  2        ! cmd_desc, we will need to go back to the original input_desc to
;  1963     2085  2        ! get at the original version of the name. For this, we need these
;  1964     2086  2        ! two pointers.
;  1965     2087  2        !
;  1966     2088  2        ! The pointer to the upcased string is actually a vector containing
;  1967     2089  2        ! pointers to the beginning and the end of the string.
;  1968     2090  2        !
;  1969     2091  2        dbg$gl_orig_command_ptr = .input_desc[dsc$a_pointer];
;  1970     2092  2        dbg$gl_upcase_command_ptr[0] = .cmd_desc[dsc$a_pointer];
;  1971     2093  2        dbg$gl_upcase_command_ptr[1] = .cmd_desc[dsc$a_pointer] + .char_count - 1;
;  1972     2094  2
;  1973     2095  2        ! Fill the command buffer
;  1974     2096  2        !
;  1975     2097  2        ch$move ( .char_count, .input_desc [dsc$a_pointer], .cmd_desc [dsc$a_pointer]);
;  1976     2098  2
;  1977     2099  2
;  1978     2100  2        ! Update the input descriptor pointer. Check for a comment to skip.
;  1979     2101  2        ! The comment character is '/*' in C.
;  1980     2102  2        !
;  1981     2103  2        IF .char_string [.char_count] EQL '/'
;  1982     2104  2           AND .char_string [.char_count+1] EQL '*'
;  1983     2105  2        THEN
;  1984     2106  3            BEGIN
;  1985     2107  3            input_desc [dsc$a_pointer] = .input_desc [dsc$a_pointer] +
;  1986     2108  3                                         .input_desc[dsc$w_length] +
;  1987     2109  3                                         .char_count;
;  1988     2110  3            input_desc [dsc$w_length] = 0;
;  1989     2111  3            END
;  1990     2112  2        ELSE
;  1991     2113  2            input_desc [dsc$a_pointer] = char_string [.char_count];
;  1992     2114  2
;  1993     2115  2
;  1994     2116  2        ! Update the command descriptor
;  1995     2117  2        !
;  1996     2118  2        cmd_desc [initial_ptr] = .cmd_desc [dsc$a_pointer];
;  1997     2119  2        cmd_desc [dsc$w_length] = .char_count;
;  1998     2120  2        char_string = .cmd_desc [dsc$a_pointer];
;  1999     2121  2
;  2000     2122  2
;  2001     2123  2        ! Now check for bad chars and translate to upper case
```

```
;  2002          2124  2      !
;  2003          2125  2      char_count = 0;
;  2004          2126  2      quote_flag = false;
;  2005          2127  2      WHILE .char_count LSS .cmd_desc [dsc$w_length]
;  2006          2128  2      DO
;  2007          2129  3          BEGIN
;  2008          2130  3              IF .char_string [.char_count] EQL dbg$k_tab
;  2009          2131  3              THEN
;  2010          2132  3                  char_string [.char_count] = dbg$k_blank;  ! Convert tab to space
;  2011          2133  3
;  2012          2134  3              IF .char_string [.char_count] LSS dbg$k_blank
;  2013          2135  3              THEN
;  2014          2136  4                  BEGIN
;  2015          2137  4                  .message_vect = dbg$nmake_arg_vect (dbg$_invchar);
;  2016          2138  4                  RETURN sts$k_severe;
;  2017          2139  4                  END
;  2018          2140  3              ELSE
;  2019          2141  4                  BEGIN
;  2020          2142  4                      IF .char_string [.char_count] EQL dbg$k_quote
;  2021          2143  4                                                    OR
;  2022          2144  4                         .char_string [.char_count] EQL dbg$k_dblquote
;  2023          2145  4                      THEN
;  2024          2146  5                          BEGIN
;  2025          2147  5                          IF NOT .quote_flag
;  2026          2148  5                          THEN
;  2027          2149  6                              BEGIN
;  2028          2150  6                              quote_char = .char_string [.char_count];
;  2029          2151  6                              quote_flag = true;
;  2030          2152  6                              END
;  2031          2153  5                          ELSE
;  2032          2154  6                              BEGIN
;  2033          2155  6                              IF .char_string [.char_count] EQL .quote_char
;  2034          2156  6                              THEN
;  2035          2157  6                                  quote_flag = false;
;  2036          2158  5                              END;
;  2037          2159  4                          END;
;  2038          2160  4
;  2039          2161  4                  IF .char_string [.char_count] GEQ 'a'
;  2040          2162  4                                  AND
;  2041          2163  4                     .char_string [.char_count] LEQ 'z'
;  2042          2164  4                                  AND
;  2043          2165  4                     NOT .quote_flag
;  2044          2166  4                  THEN
;  2045          2167  4                      char_string [.char_count] = .char_string [.char_count]
;  2046          2168  4                                                - dbg$k_lcbias;
;  2047          2169  3                  END;
;  2048          2170  3
;  2049          2171  3              char_count = .char_count + 1;
;  2050          2172  3
;  2051          2173  2          END;
;  2052          2174  2
;  2053          2175  2
;  2054          2176  2      ! Termanate the command with a <cr>
;  2055          2177  2      !
;  2056          2178  2      char_string [.char_count] = dbg$k_car_return;
;  2057          2179  2      cmd_desc [dsc$w_length] = .cmd_desc [dsc$w_length] + 1;
;  2058          2180  2
```

```
; 2059          2181  2     RETURN sts$k_success;
; 2060          2182  1     END;
; INFO#250                  L1:2063
; Referenced LOCAL symbol QUOTE_CHAR is probably not initialized


                                    0FFC 00000 GET_C_CMD_STRING:
                                               .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1953
                                    5E      04 C2 00002         SUBL2   #4, SP                        ;
                                    5A   04 AC D0 00005         MOVL    INPUT_DESC, R10               ; 1996
                                    59   04 AA 9E 00009         MOVAB   4(R10), R9                    ;
                                    57      69 D0 0000D         MOVL    (R9), CHAR_STRING             ;
                                            56 D4 00010         CLRL    CHAR_COUNT                    ; 1997
                                    21    6647 91 00012         CMPB    (CHAR_COUNT)[CHAR_STRING], #33 ; 2004
                                            0D 13 00016         BEQL    1$                            ;
                                    2F    6647 91 00018         CMPB    (CHAR_COUNT)[CHAR_STRING], #47 ; 2005
                                            13 12 0001C         BNEQ    2$                            ;
                                    2A 01 A647 91 0001E         CMPB    1(CHAR_COUNT)[CHAR_STRING], #42 ; 2006
                                            0C 12 00023         BNEQ    2$                            ;
                                    50      6A 3C 00025 1$:     MOVZWL  (R10), R0                     ; 2010
                                    69      50 C0 00028         ADDL2   R0, (R9)                      ;
                                            6A B4 0002B         CLRW    (R10)                         ; 2011
                                    50      02 D0 0002D         MOVL    #2, R0                        ; 2012
                                               04 00030         RET                                   ;
                                 0C AC      5A D0 00031 2$:     MOVL    R10, CIS_DESC                 ; 2023
                                    50   0C AC D0 00035         MOVL    CIS_DESC, R0                  ; 2024
                                 14 A0      69 D0 00039         MOVL    (R9), 20(R0)                  ;
                                 34 A0      6A B0 0003D         MOVW    (R10), 52(R0)                 ; 2025
                                    57      69 D0 00041         MOVL    (R9), CHAR_STRING             ; 2030
                                            56 D4 00044         CLRL    CHAR_COUNT                    ; 2031
                                            5B D4 00046         CLRL    QUOTE_FLAG                    ; 2032
                                            6A B5 00048 3$:     TSTW    (R10)                         ; 2033
                                            4B 13 0004A         BEQL    8$                            ;
                                    50    6647 9A 0004C         MOVZBL  (CHAR_COUNT)[CHAR_STRING], R0 ; 2036
                                    0D      50 91 00050         CMPB    R0, #13                       ;
                                            42 13 00053         BEQL    8$                            ;
                                    0A      50 91 00055         CMPB    R0, #10                       ; 2038
                                            3D 13 00058         BEQL    8$                            ;
                                            50 D5 0005A         TSTL    R0                            ; 2040
                                            39 13 0005C         BEQL    8$                            ;
                                    14      5B E8 0005E         BLBS    QUOTE_FLAG, 4$                ; 2042
                                    3B      50 91 00061         CMPB    R0, #59                       ;
                                            31 13 00064         BEQL    8$                            ;
                                    0C      5B E8 00066         BLBS    QUOTE_FLAG, 4$                ; 2044
                                    2F      50 91 00069         CMPB    R0, #47                       ;
                                            07 12 0006C         BNEQ    4$                            ;
                                    2A 01 A647 91 0006E         CMPB    1(CHAR_COUNT)[CHAR_STRING], #42 ; 2045
                                            22 13 00073         BEQL    8$                            ;
                                    27      50 91 00075 4$:     CMPB    R0, #39                       ; 2050
                                            05 13 00078         BEQL    5$                            ;
                                    22      50 91 0007A         CMPB    R0, #34                       ; 2052
                                            12 12 0007D         BNEQ    7$                            ;
                                    08      5B E8 0007F 5$:     BLBS    QUOTE_FLAG, 6$                ; 2055
                                    6E      50 D0 00082         MOVL    R0, QUOTE_CHAR                ; 2058
```

