```
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 SSSSSSSS DDDDDDDD AAAAAA TTTTTTTTTT AAAAAA
DDDDDDDD BBBBBBBB GGGGGGGG NN NN SSSSSSSS DDDDDDDD AAAAAA TTTTTTTTTT AAAAAA
DD DD BB BB GG NN NN SS DD DD AA AA TT AA AA
DD DD BB BB GG NN NN SS DD DD AA AA TT AA AA
DD DD BB BB GG NNNN NN SS DD DD AA AA TT AA AA
DD DD BB BB GG NNNN NN SS DD DD AA AA TT AA AA
DD DD BBBBBBBB GG NN NN NN SSSSSS DD DD AA AA TT AA AA
DD DD BBBBBBBB GG NN NN NN SSSSSS DD DD AA AA TT AA AA
DD DD BB BB GG GGGGG NN NNNN SS DD DD AAAAAAAAAA TT AAAAAAAAAA
DD DD BB BB GG GGGGG NN NNNN SS DD DD AAAAAAAAAA TT AAAAAAAAAA
DD DD BB BB GG GG NN NN SS DD DD AA AA TT AA AA
DD DD BB BB GG GG NN NN SS DD DD AA AA TT AA AA
DDDDDDDD BBBBBBBB GGGGGG NN NN SSSSSSSS DDDDDDDD AA AA TT AA AA
DDDDDDDD BBBBBBBB GGGGGG NN NN SSSSSSSS DDDDDDDD AA AA TT AA AA
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
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

```
;     1          0001  0 MODULE DBGNSDATA (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                             *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                             *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1 !
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:
;    32          0032  1 !
;    33          0033  1 !       DEBUG
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This module consists of data structures and data structure management
;    38          0038  1 !       routines that manage the user's session specific data during the
;    39          0039  1 !       processing of debugger commands while under the control of the new
;    40          0040  1 !       Command Line Interpreter. Specifically, this module keeps track of
;    41          0041  1 !       scopes, types, override types, radicies, current location, and
;    42          0042  1 !       last value.
;    43          0043  1 !
;    44          0044  1 ! ENVIRONMENT:
;    45          0045  1 !
;    46          0046  1 !       VAX/VMS
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:
;    49          0049  1 !
;    50          0050  1 !       David Plummer
;    51          0051  1 !
;    52          0052  1 ! CREATION DATE:
;    53          0053  1 !
;    54          0054  1 !       10-Jul-80
;    55          0055  1 !
;    56          0056  1 ! VERSION:
;    57          0057  1 !
```

```
;   58          0058  1 !       V02.2-001
;   59          0059  1 !
;   60          0060  1 ! MODIFIED BY:
;   61          0061  1 !
;   62          0062  1 !--
```

```
;    64          0063  1 !
;    65          0064  1 ! TABLE OF CONTENTS:
;    66          0065  1 !
;    67          0066  1
;    68          0067  1 FORWARD ROUTINE
;    69          0068  1         DBG$NGET_RADIX,                     ! Returns the user specified or default radix
;    70          0069  1         DBG$NGET_TRANS_RADIX,               ! Translates a default radix into the actual radix
;    71          0070  1         DBG$NGET_DEFAULT_TYPE   : NOVALUE,  ! Returns user set default type
;    72          0071  1         DBG$NGET_OVERRIDE_TYPE,             ! Returns user set override type
;    73          0072  1         DBG$NGET_MODE,                      ! Returns true if 'symbolic', false
;    74          0073  1                                             ! if 'nosymbolic'
;    75          0074  1         DBG$NREPORT_LAST_VAL,               ! Returns the last value given to the
;    76          0075  1                                             ! user that was stored by the DEBUGGER
;    77          0076  1         DBG$NREPORT_LAST_LOC,               ! Returns the last location that was
;    78          0077  1                                             ! shown to the user by the DEBUGGER
;    79          0078  1         DBG$NSET_LAST_TYPLEN    : NOVALUE,  ! Stores potential last loc type and length
;    80          0079  1         DBG$NGET_POTENTIAL_TYPE : NOVALUE,  ! Returns the type associted with '.'
;    81          0080  1                                             ! by the Address Expression Interpreter
;    82          0081  1         DBG$NSAVE_LAST_VAL      : NOVALUE,  ! Stores a value descriptor for last val
;    83          0082  1         DBG$NSAVE_LAST_LOC      : NOVALUE,  ! Stores an address expression desc for
;    84          0083  1                                             ! current location
;    85          0084  1         DBG$NCANCEL_LOC_AND_VAL : NOVALUE,  ! Cancel last val and current loc for both
;    86          0085  1                                             ! old and new debuggers
;    87          0086  1         DBG$NPERM_SYM_INT;                  ! Permanent symbol Interpreter
;    88          0087  1
;    89          0088  1
;    90          0089  1 !
;    91          0090  1 ! REQUIRE FILES:
;    92          0091  1 !
;    93          0092  1
;    94          0093  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    95          0227  1
;    96          0228  1 LIBRARY 'LIB$:DBGGEN.L32';
;    97          0229  1
;    98          0230  1 !
;    99          0231  1 ! OWN STORAGE:
;   100          0232  1 !
;   101          0233  1 OWN
;   102          0234  1         LAST_VALUE              : REF dbg$dhead INITIAL (0), ! Pointer to last value
;   103          0235  1         LAST_LOC                : REF dbg$aed INITIAL (0),   ! Pointer to last location
;   104          0236  1         POTENTIAL_LAST_TYPE     : INITIAL (-1),              ! Possible type for current loc
;   105          0237  1         POTENTIAL_LAST_LENGTH   : INITIAL (0),               ! Possible length for current loc
;   106          0238  1         LAST_TYPE               : INITIAL (-1),              ! Real type for current loc
;   107          0239  1         LAST_LENGTH             : INITIAL (0);               ! Real length for current loc
;   108          0240  1
;   109          0241  1
;   110          0242  1
;   111          0243  1 !
;   112          0244  1 ! EXTERNAL REFERENCES:
;   113          0245  1 !
;   114          0246  1
;   115          0247  1 EXTERNAL ROUTINE
;   116          0248  1         DBG$COPY_MEMORY,                    ! Make a copy of a memory block
;   117          0249  1         DBG$GET_TEMPMEM,                    ! Get temporary memory block
;   118          0250  1         DBG$NCOPY_DESC,                     ! Copies primary and value descriptors
;   119          0251  1         DBG$NFREE_DESC,                     ! Frees copied descriptors
;   120          0252  1         DBG$NGET_SYMID,                     ! Obtains symid list for descriptors
```