```
                                5B            01  D0 00085          MOVL    #1, QUOTE_FLAG                          ; 2059
                                              07  11 00088          BRB     7$                                      ; 2055
                                6E            50  D1 0008A 6$:      CMPL    R0, QUOTE_CHAR                          ; 2063
                                              02  12 0008D          BNEQ    7$                                      ;
                                              5B  D4 0008F          CLRL    QUOTE_FLAG                              ; 2065
                                              56  D6 00091 7$:      INCL    CHAR_COUNT                              ; 2069
                                              6A  B7 00093          DECW    (R10)                                   ; 2070
                                              B1  11 00095          BRB     3$                                      ; 2033
                                58          08  AC  D0 00097 8$:      MOVL    CMD_DESC, R8                            ; 2077
                           50                   56  C7 0009B          DIVL3   #4, CHAR_COUNT, R0                      ;
                                            01  A0  9F 0009F          PUSHAB  1(R0)                                   ;
                                00000000G   00      01  FB 000A2          CALLS   #1, DBG$GET_TEMPMEM                     ;
                                      04    A8      50  D0 000A9          MOVL    R0, 4(R8)                               ;
                                00000000'   EF      69  D0 000AD          MOVL    (R9), DBG$GL_ORIG_COMMAND_PTR           ; 2091
                                00000000'   EF  04  A8  D0 000B4          MOVL    4(R8), DBG$GL_UPCASE_COMMAND_PTR        ; 2092
                           50                   56  04  A8  C1 000BC          ADDL3   4(R8), CHAR_COUNT, R0           ; 2093
                                00000000'   EF  FF  A0  9E 000C1          MOVAB   -1(R0), DBG$GL_UPCASE_COMMAND_PTR+4     ;
                      04   B8          00      B9      56  28 000C9          MOVC3   CHAR_COUNT, @0(R9), @4(R8)      ; 2097
                                              2F      6647  91 000CF          CMPB    (CHAR_COUNT)[CHAR_STRING], #47  ; 2103
                                                15  12 000D3          BNEQ    9$                                      ;
                                              2A  01  A647  91 000D5          CMPB    1(CHAR_COUNT)[CHAR_STRING], #42 ; 2104
                                                0E  12 000DA          BNEQ    9$                                      ;
                                              50      6A  3C 000DC          MOVZWL  (R10), R0                       ; 2108
                                              50      69  C0 000DF          ADDL2   (R9), R0                        ;
                           69                 50      56  C1 000E2          ADDL3   CHAR_COUNT, R0, (R9)            ; 2109
                                                      6A  B4 000E6          CLRW    (R10)                           ; 2110
                                                      04  11 000E8          BRB     10$                             ; 2103
                           69                 57      56  C1 000EA 9$:      ADDL3   CHAR_COUNT, CHAR_STRING, (R9)   ; 2113
                                    08    A8      04  A8  D0 000EE 10$:     MOVL    4(R8), 8(R8)                    ; 2118
                                          68      56  B0 000F3          MOVW    CHAR_COUNT, (R8)                        ; 2119
                                          57      04  A8  D0 000F6          MOVL    4(R8), CHAR_STRING              ; 2120
                                                      56  D4 000FA          CLRL    CHAR_COUNT                      ; 2125
                                                      5B  D4 000FC          CLRL    QUOTE_FLAG                      ; 2126
                      56                68          10      00  ED 000FE 11$:     CMPZV   #0, #T6, (R8), CHAR_COUNT       ; 2127
                                                      5B  15 00103          BLEQ    18$                             ;
                                                09        6647  91 00105          CMPB    (CHAR_COUNT)[CHAR_STRING], #9   ; 2130
                                                          04  12 00109          BNEQ    12$                             ;
                                                    6647      20  90 0010B          MOVB    #32, (CHAR_COUNT)[CHAR_STRING]  ; 2132
                                                52        6647  9A 0010F 12$:     MOVZBL  (CHAR_COUNT)[CHAR_STRING], R2   ; 2134
                                                20          52  91 00113          CMPB    R2, #32                         ;
                                                            15  1E 00116          BGEQU   13$                             ;
                                                  000281A0      8F  DD 00118          PUSHL   #164256                         ; 2137
                                00000000G   00                  01  FB 0011E          CALLS   #1, DBG$NMAKE_ARG_VECT          ;
                                      10    BC                  50  D0 00125          MOVL    R0, @MESSAGE_VECT               ;
                                            50                  04  D0 00129          MOVL    #4, R0                  ; 2138
                                                                04  0012C          RET                             ;
                                            27                  52  91 0012D 13$:     CMPB    R2, #39                 ; 2142
                                                                05  13 00130          BEQL    14$                     ;
                                            22                  52  91 00132          CMPB    R2, #34                 ; 2144
                                                                12  12 00135          BNEQ    16$                     ;
                                            08                  5B  E8 00137 14$:     BLBS    QUOTE_FLAG, 15$         ; 2147
                                            6E                  52  D0 0013A          MOVL    R2, QUOTE_CHAR          ; 2150
                                            5B                  01  D0 0013D          MOVL    #1, QUOTE_FLAG          ; 2151
                                                                07  11 00140          BRB     16$                     ; 2147
                                            6E                  52  D1 00142 15$:     CMPL    R2, QUOTE_CHAR          ; 2155
                                                                02  12 00145          BNEQ    16$                     ;
                                                                5B  D4 00147          CLRL    QUOTE_FLAG              ; 2157
```

```
                61    8F            52  91 00149 16$:     CMPB    R2, #97                                 ; 2161
                                    0D  1F 0014D           BLSSU   17$                                     ;
                7A    8F            52  91 0014F           CMPB    R2, #122                                ; 2163
                                    07  1A 00153           BGTRU   17$                                     ;
                   04               5B  E8 00155           BLBS    QUOTE_FLAG, 17$                         ; 2165
                 6647               20  82 00158           SUBB2   #32, (CHAR_COUNT)[CHAR_STRING]          ; 2168
                                    56  D6 0015C 17$:       INCL    CHAR_COUNT                              ; 2171
                                    9E  11 0015E           BRB     11$                                     ; 2127
                 6647               0D  90 00160 18$:      MOVB    #13, (CHAR_COUNT)[CHAR_STRING]          ; 2178
                                    68  B6 00164           INCW    (R8)                                    ; 2179
                   50               01  D0 00166           MOVL    #1, R0                                  ; 2181
                                        04 00169           RET                                             ; 2182

; Routine Size:  362 bytes,    Routine Base:  DBG$CODE + 0816
```

```
;  2062          2183  1 ROUTINE GET_ADA_CMD_STRING(INPUT_DESC, CMD_DESC, CIS_DESC, MESSAGE_VECT) =
;  2063          2184  1
;  2064          2185  1 !++
;  2065          2186  1 ! FUNCTIONAL DESCRIPTION:
;  2066          2187  1 !
;  2067          2188  1 !       This routine gets the first command from the input line.  Also,
;  2068          2189  1 !       uppercases the line except for what is in quotes.  This routine
;  2069          2190  1 !       takes care of stripping the comments off the end of a DEBUG
;  2070          2191  1 !       command. For Ada the comment character is '!'. There is also
;  2071          2192  1 !       special code for the Ada tick operator that looks a lot
;  2072          2193  1 !       like a single quote. And picking up a quoted quote is different.
;  2073          2194  1 !       (e.g. ''' is the character single quote.
;  2074          2195  1 !
;  2075          2196  1 ! FORMAL PARAMETERS:
;  2076          2197  1 !
;  2077          2198  1 !       input_desc -            a VAX standard descriptor of the input line
;  2078          2199  1 !
;  2079          2200  1 !       cmd_desc -              a descriptor that will hold the next command
;  2080          2201  1 !                               line.
;  2081          2202  1 !       cis_desc -              a descriptor for the current command input
;  2082          2203  1 !                               stream.  Just another copy of the above in
;  2083          2204  1 !                               case the command is a WHILE-DO.
;  2084          2205  1 !       message_vect -          the address of a longword to contain the address
;  2085          2206  1 !                               of a message argument vector.
;  2086          2207  1 !
;  2087          2208  1 ! ROUTINE VALUE:
;  2088          2209  1 !
;  2089          2210  1 !       A status of the routine.
;  2090          2211  1 !
;  2091          2212  1 !--
;  2092          2213  1
;  2093          2214  2     BEGIN
;  2094          2215  2
;  2095          2216  2     MAP
;  2096          2217  2         INPUT_DESC      : REF dbg$stg_desc,     ! Command line
;  2097          2218  2         CIS_DESC        : REF CIS$LINK,         ! Current command input stream
;  2098          2219  2         CMD_DESC        : REF BLOCK [,BYTE];    ! We don't REF to dbg$stg_desc
;  2099          2220  2                                                 ! because of the extra longword
;  2100          2221  2                                                 ! for the initial dsc$a_pointer
;  2101          2222  2     LOCAL
;  2102          2223  2         CHAR_COUNT,
;  2103          2224  2         CHAR_STRING     : REF VECTOR [,BYTE],   ! Vector of characters
;  2104          2225  2         QUOTE_FLAG,
;  2105          2226  2         QUOTE_CHAR;
;  2106          2227  2
;  2107          2228  2
;  2108          2229  2     char_string = .input_desc[dsc$a_pointer];
;  2109          2230  2     char_count = 0;
;  2110          2231  2
;  2111          2232  2     ! Check for a comment line. For all languages except C, the comment
;  2112          2233  2     ! character is '!'.
;  2113          2234  2     !
;  2114          2235  2     IF .char_string [.char_count] EQL '!'
;  2115          2236  2     THEN
;  2116          2237  3         BEGIN
;  2117          2238  3         input_desc [dsc$a_pointer] = .input_desc [dsc$a_pointer] +
;  2118          2239  3                                      .input_desc [dsc$w_length];
```

```
;  2119          2240  3          input_desc [dsc$w_length] = 0;
;  2120          2241  3          RETURN sts$k_error;
;  2121          2242  2          END;
;  2122          2243  2
;  2123          2244  2
;  2124          2245  2      ! Before proceeding, we fill in the CIS$A_WHILE_CLAUSE field
;  2125          2246  2      ! to point to the beginning of the command. This is in case the command
;  2126          2247  2      ! is a WHILE; then we are able to iterate by backing up to the
;  2127          2248  2      ! beginning of the command.
;  2128          2249  2      ! The following code relies on the fact that INPUT_DESC is superimposed
;  2129          2250  2      ! on the top link pointed to by DBG$GL_CISHEAD.
;  2130          2251  2      !
;  2131          2252  2      cis_desc = .input_desc;
;  2132          2253  2      cis_desc [cis$a_while_clause] = .input_desc [dsc$a_pointer];
;  2133          2254  2      cis_desc [cis$w_while_length] = .input_desc [dsc$w_length];
;  2134          2255  2
;  2135          2256  2
;  2136          2257  2      ! Now count the characters in the command
;  2137          2258  2      !
;  2138          2259  2      char_string = .input_desc [dsc$a_pointer];
;  2139          2260  2      char_count = 0;
;  2140          2261  2      quote_flag = false;
;  2141          2262  2      WHILE .char_count LSS .input_desc [dsc$w_length]
;  2142          2263  2      DO
;  2143          2264  3          BEGIN
;  2144          2265  3              IF .char_string [.char_count] EQL dbg$k_car_return
;  2145          2266  3                                           OR
;  2146          2267  3                 .char_string [.char_count] EQL dbg$k_line_feed
;  2147          2268  3                                           OR
;  2148          2269  3                 .char_string [.char_count] EQL dbg$k_null
;  2149          2270  3                                           OR
;  2150          2271  4                 ((NOT .quote_flag) AND .char_string [.char_count] EQL ';')
;  2151          2272  3                                           OR
;  2152          2273  4                 ((NOT .quote_flag) AND .char_string [.char_count] EQL '!')
;  2153          2274  3              THEN
;  2154          2275  3                  EXITLOOP
;  2155          2276  3              ELSE
;  2156          2277  4                  BEGIN
;  2157          2278  4                      IF .char_string [.char_count] EQL dbg$k_quote
;  2158          2279  4                                                   OR
;  2159          2280  4                         .char_string [.char_count] EQL dbg$k_dblquote
;  2160          2281  4                      THEN
;  2161          2282  5                          BEGIN
;  2162          2283  5                          IF NOT .quote_flag
;  2163          2284  5                          THEN
;  2164          2285  5                              ! Make sure this is not a tick operator.  This is
;  2165          2286  5                              ! nasty stuff... (e.g. '(';') => TICK, but '(';')' => QUOTE)
;  2166          2287  5                              !
;  2167          2288  5                              IF (.char_string [.char_count] EQL dbg$k_quote) AND
;  2168          2289  5                                 (.char_count LEQ .input_desc [dsc$w_length] - 2) AND
;  2169          2290  6                                 (.char_string [.char_count + 2] EQL dbg$k_quote)
;  2170          2291  5                              THEN
;  2171          2292  6                                  BEGIN
;  2172          2293  7                                  IF (.char_string [.char_count + 1] NEQ dbg$k_left_parenthesis)
;  2173          2294  6                                  THEN
;  2174          2295  7                                      BEGIN
;  2175          2296  7                                      quote_char = .char_string [.char_count];
```

```
;  2176          2297  7                                    quote_flag = true;
;  2177          2298  7                                    END
;  2178          2299  6                                ELSE
;  2179          2300  6                                    IF (.char_count LEQ .input_desc [dsc$w_length] - 4) AND
;  2180          2301  7                                       (.char_string [.char_count + 4] NEQ dbg$k_quote)
;  2181          2302  6                                    THEN
;  2182          2303  7                                        BEGIN
;  2183          2304  7                                        quote_char = .char_string [.:har_count];
;  2184          2305  7                                        quote_flag = true;
;  2185          2306  7                                        END
;  2186          2307  6                                    ELSE
;  2187          2308  6                                        IF (.char_count LEQ .input_desc [dsc$w_length] - 6) AND
;  2188          2309  7                                           (.char_string [.char_count + 6] EQL dbg$k_quote)
;  2189          2310  6                                        THEN
;  2190          2311  7                                            BEGIN
;  2191          2312  7                                            quote_char = .char_string [.char_count];
;  2192          2313  7                                            quote_flag = true;
;  2193          2314  6                                            END;
;  2194          2315  6                                END
;  2195          2316  5                            ELSE
;  2196          2317  6                                BEGIN
;  2197          2318  6                                IF .char_string [.char_count] EQL .quote_char
;  2198          2319  6                                THEN
;  2199          2320  6                                    quote_flag = false;
;  2200          2321  5                                END;
;  2201          2322  4                            END;
;  2202          2323  4
;  2203          2324  4                        char_count = .char_count + 1;
;  2204          2325  3                    END;
;  2205          2326  2            END;
;  2206          2327  2
;  2207          2328  2            ! Make sure the length is correct.
;  2208          2329  2            !
;  2209          2330  2            input_desc [dsc$w_length] = .input_desc [dsc$w_length] - .char_count;
;  2210          2331  2
;  2211          2332  2            ! Now try to get storage for the command string
;  2212          2333  2            !
;  2213          2334  2            cmd_desc [dsc$a_pointer] = dbg$get_tempmem((.char_count / %UPVAL) + 1);
;  2214          2335  2
;  2215          2336  2
;  2216          2337  2            ! Save away pointers both to the original input string, and to
;  2217          2338  2            ! the copied string in cmd_desc. These are used later as follows:
;  2218          2339  2            ! In the language C, a lower case name represents a distinct object
;  2219          2340  2            ! from its upper-case counterpart. Since we upper-case commands in
;  2220          2341  2            ! cmd_desc, we will need to go back to the original input_desc to
;  2221          2342  2            ! get at the original version of the name. For this, we need these
;  2222          2343  2            ! two pointers.
;  2223          2344  2            !
;  2224          2345  2            ! The pointer to the upcased string is actually a vector containing
;  2225          2346  2            ! pointers to the beginning and the end of the string.
;  2226          2347  2            !
;  2227          2348  2            dbg$gl_orig_command_ptr = .input_desc[dsc$a_pointer];
;  2228          2349  2            dbg$gl_upcase_command_ptr[0] = .cmd_desc[dsc$a_pointer];
;  2229          2350  2            dbg$gl_upcase_command_ptr[1] = .cmd_desc[dsc$a_pointer] + .char_count - 1;
;  2230          2351  2
;  2231          2352  2            ! Fill the command buffer
;  2232          2353  2            !
```