```
;  121          0253  1          DBG$NMAKE_ARG_VECT,                    ! Constructs a message arguemnt vector
;  122          0254  1          DBG$NOUT_INFO,                         ! Outputs an informational message
;  123          0255  1          DBG$SET_MOD_LVL,                       ! Sets mode pointer
;  124          0256  1          DBG$STA_LOCK_SYMID   : NOVALUE,        ! Saves symids
;  125          0257  1          DBG$STA_UNLOCK_SYMID : NOVALUE,        ! Releases symids
;  126          0258  1          DBG$REL_MEMORY : NOVALUE;              ! Release memory block
;  127          0259  1
;  128          0260  1 EXTERNAL
;  129          0261  1          DBG$RUNFRAME           : BLOCK [,BYTE],    ! User runframe
;  130          0262  1          DBG$GB_RADIX           : VECTOR[3, BYTE],  ! Radix settings
;  131          0263  1          DBG$GL_LAST_VAL,                           ! Old last value
;  132          0264  1          DBG$GL_LAST_LOC,                           ! Old last location
;  133          0265  1          DBG$GL_NEXT_LOC,                           ! Old next location
;  134          0266  1          DBG$GL_GBLTYP,                             ! Override type
;  135          0267  1          DBG$GW_GBLLNGTH        : WORD,             ! Override length
;  136          0268  1          DBG$GL_DFLTTYP,                            ! Default type
;  137          0269  1          DBG$GW_DFLTLENG        : WORD,             ! Default length
;  138          0270  1          DBG$GB_MOD_PTR         : REF VECTOR [,BYTE], ! Mode structure
;  139          0271  1          DBG$GB_LANGUAGE        : BYTE;             ! Language setting
```

```
;  141          0272  1 GLOBAL ROUTINE DBG$NGET_RADIX =
;  142          0273  1 !++
;  143          0274  1 ! FUNCTIONAL DESCRIPTION:
;  144          0275  1 !
;  145          0276  1 !       Returns the current output radix. This is obtained by first checking
;  146          0277  1 !       whether there is a RADIX/OVERRIDE in effect, and if so, returning
;  147          0278  1 !       that. If not, return the current radix as specified by SET RADIX
;  148          0279  1 !       or by the language default.
;  149          0280  1 !
;  150          0281  1 ! FORMAL PARAMETERS:
;  151          0282  1 !
;  152          0283  1 !       NONE
;  153          0284  1 !
;  154          0285  1 ! IMPLICIT INPUTS:
;  155          0286  1 !
;  156          0287  1 !       DBG$GB_RADIX    - Global vector containing radix settings.
;  157          0288  1 !
;  158          0289  1 !--
;  159          0290  2     BEGIN
;  160          0291  2     LOCAL
;  161          0292  2         RADIX;
;  162          0293  2     RADIX = .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT_OVER];
;  163          0294  2     IF .RADIX NEQ DBG$K_DEFAULT
;  164          0295  2     THEN
;  165          0296  2         RETURN .RADIX;
;  166          0297  2     RETURN .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT];
;  167          0298  1     END;                            ! End of dbg$nget_radix


                                .TITLE  DBGNSDATA
                                .IDENT  \V04-000\

                                .PSECT  DBG$OWN,NOEXE,  PIC,2

00000000  00000 LAST_VALUE:
                                .LONG   0                                       ;
00000000  00004 LAST_LOC:
                                .LONG   0                                       ;
FFFFFFFF  00008 POTENTIAL_LAST_TYPE:
                                .LONG   -1                                      ;
00000000  0000C POTENTIAL_LAST_LENGTH:
                                .LONG   0                                       ;
FFFFFFFF  00010 LAST_TYPE:
                                .LONG   -1                                      ;
00000000  00014 LAST_LENGTH:
                                .LONG   0                                       ;

                                .EXTRN  DBG$COPY_MEMORY
                                .EXTRN  DBG$GET_TEMPMEM
                                .EXTRN  DBG$NCOPY_DESC, DBG$NFREE_DESC
                                .EXTRN  DBG$NGET_SYMID, DBG$NMAKE_ARG_VECT
                                .EXTRN  DBG$NOUT_INFO, DBG$SET_MOD_LVC
                                .EXTRN  DBG$STA_LOCK_SYMID
                                .EXTRN  DBG$STA_UNLOCK_SYMID
                                .EXTRN  DBG$REL_MEMORY, DBG$RUNFRAME
                                .EXTRN  DBG$GB_RADIX, DBG$GL_LAST_VAL
                                .EXTRN  DBG$GL_LAST_LOC
```