```
;  2233      2354  2      ch$move ( .char_count, .input_desc [dsc$a_pointer], .cmd_desc [dsc$a_pointer]);
;  2234      2355  2
;  2235      2356  2
;  2236      2357  2      ! Update the input descriptor pointer. Check for a comment to skip.
;  2237      2358  2      ! The comment character is '!' in all languages except C.
;  2238      2359  2      !
;  2239      2360  2      IF .char_string [.char_count] EQL '!'
;  2240      2361  2      THEN
;  2241      2362  3          BEGIN
;  2242      2363  3          input_desc [dsc$a_pointer] = .input_desc [dsc$a_pointer] +
;  2243      2364  3                                       .input_desc[dsc$w_length] +
;  2244      2365  3                                       .char_count;
;  2245      2366  3          input_desc [dsc$w_length] = 0;
;  2246      2367  3          END
;  2247      2368  2      ELSE
;  2248      2369  2          input_desc [dsc$a_pointer] = char_string [.char_count];
;  2249      2370  2
;  2250      2371  2
;  2251      2372  2      ! Update the command descriptor
;  2252      2373  2      !
;  2253      2374  2      cmd_desc [initial_ptr] = .cmd_desc [dsc$a_pointer];
;  2254      2375  2      cmd_desc [dsc$w_length] = .char_count;
;  2255      2376  2      char_string = .cmd_desc [dsc$a_pointer];
;  2256      2377  2
;  2257      2378  2
;  2258      2379  2      ! Now check for bad chars and translate to upper case
;  2259      2380  2      !
;  2260      2381  2      char_count = 0;
;  2261      2382  2      quote_flag = false;
;  2262      2383  2      WHILE .char_count LSS .cmd_desc [dsc$w_length]
;  2263      2384  2      DO
;  2264      2385  3          BEGIN
;  2265      2386  3          IF .char_string [.char_count] EQL dbg$k_tab
;  2266      2387  3          THEN
;  2267      2388  3              char_string [.char_count] = dbg$k_blank;  ! Convert tab to space
;  2268      2389  3
;  2269      2390  3          IF .char_string [.char_count] LSS dbg$k_blank
;  2270      2391  3          THEN
;  2271      2392  4              BEGIN
;  2272      2393  4              .message_vect = dbg$nmake_arg_vect (dbg$_invchar);
;  2273      2394  4              RETURN sts$k_severe;
;  2274      2395  4              END
;  2275      2396  3          ELSE
;  2276      2397  4              BEGIN
;  2277      2398  4              IF .char_string [.char_count] EQL dbg$k_quote
;  2278      2399  4                                           OR
;  2279      2400  4                 .char_string [.char_count] EQL dbg$k_dblquote
;  2280      2401  4              THEN
;  2281      2402  5                  BEGIN
;  2282      2403  5                  IF NOT .quote_flag
;  2283      2404  5                  THEN
;  2284      2405  5                      ! Make sure this is not a tick operator.  This is
;  2285      2406  5                      ! nasty stuff... (e.g. '(';') => TICK, but '(';')' => QUOTE)
;  2286      2407  5                      !
;  2287      2408  5                      IF (.char_string [.char_count] EQL dbg$k_quote) AND
;  2288      2409  5                         (.char_count LEQ .cmd_desc [dsc$w_length] - 2) AND
;  2289      2410  6                         (.char_string [.char_count + 2] EQL dbg$k_quote)
```

```
;  2290          2411  5                    THEN
;  2291          2412  6                        BEGIN
;  2292          2413  7                        IF (.char_string [.char_count + 1] NEQ dbg$k_left_parenthesis)
;  2293          2414  6                        THEN
;  2294          2415  7                            BEGIN
;  2295          2416  7                            quote_char = .char_string [.char_count];
;  2296          2417  7                            quote_flag = true;
;  2297          2418  7                            END
;  2298          2419  6                        ELSE
;  2299          2420  6                            IF (.char_count LEQ .cmd_desc [dsc$w_length] - 4) AND
;  2300          2421  7                               (.char_string [.char_count + 4] NEQ dbg$k_quote)
;  2301          2422  6                            THEN
;  2302          2423  7                                BEGIN
;  2303          2424  7                                quote_char = .char_string [.char_count];
;  2304          2425  7                                quote_flag = true;
;  2305          2426  7                                END
;  2306          2427  6                            ELSE
;  2307          2428  6                                IF (.char_count LEQ .cmd_desc [dsc$w_length] - 6) AND
;  2308          2429  7                                   (.char_string [.char_count + 6] EQL dbg$k_quote)
;  2309          2430  6                                THEN
;  2310          2431  7                                    BEGIN
;  2311          2432  7                                    quote_char = .char_string [.char_count];
;  2312          2433  7                                    quote_flag = true;
;  2313          2434  6                                    END;
;  2314          2435  6                        END
;  2315          2436  5                ELSE
;  2316          2437  6                    BEGIN
;  2317          2438  6                    IF .char_string [.char_count] EQL .quote_char
;  2318          2439  6                    THEN
;  2319          2440  6                        quote_flag = false;
;  2320          2441  5                    END;
;  2321          2442  4                END;
;  2322          2443  4
;  2323          2444  4                IF .char_string [.char_count] GEQ 'a'
;  2324          2445  4                                AND
;  2325          2446  4                   .char_string [.char_count] LEQ 'z'
;  2326          2447  4                                AND
;  2327          2448  4                   NOT .quote_flag
;  2328          2449  4                THEN
;  2329          2450  4                    char_string [.char_count] = .char_string [.char_count]
;  2330          2451  4                                              - dbg$k_lcbias;
;  2331          2452  3                END;
;  2332          2453  3
;  2333          2454  3            char_count = .char_count + 1;
;  2334          2455  3
;  2335          2456  2        END;
;  2336          2457  2
;  2337          2458  2
;  2338          2459  2    ! Termanate the command with a <cr>
;  2339          2460  2    !
;  2340          2461  2    char_string [.char_count] = dbg$k_car_return;
;  2341          2462  2    cmd_desc [dsc$w_length] = .cmd_desc [dsc$w_length] + 1;
;  2342          2463  2
;  2343          2464  2    RETURN sts$k_success;
;  2344          2465  1    END;
;  INFO#250                L1:2318
;  Referenced LOCAL symbol QUOTE_CHAR is probably not initialized
```