```
                                                .EXTRN  DBG$GL_NEXT_LOC
                                                .EXTRN  DBG$GL_GBLTYP, DBG$GW_GBLLNGTH
                                                .EXTRN  DBG$GL_DFLTYP, DBG$GW_DFLTLENG
                                                .EXTRN  DBG$GB_MOD_PTR, DBG$GB_LANGUAGE

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

                                  0000 00000    .ENTRY  DBG$NGET_RADIX, Save nothing                    ; 0272
                50 00000000G   00   9A 00002    MOVZBL  DBG$GB_RADIX+2, RADIX                           ; 0293
                01             50   D1 00009    CMPL    RADIX, #1                                       ; 0294
                               07   12 0000C    BNEQ    1$                                              ;
                50 00000000G   00   9A 0000E    MOVZBL  DBG$GB_RADIX+1, R0                              ; 0297
                                   04 00015 1$: RET                                                     ; 0298

; Routine Size:  22 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  169          0299  1 GLOBAL ROUTINE DBG$NGET_TRANS_RADIX (RADIX) =
;  170          0300  1 !++
;  171          0301  1 ! FUNCTIONAL DESCRIPTION:
;  172          0302  1 !
;  173          0303  1 !       Returns a translation of the default radix if dbg$k_default was suppplied.
;  174          0304  1 !       Otherwise, simply returns the input value.
;  175          0305  1 !
;  176          0306  1 ! FORMAL PARAMETERS:
;  177          0307  1 !
;  178          0308  1 !       RADIX   - A longword containing the radix literal.
;  179          0309  1 !
;  180          0310  1 ! IMPLICIT INPUTS:
;  181          0311  1 !
;  182          0312  1 !       NONE
;  183          0313  1 !
;  184          0314  1 ! IMPLICIT OUTPUTS:
;  185          0315  1 !
;  186          0316  1 !       NONE
;  187          0317  1 !
;  188          0318  1 ! ROUTINE VALUE:
;  189          0319  1 !
;  190          0320  1 !       An unsigned integer longword corresponding to the translated radix literal.
;  191          0321  1 !
;  192          0322  1 ! COMPLETION CODES:
;  193          0323  1 !
;  194          0324  1 !       NONE
;  195          0325  1 !
;  196          0326  1 ! SIDE EFFECTS:
;  197          0327  1 !
;  198          0328  1 !       NONE
;  199          0329  1 !
;  200          0330  1 !--
;  201          0331  2     BEGIN
;  202          0332  2     IF .radix EQL dbg$k_default
;  203          0333  2     THEN
;  204          0334  3         BEGIN
;  205          0335  3         IF .dbg$gb_language NEQ dbg$k_macro
;  206          0336  3                             AND
;  207          0337  3            .dbg$gb_language NEQ dbg$k_bliss
;  208          0338  3         THEN
;  209          0339  3             RETURN dbg$k_decimal
;  210          0340  3         ELSE
;  211          0341  3             RETURN dbg$k_hex;
;  212          0342  3         END
;  213          0343  2     ELSE
;  214          0344  3         BEGIN
;  215          0345  3         RETURN .radix;
;  216          0346  2         END;
;  217          0347  1     END;                    ! End of dbg$nget_trans_radix


                                        0000 00000          .ENTRY  DBG$NGET_TRANS_RADIX, Save nothing          ; 0299
                            01      04  AC  D1 00002         CMPL    RADIX, #1                                   ; 0332
                                        16  12 00006         BNEQ    2$                                          ;
```

```
                50 00000000G   00  9A 00008         MOVZBL  DBG$GB_LANGUAGE, R0                ; 0335
                               09  13 0000F         BEQL    1$                                 ;
                02             50  91 00011         CMPB    R0, #2                             ; 0337
                               04  13 00014         BEQL    1$                                 ;
                50             0A  D0 00016         MOVL    #10, R0                            ; 0341
                               04  00019           RET                                        ;
                50             10  D0 0001A 1$:     MOVL    #16, R0                            ;
                               04  0001D           RET                                        ; 0344
                50       04    AC  D0 0001E 2$:     MOVL    RADIX, R0                          ; 0345
                               04  00022           RET                                        ; 0347

; Routine Size:  35 bytes,    Routine Base:  DBG$CODE + 0016
```

```
;  219          0348  1 GLOBAL ROUTINE DBG$NGET_DEFAULT_TYPE (TYPE, LENGTH) : NOVALUE =
;  220          0349  1
;  221          0350  1 !++
;  222          0351  1 ! FUNCTIONAL DESCRIPTION:
;  223          0352  1 !
;  224          0353  1 !       Returns user specified default type and length.
;  225          0354  1 !
;  226          0355  1 ! FORMAL PARAMETERS:
;  227          0356  1 !
;  228          0357  1 !       TYPE    - The address of a longword to contain the default type
;  229          0358  1 !
;  230          0359  1 !       LENGTH  - The address of a longword to contain the default length
;  231          0360  1 !
;  232          0361  1 ! IMPLICIT INPUTS:
;  233          0362  1 !
;  234          0363  1 !       DBG$GL_DFLTTYP          - Contains default type
;  235          0364  1 !
;  236          0365  1 !       DBG$GW_DFLTLENG         - Contains default length
;  237          0366  1 !
;  238          0367  1 ! IMPLICIT OUTPUTS:
;  239          0368  1 !
;  240          0369  1 !       NONE
;  241          0370  1 !
;  242          0371  1 ! ROUTINE VALUE:
;  243          0372  1 !
;  244          0373  1 !       NONE
;  245          0374  1 !
;  246          0375  1 ! SIDE EFFECTS:
;  247          0376  1 !
;  248          0377  1 !       NONE
;  249          0378  1 !
;  250          0379  1 !--
;  251          0380  2     BEGIN
;  252          0381  2     .type = .dbg$gl_dflttyp;
;  253          0382  2     .length = .dbg$gw_dfltleng;
;  254          0383  2     RETURN sts$k_success;
;  255          0384  1     END;