```
                                 OFFC 00000 GET_ADA_CMD_STRING:
                                            .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                    ; 2183
                              5E      04  C2 00002         SUBL2   #4, SP
                              5A  04  AC  D0 00005         MOVL    INPUT_DESC, R10                         ; 2229
                              59  04  AA  9E 00009         MOVAB   4(R10), R9
                              57      69  D0 0000D         MOVL    (R9), CHAR_STRING
                                      56  D4 00010         CLRL    CHAR_COUNT                              ; 2230
                              21    6647  91 00012         CMPB    (CHAR_COUNT)[CHAR_STRING], #33          ; 2235
                                      0C  12 00016         BNEQ    1$
                              50      6A  3C 00018         MOVZWL  (R10), R0                               ; 2239
                              69      50  C0 00019         ADDL2   R0, (R9)
                                      6A  B4 0001E         CLRW    (R10)                                   ; 2240
                              50      02  D0 00020         MOVL    #2, R0                                  ; 2241
                                      04  00023         RET
                          0C  AC      5A  D0 00024 1$:     MOVL    R10, CIS_DESC                           ; 2252
                              50  0C  AC  D0 00028         MOVL    CIS_DESC, R0                            ; 2253
                          14  A0      69  D0 0002C         MOVL    (R9), 20(R0)
                          34  A0      6A  B0 00030         MOVW    (R10), 52(R0)                           ; 2254
                              57      69  D0 00034         MOVL    (R9), CHAR_STRING                       ; 2259
                                      56  D4 00037         CLRL    CHAR_COUNT                              ; 2260
                                      5B  D4 00039         CLRL    QUOTE_FLAG                              ; 2261
        56          6A            10    00  ED 0003B 2$:     CMPZV   #0, #6, (R10), CHAR_COUNT           ; 2262
                                          03  14 00040         BGTR    4$
                                        0089  31 00042 3$:     BRW     12$
                              50      6647  9A 00045 4$:     MOVZBL  (CHAR_COUNT)[CHAR_STRING], R0           ; 2265
                              0D        50  91 00049         CMPB    R0, #13
                                        F4  13 0004C         BEQL    3$
                              0A        50  91 0004E         CMPB    R0, #10                                 ; 2267
                                        7B  13 00051         BEQL    12$
                                        50  D5 00053         TSTL    R0                                      ; 2269
                                        77  13 00055         BEQL    12$
                              0D        5B  E8 00057         BLBS    QUOTE_FLAG, 5$                          ; 2271
                              3B        50  91 0005A         CMPB    R0, #59
                                        6F  13 0005D         BEQL    12$
                              05        5B  E8 0005F         BLBS    QUOTE_FLAG, 5$                          ; 2273
                              21        50  91 00062         CMPB    R0, #33
                                        67  13 00065         BEQL    12$
                                        51  D4 00067 5$:     CLRL    R1                                      ; 2278
                              27        50  91 00069         CMPB    R0, #39
                                        04  12 0006C         BNEQ    6$
                                        51  D6 0006E         INCL    R1
                                        05  11 00070         BRB     7$
                              22        50  91 00072 6$:     CMPB    R0, #34                                 ; 2280
                                        52  12 00075         BNEQ    11$
                              4F        5B  E8 00077 7$:     BLBS    QUOTE_FLAG, 11$                         ; 2283
                              45        51  E9 0007A         BLBC    R1, 10$                                 ; 2288
                              51        6A  3C 0007D         MOVZWL  (R10), R1                               ; 2289
                              51        02  C2 00080         SUBL2   #2, R1
                              51        56  D1 00083         CMPL    CHAR_COUNT, R1
                                        3A  14 00086         BGTR    10$
                              27  02  A647  91 00088         CMPB    2(CHAR_COUNT)[CHAR_STRING], #39         ; 2290
                                        33  12 0008D         BNEQ    10$
```

```
                        28      01 A647   91 0008F          CMPB    1(CHAR_COUNT)[CHAR_STRING], #40          ; 2293
                                  24   12 00094             BNEQ    9$
                        51        6A   3C 00096             MOVZWL  (R10), R1                                ; 2300
                        51        04   C2 00099             SUBL2   #4, R1
                        51        56   D1 0009C             CMPL    CHAR_COUNT, R1
                                  07   14 0009F             BGTR    8$
                        27      04 A647   91 000A1          CMPB    4(CHAR_COUNT)[CHAR_STRING], #39          ; 2301
                                  12   12 000A6             BNEQ    9$
                        51        6A   3C 000A8 8$:         MOVZWL  (R10), R1                                ; 2308
                        51        06   C2 000AB             SUBL2   #6, R1
                        51        56   D1 000AE             CMPL    CHAR_COUNT, R1
                                  16   14 000B1             BGTR    11$
                        27      06 A647   91 000B3          CMPB    6(CHAR_COUNT)[CHAR_STRING], #39          ; 2309
                                  0F   12 000B8             BNEQ    11$
                        6E        50   D0 000BA 9$:         MOVL    R0, QUOTE_CHAR                           ; 2312
                        5B        01   D0 000BD             MOVL    #1, QUOTE_FLAG                           ; 2313
                                  07   11 000C0             BRB     11$                                      ; 2288
                        6E        50   D1 000C2 10$:        CMPL    R0, QUOTE_CHAR                           ; 2318
                                  02   12 000C5             BNEQ    11$
                                  5B   D4 000C7             CLRL    QUOTE_FLAG                               ; 2320
                                  56   D6 000C9 11$:        INCL    CHAR_COUNT                               ; 2324
                                FF6D   31 000CB             BRW     2$                                       ; 2262
                        6A        56   A2 000CE 12$:        SUBW2   CHAR_COUNT, (R10)                        ; 2330
                        58      08 AC   D0 000D1            MOVL    CMD_DESC, R8                             ; 2334
              50                56   04 C7 000D5            DIVL3   #4, CHAR_COUNT, R0
                                01 A0   9F 000D9            PUSHAB  1(R0)
    00000000G   00                  01   FB 000DC          CALLS   #1, DBG$GET_TEMPMEM
                      04        A8      50   D0 000E3      MOVL    R0, 4(R8)
                00000000'       EF      69   D0 000E7      MOVL    (R9), DBG$GL_ORIG_COMMAND_PTR            ; 2348
                00000000'       EF    04 A8  D0 000EE      MOVL    4(R8), DBG$GL_UPCASE_COMMAND_PTR         ; 2349
              50                56    04 A8  C1 000F6      ADDL3   4(R8), CHAR_COUNT, R0                    ; 2350
                00000000'       EF    FF A0  9E 000FB      MOVAB   -1(R0), DBG$GL_UPCASE_COMMAND_PTR+4
          04  B8        00       B9      56   28 00103      MOVC3   CHAR_COUNT, @0(R9), @4(R8)               ; 2354
                                  21        6647   91 00109 CMPB    (CHAR_COUNT)[CHAR_STRING], #33           ; 2360
                                              0E   12 0010D BNEQ    13$
                                  50          6A   3C 0010F MOVZWL  (R10), R0                               ; 2364
                                  50          69   C0 00112 ADDL2   (R9), R0
                        69        50          56   C1 00115 ADDL3   CHAR_COUNT, R0, (R9)                    ; 2365
                                              6A   B4 00119 CLRW    (R10)                                    ; 2366
                                              04   11 0011B BRB     14$                                      ; 2360
                        69                    57        56 C1 0011D 13$:    ADDL3   CHAR_COUNT, CHAR_STRING, (R9)  ; 2369
                              08      A8      04 A8   D0 00121 14$:         MOVL    4(R8), 8(R8)             ; 2374
                                      68        56   B0 00126             MOVW    CHAR_COUNT, (R8)         ; 2375
                                      57      04 A8   D0 00129             MOVL    4(R8), CHAR_STRING       ; 2376
                                              56   D4 0012D             CLRL    CHAR_COUNT               ; 2381
                                              5B   D4 0012F             CLRL    QUOTE_FLAG               ; 2382
              56        68                  10        00 ED 00131 15$:    CMPZV   #0, #16, (R8), CHAR_COUNT        ; 2383
                                              03   14 00136             BGTR    16$
                                            00A2   31 00138             BRW     26$
                                      09          6647   91 0013B 16$:  CMPB    (CHAR_COUNT)[CHAR_STRING], #9    ; 2386
                                              04   12 0013F             BNEQ    17$
                                    6647          20   90 00141       MOVB    #32, (CHAR_COUNT)[CHAR_STRING]   ; 2388
                                      52          6647   9A 00145 17$: MOVZBL  (CHAR_COUNT)[CHAR_STRING], R2   ; 2390
                                      20            52   91 00149     CMPB    R2, #32
                                                    15   1E 0014C     BGEQU   18$
                                          000281A0    8F   DD 00150   PUSHL   #164256                         ; 2393
                00000000G   00                      01   FB 00154     CALLS   #1, DBG$NMAKE_ARG_VECT
```