                                                0000 00000          .ENTRY  DBG$NGET_DEFAULT_TYPE, Save nothing          ; 0348
                                    04   BC 00000000G   00   D0 00002  MOVL   DBG$GL_DFLTTYP, @TYPE                       ; 0381
                                    08   BC 00000000G   00   3C 0000A  MOVZWL DBG$GW_DFLTLENG, @LENGTH                   ; 0382
                                                        04 00012        RET                                              ; 0384

; Routine Size:  19 bytes,    Routine Base:  DBG$CODE + 0039

;  256          0385  1
```

```
;  258          0386  1 GLOBAL ROUTINE DBG$NGET_OVERRIDE_TYPE (TYPE, LENGTH) =
;  259          0387  1
;  260          0388  1 !++
;  261          0389  1 ! FUNCTIONAL DESCRIPTION:
;  262          0390  1 !
;  263          0391  1 !       Returns user specified override type and length, if set.
;  264          0392  1 !
;  265          0393  1 ! FORMAL PARAMETERS:
;  266          0394  1 !
;  267          0395  1 !       TYPE    - The address of a longword to contain the override type
;  268          0396  1 !
;  269          0397  1 !       LENGTH  - The address of a longword to contain override length
;  270          0398  1 !
;  271          0399  1 ! IMPLICIT INPUTS:
;  272          0400  1 !
;  273          0401  1 !       DBG$GL_GBLTYP   - Contains -1 (no override) or user specified override type
;  274          0402  1 !
;  275          0403  1 !       DBG$GW_GBLLNGTH - Contains override length
;  276          0404  1 !
;  277          0405  1 ! IMPLICIT OUTPUTS:
;  278          0406  1 !
;  279          0407  1 !       NONE
;  280          0408  1 !
;  281          0409  1 ! ROUTINE VALUE:
;  282          0410  1 !
;  283          0411  1 !       An unsigned integer longword completion code
;  284          0412  1 !
;  285          0413  1 ! COMPLETION CODES:
;  286          0414  1 !
;  287          0415  1 !       STS$K_SUCCESS   - If an overrride type has been set
;  288          0416  1 !
;  289          0417  1 !       STS$K_ERROR     - No override set
;  290          0418  1 !
;  291          0419  1 ! SIDE EFFECTS:
;  292          0420  1 !
;  293          0421  1 !       NONE
;  294          0422  1 !
;  295          0423  1 !--
;  296          0424  2     BEGIN
;  297          0425  2
;  298          0426  2     IF .dbg$gl_gbltyp EQL -1
;  299          0427  2     THEN
;  300          0428  2         RETURN sts$k_error;
;  301          0429  2
;  302          0430  2     .type = .dbg$gl_gbltyp;
;  303          0431  2     .length = .dbg$gw_gbllngth;
;  304          0432  2
;  305          0433  2     RETURN sts$k_success;
;  306          0434  2
;  307          0435  1     END;


                                0000 00000          .ENTRY  DBG$NGET_OVERRIDE_TYPE, Save nothing        ; 0386
                    50 00000000G  00   D0 00002     MOVL    DBG$GL_GBLTYP, R0                           ; 0426
```

```
                 FFFFFFFF   8F              50  D1 00009          CMPL    R0, #-1                                 ;
                                            04  12 00010          BNEQ    1$                                      ;
                            50              02  D0 00012          MOVL    #2, R0                                  ; 0428
                                            04  00015          RET                                             ;
                       04   BC              50  D0 00016 1$:     MOVL    R0, @TYPE                               ; 0430
                       08   BC 00000000G     00  3C 0001A          MOVZWL  DBG$GW_GBLLNGTH, @LENGTH                ; 0431
                            50              01  D0 00022          MOVL    #1, R0                                  ; 0433
                                            04  00025          RET                                             ; 0435