```
                10 BC          50  D0 0015B          MOVL    R0, @MESSAGE_VECT
                   50          04  D0 0015F          MOVL    #4, R0                                  ; 2394
                                   04 00162          RET
                               50  D4 00163 18$:     CLRL    R0                                      ; 2398
                   27          52  91 00165          CMPB    R2, #39
                               04  12 00168          BNEQ    19$
                               50  D6 0016A          INCL    R0
                               05  11 0016C          BRB     20$
                   22          52  91 0016E 19$:     CMPB    R2, #34                                 ; 2400
                               52  12 00171          BNEQ    24$
                   4F          5B  E8 00173 20$:     BLBS    QUOTE_FLAG, 24$                         ; 2403
                   45          50  E9 00176          BLBC    R0, 23$                                 ; 2408
                   50          68  3C 00179          MOVZWL  (R8), R0                                ; 2409
                   50          02  C2 0017C          SUBL2   #2, R0
                   50          56  D1 0017F          CMPL    CHAR_COUNT, R0
                               3A  14 00182          BGTR    23$
                   27    02 A647  91 00184          CMPB    2(CHAR_COUNT)[CHAR_STRING], #39         ; 2410
                               33  12 00189          BNEQ    23$
                   28    01 A647  91 0018B          CMPB    1(CHAR_COUNT)[CHAR_STRING], #40         ; 2413
                               24  12 00190          BNEQ    22$
                   50          68  3C 00192          MOVZWL  (R8), R0                                ; 2420
                   50          04  C2 00195          SUBL2   #4, R0
                   50          56  D1 00198          CMPL    CHAR_COUNT, R0
                               07  14 0019B          BGTR    21$
                   27    04 A647  91 0019D          CMPB    4(CHAR_COUNT)[CHAR_STRING], #39         ; 2421
                               12  12 001A2          BNEQ    22$
                   50          68  3C 001A4 21$:     MOVZWL  (R8), R0                                ; 2428
                   50          06  C2 001A7          SUBL2   #6, R0
                   50          56  D1 001AA          CMPL    CHAR_COUNT, R0
                               16  14 001AD          BGTR    24$
                   27    06 A647  91 001AF          CMPB    6(CHAR_COUNT)[CHAR_STRING], #39         ; 2429
                               0F  12 001B4          BNEQ    24$
                   6E          52  D0 001B6 22$:     MOVL    R2, QUOTE_CHAR                          ; 2432
                   5B          01  D0 001B9          MOVL    #1, QUOTE_FLAG                          ; 2433
                               07  11 001BC          BRB     24$                                     ; 2408
                   6E          52  D1 001BE 23$:     CMPL    R2, QUOTE_CHAR                          ; 2438
                               02  12 001C1          BNEQ    24$
                               5B  D4 001C3          CLRL    QUOTE_FLAG                              ; 2440
                61 8F          52  91 001C5 24$:     CMPB    R2, #97                                 ; 2444
                               0D  1F 001C9          BLSSU   25$
                7A 8F          52  91 001CB          CMPB    R2, #122                                ; 2446
                               07  1A 001CF          BGTRU   25$
                   04          5B  E8 001D1          BLBS    QUOTE_FLAG, 25$                         ; 2448
                 6647          20  82 001D4          SUBB2   #32, (CHAR_COUNT)[CHAR_STRING]          ; 2451
                               56  D6 001DB 25$:     INCL    CHAR_COUNT                              ; 2454
                             FF54  31 001DA          BRW     15$                                     ; 2383
                 6647          0D  90 001DD 26$:     MOVB    #13, (CHAR_COUNT)[CHAR_STRING]          ; 2461
                               68  B6 001E1          INCW    (R8)                                    ; 2462
                   50          01  D0 001E3          MOVL    #1, R0                                  ; 2464
                                   04 001E6          RET                                             ; 2465

; Routine Size:  487 bytes,    Routine Base:  DBG$CODE + 0980
```

```
;  2346     2466  1 ROUTINE GET_NORMAL_CMD_STRING(INPUT_DESC, CMD_DESC, CIS_DESC, MESSAGE_VECT) =
;  2347     2467  1
;  2348     2468  1 !++
;  2349     2469  1 ! FUNCTIONAL DESCRIPTION:
;  2350     2470  1 !
;  2351     2471  1 !       This routine gets the first command from the input line.  Also,
;  2352     2472  1 !       uppercases the line except for what is in quotes.  This routine
;  2353     2473  1 !       takes care of stripping the comments off the end of a DEBUG
;  2354     2474  1 !       command. For all languages except C, the comment character is
;  2355     2475  1 !       '!'.
;  2356     2476  1 !
;  2357     2477  1 ! FORMAL PARAMETERS:
;  2358     2478  1 !
;  2359     2479  1 !       input_desc -            a VAX standard descriptor of the input line
;  2360     2480  1 !
;  2361     2481  1 !       cmd_desc -              a descriptor that will hold the next command
;  2362     2482  1 !                               line.
;  2363     2483  1 !       cis_desc -              a descriptor for the current command input
;  2364     2484  1 !                               stream.  Just another copy of the above in
;  2365     2485  1 !                               case the command is a WHILE-DO.
;  2366     2486  1 !       message_vect -          the address of a longword to contain the address
;  2367     2487  1 !                               of a message argument vector.
;  2368     2488  1 !
;  2369     2489  1 ! ROUTINE VALUE:
;  2370     2490  1 !
;  2371     2491  1 !       A status of the routine.
;  2372     2492  1 !
;  2373     2493  1 !--
;  2374     2494  1
;  2375     2495  2     BEGIN
;  2376     2496  2
;  2377     2497  2     MAP
;  2378     2498  2         INPUT_DESC      : REF dbg$stg_desc,     ! Command line
;  2379     2499  2         CIS_DESC        : REF CIS$LINR,         ! Current command input stream
;  2380     2500  2         CMD_DESC        : REF BLOCK [,BYTE];    ! We don't REF to dbg$stg_desc
;  2381     2501  2                                                 ! because of the extra longword
;  2382     2502  2                                                 ! for the initial dsc$a_pointer
;  2383     2503  2     LOCAL
;  2384     2504  2         CHAR_COUNT,
;  2385     2505  2         CHAR_STRING     : REF VECTOR [,BYTE],   ! Vector of characters
;  2386     2506  2         QUOTE_FLAG,
;  2387     2507  2         QUOTE_CHAR;
;  2388     2508  2
;  2389     2509  2
;  2390     2510  2     char_string = .input_desc[dsc$a_pointer];
;  2391     2511  2     char_count = 0;
;  2392     2512  2
;  2393     2513  2     ! Check for a comment line. For all languages except C, the comment
;  2394     2514  2     ! character is '!'.
;  2395     2515  2     !
;  2396     2516  2     IF .char_string [.char_count] EQL '!'
;  2397     2517  2     THEN
;  2398     2518  3         BEGIN
;  2399     2519  3         input_desc [dsc$a_pointer] = .input_desc [dsc$a_pointer] +
;  2400     2520  3                                      .input_desc [dsc$w_length];
;  2401     2521  3         input_desc [dsc$w_length] = 0;
;  2402     2522  3         RETURN sts$k_error;
```

```
;  2403          2523  2          END;
;  2404          2524  2
;  2405          2525  2
;  2406          2526  2      ! Before proceeding, we fill in the CIS$A_WHILE_CLAUSE field
;  2407          2527  2      ! to point to the beginning of the command. This is in case the command
;  2408          2528  2      ! is a WHILE; then we are able to iterate by backing up to the
;  2409          2529  2      ! beginning of the command.
;  2410          2530  2      ! The following code relies on the fact that INPUT_DESC is superimposed
;  2411          2531  2      ! on the top link pointed to by DBG$GL_CISHEAD.
;  2412          2532  2      !
;  2413          2533  2      cis_desc = .input_desc;
;  2414          2534  2      cis_desc [cis$a_while_clause] = .input_desc [dsc$a_pointer];
;  2415          2535  2      cis_desc [cis$w_while_length] = .input_desc [dsc$w_length];
;  2416          2536  2
;  2417          2537  2
;  2418          2538  2      ! Now count the characters in the command
;  2419          2539  2      !
;  2420          2540  2      char_string = .input_desc [dsc$a_pointer];
;  2421          2541  2      char_count = 0;
;  2422          2542  2      quote_flag = false;
;  2423          2543  2      WHILE .input_desc [dsc$w_length] GTR 0
;  2424          2544  2      DO
;  2425          2545  3          BEGIN
;  2426          2546  3              IF .char_string [.char_count] EQL dbg$k_car_return
;  2427          2547  3                                  OR
;  2428          2548  3                 .char_string [.char_count] EQL dbg$k_line_feed
;  2429          2549  3                                  OR
;  2430          2550  3                 .char_string [.char_count] EQL dbg$k_null
;  2431          2551  3                                  OR
;  2432          2552  4                 ((NOT .quote_flag) AND .char_string [.char_count] EQL ';')
;  2433          2553  4                                  OR
;  2434          2554  4                 ((NOT .quote_flag) AND .char_string [.char_count] EQL '!')
;  2435          2555  3              THEN
;  2436          2556  3                  EXITLOOP
;  2437          2557  3              ELSE
;  2438          2558  4                  BEGIN
;  2439          2559  4                      IF .char_string [.char_count] EQL dbg$k_quote
;  2440          2560  4                                          OR
;  2441          2561  4                         .char_string [.char_count] EQL dbg$k_dblquote
;  2442          2562  4                      THEN
;  2443          2563  5                          BEGIN
;  2444          2564  5                          IF NOT .quote_flag
;  2445          2565  5                          THEN
;  2446          2566  6                              BEGIN
;  2447          2567  6                              quote_char = .char_string [.char_count];
;  2448          2568  6                              quote_flag = true;
;  2449          2569  6                              END
;  2450          2570  5                          ELSE
;  2451          2571  6                              BEGIN
;  2452          2572  6                              IF .char_string [.char_count] EQL .quote_char
;  2453          2573  6                              THEN
;  2454          2574  6                                  quote_flag = false;
;  2455          2575  5                              END;
;  2456          2576  4                          END;
;  2457          2577  4
;  2458          2578  4                      char_count = .char_count + 1;
;  2459          2579  4                      input_desc [dsc$w_length] = .input_desc [dsc$w_length] - 1;
```

```
;2460          2580  3                END;
;2461          2581  2            END;
;2462          2582  2
;2463          2583  2
;2464          2584  2        ! Now try to get storage for the command string
;2465          2585  2        !
;2466          2586  2        cmd_desc [dsc$a_pointer] = dbg$get_tempmem((.char_count / %UPVAL) + 1);
;2467          2587  2
;2468          2588  2
;2469          2589  2        ! Save away pointers both to the original input string, and to
;2470          2590  2        ! the copied string in cmd_desc. These are used later as follows:
;2471          2591  2        ! In the language C, a lower case name represents a distinct object
;2472          2592  2        ! from its upper-case counterpart. Since we upper-case commands in
;2473          2593  2        ! cmd_desc, we will need to go back to the original input_desc to
;2474          2594  2        ! get at the original version of the name. For this, we need these
;2475          2595  2        ! two pointers.
;2476          2596  2        !
;2477          2597  2        ! The pointer to the upcased string is actually a vector containing
;2478          2598  2        ! pointers to the beginning and the end of the string.
;2479          2599  2        !
;2480          2600  2        dbg$gl_orig_command_ptr = .input_desc[dsc$a_pointer];
;2481          2601  2        dbg$gl_upcase_command_ptr[0] = .cmd_desc[dsc$a_pointer];
;2482          2602  2        dbg$gl_upcase_command_ptr[1] = .cmd_desc[dsc$a_pointer] + .char_count - 1;
;2483          2603  2
;2484          2604  2        ! Fill the command buffer
;2485          2605  2        !
;2486          2606  2        ch$move ( .char_count, .input_desc [dsc$a_pointer], .cmd_desc [dsc$a_pointer]);
;2487          2607  2
;2488          2608  2
;2489          2609  2        ! Update the input descriptor pointer. Check for a comment to skip.
;2490          2610  2        ! The comment character is '!' in all languages except C.
;2491          2611  2        !
;2492          2612  2        IF .char_string [.char_count] EQL '!'
;2493          2613  2        THEN
;2494          2614  3            BEGIN
;2495          2615  3            input_desc [dsc$a_pointer] = .input_desc [dsc$a_pointer] +
;2496          2616  3                                         .input_desc[dsc$w_length] +
;2497          2617  3                                         .char_count;
;2498          2618  3            input_desc [dsc$w_length] = 0;
;2499          2619  3            END
;2500          2620  2        ELSE
;2501          2621  2            input_desc [dsc$a_pointer] = char_string [.char_count];
;2502          2622  2
;2503          2623  2
;2504          2624  2        ! Update the command descriptor
;2505          2625  2        !
;2506          2626  2        cmd_desc [initial_ptr] = .cmd_desc [dsc$a_pointer];
;2507          2627  2        cmd_desc [dsc$w_length] = .char_count;
;2508          2628  2        char_string = .cmd_desc [dsc$a_pointer];
;2509          2629  2
;2510          2630  2
;2511          2631  2        ! Now check for bad chars and translate to upper case
;2512          2632  2        !
;2513          2633  2        char_count = 0;
;2514          2634  2        quote_flag = false;
;2515          2635  2        WHILE .char_count LSS .cmd_desc [dsc$w_length]
;2516          2636  2        DO
```

```
; 2517          2637  3          BEGIN
; 2518          2638  3              IF .char_string [.char_count] EQL dbg$k_tab
; 2519          2639  3              THEN
; 2520          2640  3                  char_string [.char_count] = dbg$k_blank;  ! Convert tab to space
; 2521          2641  3
; 2522          2642  3              IF .char_string [.char_count] LSS dbg$k_blank
; 2523          2643  3              THEN
; 2524          2644  4                  BEGIN
; 2525          2645  4                  .message_vect = dbg$nmake_arg_vect (dbg$_invchar);
; 2526          2646  4                  RETURN sts$k_severe;
; 2527          2647  4                  END
; 2528          2648  3              ELSE
; 2529          2649  4                  BEGIN
; 2530          2650  4                      IF .char_string [.char_count] EQL dbg$k_quote
; 2531          2651  4                                                    OR
; 2532          2652  4                         .char_string [.char_count] EQL dbg$k_dblquote
; 2533          2653  4                      THEN
; 2534          2654  5                          BEGIN
; 2535          2655  5                          IF NOT .quote_flag
; 2536          2656  5                          THEN
; 2537          2657  6                              BEGIN
; 2538          2658  6                              quote_char = .char_string [.char_count];
; 2539          2659  6                              quote_flag = true;
; 2540          2660  6                              END
; 2541          2661  5                          ELSE
; 2542          2662  6                              BEGIN
; 2543          2663  6                              IF .char_string [.char_count] EQL .quote_char
; 2544          2664  6                              THEN
; 2545          2665  6                                  quote_flag = false;
; 2546          2666  5                              END;
; 2547          2667  4                          END;
; 2548          2668  4
; 2549          2669  4                      IF .char_string [.char_count] GEQ 'a'
; 2550          2670  4                                                    AND
; 2551          2671  4                         .char_string [.char_count] LEQ 'z'
; 2552          2672  4                                                    AND
; 2553          2673  4                         NOT .quote_flag
; 2554          2674  4                      THEN
; 2555          2675  4                          char_string [.char_count] = .char_string [.char_count]
; 2556          2676  4                                                    - dbg$k_lcbias;
; 2557          2677  3                      END;
; 2558          2678  3
; 2559          2679  3                  char_count = .char_count + 1;
; 2560          2680  3
; 2561          2681  2              END;
; 2562          2682  2
; 2563          2683  2
; 2564          2684  2          ! Termanate the command with a <cr>
; 2565          2685  2          !
; 2566          2686  2          char_string [.char_count] = dbg$k_car_return;
; 2567          2687  2          cmd_desc [dsc$w_length] = .cmd_desc [dsc$w_length] + 1;
; 2568          2688  2
; 2569          2689  2          RETURN sts$k_success;
; 2570          2690  1          END;
; INFO#250                   L1:2572
; Referenced LOCAL symbol QUOTE_CHAR is probably not initialized
```

```
                        OFFC 00000 GET_NORMAL_CMD_STRING:
                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                ; 2466
                          5E    04 C2 00002          SUBL2   #4, SP                           ;
                          5A 04 AC D0 00005          MOVL    INPUT_DESC, R10                  ; 2510
                          59 04 AA 9E 00009          MOVAB   4(R10), R9                       ;
                          58    69 D0 0000D          MOVL    (R9), CHAR_STRING                ;
                                56 D4 00010          CLRL    CHAR_COUNT                       ; 2511
                          21  6648 91 00012          CMPB    (CHAR_COUNT)[CHAR_STRING], #33   ; 2516
                                0C 12 00016          BNEQ    1$                               ;
                          50    6A 3C 00018          MOVZWL  (R10), R0                        ; 2520
                          69    50 C0 0001B          ADDL2   R0, (R9)                         ;
                                6A B4 0001E          CLRW    (R10)                            ; 2521
                          50    02 D0 00020          MOVL    #2, R0                           ; 2522
                                   04 00023          RET                                      ;
                       0C AC    5A D0 00024 1$:      MOVL    R10, CIS_DESC                    ; 2533
                       50 0C AC D0 00028             MOVL    CIS_DESC, R0                     ; 2534
                    14 A0    69 D0 0002C             MOVL    (R9), 20(R0)                     ;
                    34 A0    6A B0 00030             MOVW    (R10), 52(R0)                    ; 2535
                          58    69 D0 00034          MOVL    (R9), CHAR_STRING                ; 2540
                                56 D4 00037          CLRL    CHAR_COUNT                       ; 2541
                                5B D4 00039          CLRL    QUOTE_FLAG                       ; 2542
                                6A B5 0003B 2$:      TSTW    (R10)                            ; 2543
                                44 13 0003D          BEQL    7$                               ;
                          50  6648 9A 0003F          MOVZBL  (CHAR_COUNT)[CHAR_STRING], R0    ; 2546
                          0D    50 91 00043          CMPB    R0, #13                          ;
                                38 13 00046          BEQL    7$                               ;
                          0A    50 91 00048          CMPB    R0, #10                          ; 2548
                                36 13 0004B          BEQL    7$                               ;
                                50 D5 0004D          TSTL    R0                               ; 2550
                                32 13 0004F          BEQL    7$                               ;
                          0D    5B E8 00051          BLBS    QUOTE_FLAG, 3$                   ; 2552
                          3B    50 91 00054          CMPB    R0, #59                          ;
                                2A 13 00057          BEQL    7$                               ;
                          05    5B E8 00059          BLBS    QUOTE_FLAG, 3$                   ; 2554
                          21    50 91 0005C          CMPB    R0, #33                          ;
                                22 13 0005F          BEQL    7$                               ;
                          27    50 91 00061 3$:      CMPB    R0, #39                          ; 2559
                                05 13 00064          BEQL    4$                               ;
                          22    50 91 00066          CMPB    R0, #34                          ; 2561
                                12 12 00069          BNEQ    6$                               ;
                          08    5B E8 0006B 4$:      BLBS    QUOTE_FLAG, 5$                   ; 2564
                          6E    50 D0 0006E          MOVL    R0, QUOTE_CHAR                   ; 2567
                          5B    01 D0 00071          MOVL    #1, QUOTE_FLAG                   ; 2568
                                07 11 00074          BRB     6$                               ; 2564
                          6E    50 D1 00076 5$:      CMPL    R0, QUOTE_CHAR                   ; 2572
                                02 12 00079          BNEQ    6$                               ;
                                5B D4 0007B          CLRL    QUOTE_FLAG                       ; 2574
                                56 D6 0007D 6$:      INCL    CHAR_COUNT                       ; 2578
                                6A B7 0007F          DECW    (R10)                            ; 2579
                                B8 11 00081          BRB     2$                               ; 2543
                          57 08 AC D0 00083 7$:      MOVL    CMD_DESC, R7                     ; 2586
                50        56    04 C7 00087          DIVL3   #4, CHAR_COUNT, R0               ;
                             01 A0 9F 0008B          PUSHAB  1(R0)                            ;
```