; Routine Size:  38 bytes,    Routine Base:  DBG$CODE + 004C

;  308          0436  1
```

```
;  310     0437  1 GLOBAL ROUTINE DBG$NGET_MODE =
;  311     0438  1
;  312     0439  1 !++
;  313     0440  1 ! FUNCTIONAL DESCRIPTION:
;  314     0441  1 !
;  315     0442  1 !       This routine returns an indication of the user specified mode: that is,
;  316     0443  1 !       TRUE if 'SYMBOLIC', FALSE if 'NOSYMBOLIC'
;  317     0444  1 !
;  318     0445  1 ! FORMAL PARAMETERS:
;  319     0446  1 !
;  320     0447  1 !       NONE
;  321     0448  1 !
;  322     0449  1 ! IMPLICIT INPUTS:
;  323     0450  1 !
;  324     0451  1 !       DBG$GB_MOD_PTR  - Pointer to current mode level
;  325     0452  1 !
;  326     0453  1 ! IMPLICIT OUTPUTS:
;  327     0454  1 !
;  328     0455  1 !       NONE
;  329     0456  1 !
;  330     0457  1 ! ROUTINE VALUE:
;  331     0458  1 !
;  332     0459  1 !       TRUE    - IF mode_symbols is true
;  333     0460  1 !
;  334     0461  1 !       FALSE   - IF mode_symbols is false
;  335     0462  1 !
;  336     0463  1 ! SIDE EFFECTS:
;  337     0464  1 !
;  338     0465  1 !       NONE
;  339     0466  1 !
;  340     0467  1 !--
;  341     0468  2     BEGIN
;  342     0469  2
;  343     0470  2     LOCAL
;  344     0471  2         SAVE_LEVEL,                                 ! Holds old pointer
;  345     0472  2         LEVEL_PTR : REF VECTOR [,BYTE];             ! mode structure
;  346     0473  2
;  347     0474  2
;  348     0475  2     ! Save the present mode level
;  349     0476  2     !
;  350     0477  2     save_level= .dbg$gb_mod_ptr;
;  351     0478  2
;  352     0479  2
;  353     0480  2     ! Set the mode pointer to the user default level
;  354     0481  2     !
;  355     0482  2     dbg$set_mod_lvl (user_def_mode);
;  356     0483  2
;  357     0484  2
;  358     0485  2     ! Pick up the new pointer
;  359     0486  2     !
;  360     0487  2     level_ptr = .dbg$gb_mod_ptr;
;  361     0488  2
;  362     0489  2
;  363     0490  2     ! Reset the mode level
;  364     0491  2     !
;  365     0492  2     dbg$gb_mod_ptr = .save_level;
;  366     0493  2
```

```
;  367          0494  2     RETURN (IF .level_ptr [mode_symbols] THEN true ELSE false);
;  368          0495  2
;  369          0496  1     END;


                                    000C 00000          .ENTRY  DBG$NGET_MODE, Save R2,R3                       ; 0437
                        53 00000000G  00   9E 00002     MOVAB   DBG$GB_MOD_PTR, R3                              ;
                        52            63   D0 00009     MOVL    DBG$GB_MOD_PTR, SAVE_LEVEL                      ; 0477
                                      01   DD 0000C     PUSHL   #1                                              ; 0482
                00000000G  00         01   FB 0000E     CALLS   #1, DBG$SET_MOD_LVL                             ;
                        50            63   D0 00015     MOVL    DBG$GB_MOD_PTR, LEVEL_PTR                       ; 0487
                        63            52   D0 00018     MOVL    SAVE_LEVEL, DBG$GB_MOD_PTR                      ; 0492
                        04      02    A0   E9 0001B     BLBC    2(LEVEL_PTR), 1$                                ; 0494
                        50            01   D0 0001F     MOVL    #1, R0                                          ;
                                           04 00022     RET                                                     ;
                                      50   D4 00023 1$:  CLRL    R0                                              ;
                                           04 00025     RET                                                     ; 0496

; Routine Size:  38 bytes,    Routine Base:  DBG$CODE + 0072
```

```
;  371     0497  1 GLOBAL ROUTINE DBG$NREPORT_LAST_VAL (VALUE_DESC, MESSAGE_VECT) =
;  372     0498  1 !++
;  373     0499  1 !
;  374     0500  1 ! FUNCTIONAL DESCRIPTION:
;  375     0501  1 !
;  376     0502  1 !       Returns the last value given to the user that was stored by the
;  377     0503  1 !       DEBUGGER.
;  378     0504  1 !
;  379     0505  1 ! FORMAL PARAMETERS:
;  380     0506  1 !
;  381     0507  1 !       value_desc      -       The address of a longword to contain the
;  382     0508  1 !                               value descriptor last created to describe
;  383     0509  1 !                               the output to the user.
;  384     0510  1 !
;  385     0511  1 !       msg_vec         -       The address of a longword to contain the
;  386     0512  1 !                               address of a message argument vector
;  387     0513  1 !                               as described on page 4-119 of the
;  388     0514  1 !                               VAX/VMS system reference, volume 1A.
;  389     0515  1 !
;  390     0516  1 ! IMPLICIT INPUTS:
;  391     0517  1 !
;  392     0518  1 !       LAST_VALUE      - Pointer to the stored last value descriptor
;  393     0519  1 !
;  394     0520  1 ! IMPLICIT OUTPUTS:
;  395     0521  1 !
;  396     0522  1 !       If successful returns a value descriptor describing the last value
;  397     0523  1 !       that the user saw. Otherwise this routine returns a message argument
;  398     0524  1 !       vector describing the error message.
;  399     0525  1 !
;  400     0526  1 ! ROUTINE VALUE:
;  401     0527  1 !
;  402     0528  1 !       An unsigned integer longword completion code.
;  403     0529  1 !
;  404     0530  1 ! COMPLETION CODES:
;  405     0531  1 !
;  406     0532  1 !       STS$K_SUCCESS   (1)     - Successful return. The value descriptor
;  407     0533  1 !                                 describing the last value that the user
;  408     0534  1 !                                 saw was returned.
;  409     0535  1 !
;  410     0536  1 !       STS$K_SEVERE    (4)     - Failure. No value descriptor could be found.
;  411     0537  1 !                                 The longword to contain the address of the
;  412     0538  1 !                                 Standard VAX Message Argument Vector is
;  413     0539  1 !                                 filled in to point to a fatal error message.
;  414     0540  1 !
;  415     0541  1 ! SIDE EFFECTS:
;  416     0542  1 !
;  417     0543  1 !       None.
;  418     0544  1 !
;  419     0545  1 !--
;  420     0546  2     BEGIN
;  421     0547  2
;  422     0548  2
;  423     0549  2     ! Check for not having a last value
;  424     0550  2     !
;  425     0551  2     IF .last_value EQLA 0
;  426     0552  2     THEN
;  427     0553  3         BEGIN
```

```
;  428          0554  3          .message_vect = dbg$nmake_arg_vect (dbg$_nolastval);
;  429          0555  3          RETURN sts$k_severe;
;  430          0556  2          END;
;  431          0557  2
;  432          0558  2
;  433          0559  2      ! Return the value descriptor
;  434          0560  2      !
;  435          0561  2      .value_desc = .last_value;
;  436          0562  2
;  437          0563  2      RETURN sts$k_success;
;  438          0564  2
;  439          0565  1      END;


                                        0004 00000          .ENTRY  DBG$NREPORT_LAST_VAL, Save R2                   ; 0497
                    52 00000000'    EF  9E 00002          MOVAB   LAST_VALUE, R2                                  ;
                                    62  D5 00009          TSTL    LAST_VALUE                                      ; 0551
                                    15  12 0000B          BNEQ    1$                                              ;
                       000287E0     8F  DD 0000D          PUSHL   #165856                                         ; 0554
                00000000G   00      01  FB 00013          CALLS   #1, DBG$NMAKE_ARG_VECT                          ;
                      08    BC      50  D0 0001A          MOVL    R0, @MESSAGE_VECT                               ;
                            50      04  D0 0001E          MOVL    #4, R0                                          ; 0555
                                        04 00021          RET                                                     ;
                      04    BC      62  D0 00022 1$:      MOVL    LAST_VALUE, @VALUE_DESC                         ; 0561
                            50      01  D0 00026          MOVL    #1, R0                                          ; 0563
                                        04 00029          RET                                                     ; 0565

; Routine Size:  42 bytes,    Routine Base:  DBG$CODE + 0098

;  440          0566  1
```

```
;  442          0567  1 GLOBAL ROUTINE DBG$NREPORT_LAST_LOC( ADDR_EXP_DESC, TYPE, LENGTH, MESSAGE_VECT) =
;  443          0568  1
;  444          0569  1 !++
;  445          0570  1 !
;  446          0571  1 ! FUNCTIONAL DESCRIPTION:
;  447          0572  1 !
;  448          0573  1 !       Returns the last location that was returned shown to the user
;  449          0574  1 !       by the DEBUGGER.
;  450          0575  1 !
;  451          0576  1 ! FORMAL PARAMETERS:
;  452          0577  1 !
;  453          0578  1 !       add_exp_desc    -       The address of a longword to contain
;  454          0579  1 !                               the address of an Address Expression
;  455          0580  1 !                               Descriptor.
;  456          0581  1 !
;  457          0582  1 !       type            -       The address of a longword to contain
;  458          0583  1 !                               the type of the last location shown
;  459          0584  1 !                               to the user. This type may be one
;  460          0585  1 !                               of the following types:
;  461          0586  1 !
;  462          0587  1 !                               dsc$k_dtype_b, dsc$k_dtype_w, dsc$k_dtype_l,
;  463          0588  1 !                               dsc$k_dtype_lu, dsc$k_dtype_t, dsc$k_dtype_zi
;  464          0589  1 !
;  465          0590  1 !       length          -       The address of a longword to contain
;  466          0591  1 !                               the length of the object output to the
;  467          0592  1 !                               user.
;  468          0593  1 !
;  469          0594  1 !       msg_vec         -       The address of a longword to contain
;  470          0595  1 !                               the address of a standard VAX Message
;  471          0596  1 !                               Argument Vector as described on page
;  472          0597  1 !                               4-119 of the VAX/VMS system reference
;  473          0598  1 !                               manual, volume 1A.
;  474          0599  1 !
;  475          0600  1 ! IMPLICIT INPUTS:
;  476          0601  1 !
;  477          0602  1 !       LAST_LOC        - Pointer to the stored address expression descriptor
;  478          0603  1 !
;  479          0604  1 !       LAST_TYPE       - The type in which current loc was last displayed.
;  480          0605  1 !
;  481          0606  1 !       LAST_LENGTH     - The last length used for curren loc.
;  482          0607  1 !
;  483          0608  1 ! IMPLICIT OUTPUTS:
;  484          0609  1 !
;  485          0610  1 !       If successful, this routine returns an Address Expression Descriptor
;  486          0611  1 !       describing the last location that the user looked at and the associated
;  487          0612  1 !       type and length of the object previously looked at by the user in the
;  488          0613  1 !       DEBUGGER.
;  489          0614  1 !
;  490          0615  1 !       If unsuccessful, the longword containing the address of the
;  491          0616  1 !       Standard VAX Message Argument Vector for that buffer is updated
;  492          0617  1 !       to contain the address of the error message buffer.
;  493          0618  1 !
;  494          0619  1 ! ROUTINE VALUE:
;  495          0620  1 !
;  496          0621  1 !       STS$K_SUCCESS   (1)     - An Address Expression Descriptor describing
;  497          0622  1 !                                 the last location looked at by the user was
;  498          0623  1 !                                 found and is returned along with the
```