```
                        00000000G   00        01  FB 0008E          CALLS   #1, DBG$GET_TEMPMEM                                ;
                              04    A7        50  D0 00095          MOVL    R0, 4(R7)                                          ;
                        00000000'   EF        69  D0 00099          MOVL    (R9), DBG$GL_ORIG_COMMAND_PTR                      ; 2600
                        00000000'   EF  04    A7  D0 000A0          MOVL    4(R7), DBG$GL_UPCASE_COMMAND_PTR                   ; 2601
                             5C             56  04 A7  C1 000A8     ADDL3   4(R7), CHAR_COUNT, R0                              ; 2602
                        00000000'   EF  FF    A0  9E 000AD          MOVAB   -1(R0), DBG$GL_UPCASE_COMMAND_PTR+4                ;
                  04    B7            00    B9  56  28 000B5      MOVC3   CHAR_COUNT, @0(R9), @4(R7)                         ; 2606
                                        21      6648  91 000BB      CMPB    (CHAR_COUNT)[CHAR_STRING], #33                     ; 2612
                                                  0E  12 000BF      BNEQ    8$                                                 ;
                                        50        6A  3C 000C1      MOVZWL  (R10), R0                                          ; 2616
                                        50        69  C0 000C4      ADDL2   (R9), R0                                           ;
                                    69  50        56  C1 000C7      ADDL3   CHAR_COUNT, R0, (R9)                               ; 2617
                                                  6A  B4 000CB      CLRW    (R10)                                              ; 2618
                                                  04  11 000CD      BRB     9$                                                 ; 2612
                                    69  58        56  C1 000CF  8$:  ADDL3   CHAR_COUNT, CHAR_STRING, (R9)                     ; 2621
                                  08    A7  04    A7  D0 000D3  9$:  MOVL    4(R7), 8(R7)                                       ; 2626
                                        67        56  B0 000D8      MOVW    CHAR_COUNT, (R7)                                   ; 2627
                                        58  04    A7  D0 000DB      MOVL    4(R7), CHAR_STRING                                 ; 2628
                                                  56  D4 000DF      CLRL    CHAR_COUNT                                         ; 2633
                                                  5B  D4 000E1      CLRL    QUOTE_FLAG                                         ; 2634
                             56     67        10        00  ED 000E3 10$:  CMPZV   #0, #16, (R7), CHAR_COUNT                  ; 2635
                                                  5B  15 000E8      BLEQ    17$                                                ;
                                        09      6648  91 000EA      CMPB    (CHAR_COUNT)[CHAR_STRING], #9                      ; 2638
                                                  04  12 000EE      BNEQ    11$                                                ;
                                      6648        20  90 000F0      MOVB    #32, (CHAR_COUNT)[CHAR_STRING]                     ; 2640
                                        52      6648  9A 000F4 11$:  MOVZBL  (CHAR_COUNT)[CHAR_STRING], R2                   ; 2642
                                        20        52  91 000F8      CMPB    R2, #32                                            ;
                                                  15  1E 000FB      BGEQU   12$                                                ;
                                  000281A0    8F    DD 000FD      PUSHL   #164256                                            ; 2645
                        00000000G   00        01  FB 00103      CALLS   #1, DBG$NMAKE_ARG_VECT                             ;
                              10    BC        50  D0 0010A      MOVL    R0, @MESSAGE_VECT                                  ;
                                        50        04  D0 0010E      MOVL    #4, R0                                             ; 2646
                                                  04 00111      RET                                                        ;
                                        27        52  91 00112 12$:  CMPB    R2, #39                                            ; 2650
                                                  05  13 00115      BEQL    13$                                                ;
                                        22        52  91 00117      CMPB    R2, #34                                            ; 2652
                                                  12  12 0011A      BNEQ    15$                                                ;
                                        08        5B  E8 0011C 13$:  BLBS    QUOTE_FLAG, 14$                                    ; 2655
                                        6E        52  D0 0011F      MOVL    R2, QUOTE_CHAR                                     ; 2658
                                        5B        01  D0 00122      MOVL    #1, QUOTE_FLAG                                     ; 2659
                                                  07  11 00125      BRB     15$                                                ; 2655
                                        6E        52  D1 00127 14$:  CMPL    R2, QUOTE_CHAR                                     ; 2663
                                                  02  12 0012A      BNEQ    15$                                                ;
                                                  5B  D4 0012C      CLRL    QUOTE_FLAG                                         ; 2665
                                  61    8F        52  91 0012E 15$:  CMPB    R2, #97                                            ; 2669
                                                  0D  1F 00132      BLSSU   16$                                                ;
                                  7A    8F        52  91 00134      CMPB    R2, #122                                           ; 2671
                                                  07  1A 00138      BGTRU   16$                                                ;
                                        04        5B  E8 0013A      BLBS    QUOTE_FLAG, 16$                                    ; 2673
                                      6648        20  82 0013D      SUBB2   #32, (CHAR_COUNT)[CHAR_STRING]                     ; 2676
                                                  56  D6 00141 16$:  INCL    CHAR_COUNT                                         ; 2679
                                                  9E  11 00143      BRB     10$                                                ; 2635
                                      6648        0D  90 00145 17$:  MOVB    #13, (CHAR_COUNT)[CHAR_STRING]                     ; 2686
                                                  67  B6 00149      INCW    (R7)                                               ; 2687
                                        50        01  D0 0014B      MOVL    #1, R0                                             ; 2689
                                                  04 0014E      RET                                                        ; 2690
```

```
; Routine Size:  335 bytes,    Routine Base:  DBG$CODE + 0B67

; 2571           2691  1
; 2572           2692  0 END ELUDOM


                                                                .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;       Name                      Bytes                         Attributes
;
;  DBG$GLOBAL                        12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$OWN                           52  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$CODE                        3254  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$PLIT                          43  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                              Library Statistics
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        16         0        1000      00:01.7
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1     32         0         0           7      00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1     1545        48         3          97      00:02.0
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                           418        13         3          31      00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1      386         3         0          22      00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1      150         1         0          12      00:00.3


; Information:  4
; Warnings:     0
; Errors:       0


;                                  COMMAND QUALIFIERS

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

; Size:          3254 code + 107 data bytes
; Run Time:         01:09.5
; Elapsed Time:     03:33.8
; Lines/CPU Min:     2325
; Lexemes/CPU-Min: 17284
; Memory Used:  252 pages
; Compilation Complete
```

DBGMSG
LIS

DBGNCANCL
LIS

DBGNCNTRL
LIS

DBGMOD
LIS

DBGNERMSG
LIS

DBGNHELP
LIS

DBGNPARSE
LIS

DBGNEXCTE
LIS

DBGNPNP
LIS