```
;  499          0624  1 !                                  associated type and length of the object that
;  500          0625  1 !                                  was displayed.
;  501          0626  1 !
;  502          0627  1 !       STS$K_SEVERE    (4)      - Failure. No Address Expression Descriptor
;  503          0628  1 !                                  describing the last location that the user
;  504          0629  1 !                                  looked at could be found. The longword
;  505          0630  1 !                                  containing the address of the Standard VAX
;  506          0631  1 !                                  Message Argument Vector is filled in to
;  507          0632  1 !                                  point to a fatal error message.
;  508          0633  1 !
;  509          0634  1 ! SIDE EFFECTS:
;  510          0635  1 !
;  511          0636  1 !       None.
;  512          0637  1 !
;  513          0638  1 !--
;  514          0639  1
;  515          0640  1
;  516          0641  2     BEGIN
;  517          0642  2
;  518          0643  2
;  519          0644  2     ! Check for no current location
;  520          0645  2     !
;  521          0646  2     IF .last_loc EQLA 0
;  522          0647  2     THEN
;  523          0648  3         BEGIN
;  524          0649  3         .message_vect = dbg$nmake_arg_vect (dbg$_nocurloc);
;  525          0650  3         RETURN sts$k_severe;
;  526          0651  2         END;
;  527          0652  2
;  528          0653  2
;  529          0654  2     ! Check to see if we have a permanent symbol descriptor. If we do, then
;  530          0655  2     ! we must get a new interpretation of the register
;  531          0656  2     !
;  532          0657  2     IF .last_loc [dbg$b_aed_type] EQL dbg$k_perm_desc
;  533          0658  2     THEN
;  534          0659  3         BEGIN
;  535          0660  3         LOCAL
;  536          0661  3             PSD_PTR,
;  537          0662  3             PERM_DESC : REF dbg$permsd;
;  538          0663  3
;  539          0664  3         perm_desc = .last_loc [dbg$l_aed_value];
;  540          0665  3
;  541          0666  3         IF NOT dbg$nperm_sym_int (.perm_desc [dbg$b_permsd_id], psd_ptr, 0, .message_vect)
;  542          0667  3         THEN
;  543          0668  3             RETURN sts$k_severe;
;  544          0669  3
;  545          0670  3
;  546          0671  3         ! Release the old permanent symbol descriptor and set up the address
;  547          0672  3         ! expression descriptor to point to the new one.
;  548          0673  3         !
;  549          0674  3         dbg$rel_memory (.perm_desc);
;  550          0675  3
;  551          0676  3
;  552          0677  3         ! Make a permanent copy of the new permanent symbol descriptor
;  553          0678  3         !
;  554          0679  3         perm_desc = dbg$copy_memory (.psd_ptr);
;  555          0680  3         last_loc [dbg$l_aed_value] = .perm_desc;
```

```
;  556          0681  2          END;
;  557          0682  2
;  558          0683  2
;  559          0684  2      ! Return the requested information
;  560          0685  2      !
;  561          0686  2      .addr_exp_desc = .last_loc;
;  562          0687  2      .type = .last_type;
;  563          0688  2      .length = .last_length;
;  564          0689  2
;  565          0690  2      RETURN sts$k_success;
;  566          0691  2
;  567          0692  1      END;        ! End of dbg$nreport_last_loc


                                    000C 00000          .ENTRY  DBG$NREPORT_LAST_LOC, Save R2,R3        ; 0567
                53 00000000'    EF   9E 00002          MOVAB   LAST_LOC, R3                            ;
                5E              04   C2 00009          SUBL2   #4, SP                                  ;
                                63   D5 0000C          TSTL    LAST_LOC                                ; 0646
                                13   12 0000E          BNEQ    1$                                      ;
                   00028808     8F   DD 00010          PUSHL   #165896                                 ; 0649
00000000G       00              01   FB 00016          CALLS   #1, DBG$NMAKE_ARG_VECT                  ;
      10        BC              50   D0 0001D          MOVL    R0, @MESSAGE_VECT                       ;
                                20   11 00021          BRB     2$                                      ; 0650
                50              63   D0 00023 1$:      MOVL    LAST_LOC, R0                            ; 0657
      7B        8F              60   91 00026          CMPB    (R0), #123                              ;
                                37   12 0002A          BNEQ    4$                                      ;
                52        04    A0   D0 0002C          MOVL    4(R0), PERM_DESC                        ; 0664
                          10    AC   DD 00030          PUSHL   MESSAGE_VECT                            ; 0666
                                7E   D4 00033          CLRL    -(SP)                                   ;
                          08    AE   9F 00035          PUSHAB  PSD_PTR                                 ;
                7E              62   9A 00038          MOVZBL  (PERM_DESC), -(SP)                      ;
          0000V CF              04   FB 0003B          CALLS   #4, DBG$NPERM_SYM_INT                   ;
                04              50   E8 00040          BLBS    R0, 3$                                  ;
                50              04   D0 00043 2$:      MOVL    #4, R0                                  ; 0668
                                     04 00046          RET                                             ;
                                52   DD 00047 3$:      PUSHL   PERM_DESC                               ; 0674
00000000G       00              01   FB 00049          CALLS   #1, DBG$REL_MEMORY                      ;
                                6E   DD 00050          PUSHL   PSD_PTR                                 ; 0679
00000000G       00              01   FB 00052          CALLS   #1, DBG$COPY_MEMORY                     ;
                52              50   D0 00059          MOVL    R0, PERM_DESC                           ;
                50              63   D0 0005C          MOVL    LAST_LOC, R0                            ; 0680
          04    A0              52   D0 0005F          MOVL    PERM_DESC, 4(R0)                        ;
          04    BC              63   D0 00063 4$:      MOVL    LAST_LOC, @ADDR_EXP_DESC                ; 0686
          08    BC        0C    A3   D0 00067          MOVL    LAST_TYPE, @TYPE                        ; 0687
          0C    BC        10    A3   D0 0006C          MOVL    LAST_LENGTH, @LENGTH                    ; 0688
                50              01   D0 00071          MOVL    #1, R0                                  ; 0690
                                     04 00074          RET                                             ; 0692

; Routine Size:  117 bytes,    Routine Base:  DBG$CODE + 00C2

;  568          0693  1
```

```
;  570          0694  1 GLOBAL ROUTINE DBG$NSET_LAST_TYPLEN (TYPE, LENGTH) : NOVALUE =
;  571          0695  1
;  572          0696  1 !++
;  573          0697  1 ! FUNCTIONAL DESCRIPTION:
;  574          0698  1 !
;  575          0699  1 !       Stores the values of the input parameters for possible use of typing
;  576          0700  1 !       current location. The type for curren loc is returned by the AEI
;  577          0701  1 !
;  578          0702  1 ! FORMAL PARAMETERS:
;  579          0703  1 !
;  580          0704  1 !       TYPE    -       A longword containing the value of a vax standard type.
;  581          0705  1 !
;  582          0706  1 !       LENGTH  -       A longword containing the length in bytes to be stored.
;  583          0707  1 !
;  584          0708  1 ! IMPLICIT INPUTS:
;  585          0709  1 !
;  586          0710  1 !       POTENTIAL_LAST_TYPE     - A longword to contain the value of type
;  587          0711  1 !
;  588          0712  1 !       POTENTIAL_LAST_LENGTH   - A longword to contain the length
;  589          0713  1 !
;  590          0714  1 ! IMPLICIT OUTPUTS:
;  591          0715  1 !
;  592          0716  1 !       NONE
;  593          0717  1 !
;  594          0718  1 ! ROUTINE VALUE:
;  595          0719  1 !
;  596          0720  1 !       NOVALUE
;  597          0721  1 !
;  598          0722  1 ! COMPLETION CODES:
;  599          0723  1 !
;  600          0724  1 !       NONE
;  601          0725  1 !
;  602          0726  1 ! SIDE EFFECTS:
;  603          0727  1 !
;  604          0728  1 !       NONE
;  605          0729  1 !
;  606          0730  1 !--
;  607          0731  2     BEGIN
;  608          0732  2
;  609          0733  2
;  610          0734  2     ! Just tuck away the information
;  611          0735  2     !
;  612          0736  2     potential_last_type = .type;
;  613          0737  2     potential_last_length = .length;
;  614          0738  2
;  615          0739  2     RETURN;
;  616          0740  2
;  617          0741  1     END;        ! End of dbg$nsave_last_typlen


                                        0000 00000          .ENTRY  DBG$NSET_LAST_TYPLEN, Save nothing          ; 0694
                00000000'  EF       04  AC  7D 00002        MOVQ    TYPE, POTENTIAL_LAST_TYPE                   ; 0736
                                        04 0000A            RET                                                 ; 0741
```

```
; Routine Size:  11 bytes,    Routine Base:  DBG$CODE + 0137

;  618          0742  1
```

```
;  620          0743  1 GLOBAL ROUTINE DBG$NGET_POTENTIAL_TYPE (TYPE, LENGTH) : NOVALUE =
;  621          0744  1
;  622          0745  1 !++
;  623          0746  1 ! FUNCTIONAL DESCRIPTION:
;  624          0747  1 !
;  625          0748  1 !       This routine returns the AEI set type to be associated with current location
;  626          0749  1 !       in lieu of any overrides. This information is used to type raw addresses.
;  627          0750  1 !
;  628          0751  1 ! FORMAL PARAMETERS:
;  629          0752  1 !
;  630          0753  1 !       TYPE    - The address of a longword to contain type information
;  631          0754  1 !
;  632          0755  1 !       LENGTH  - The address of a longword to contain the length (in bytes)
;  633          0756  1 !
;  634          0757  1 ! IMPLICIT INPUTS:
;  635          0758  1 !
;  636          0759  1 !       potential_last_type     - a longword containing the type set by AEI
;  637          0760  1 !
;  638          0761  1 !       potential_last_length - a longword containing the length set by AEI
;  639          0762  1 !
;  640          0763  1 ! IMPLICIT OUTPUTS:
;  641          0764  1 !
;  642          0765  1 !       NONE
;  643          0766  1 !
;  644          0767  1 ! ROUTINE VALUE:
;  645          0768  1 !
;  646          0769  1 !       NOVALUE
;  647          0770  1 !
;  648          0771  1 ! COMPLETION CODES:
;  649          0772  1 !
;  650          0773  1 !       NONE
;  651          0774  1 !
;  652          0775  1 ! SIDE EFFECTS:
;  653          0776  1 !
;  654          0777  1 !       NONE
;  655          0778  1 !
;  656          0779  1 !--
;  657          0780  2     BEGIN
;  658          0781  2
;  659          0782  2
;  660          0783  2     ! Just return the requested information
;  661          0784  2     !
;  662          0785  2     .type = .potential_last_type;
;  663          0786  2     .length = .potential_last_length;
;  664          0787  2
;  665          0788  2     RETURN;
;  666          0789  2
;  667          0790  1     END;        ! End of dbg$nget_potential_type


                                        0000 00000          .ENTRY  DBG$NGET_POTENTIAL_TYPE, Save nothing       ; 0743
                04   BC 00000000'  EF   D0 00002          MOVL    POTENTIAL_LAST_TYPE, @TYPE                  ; 0785
                08   BC 00000000'  EF   D0 0000A          MOVL    POTENTIAL_LAST_LENGTH, @LENGTH              ; 0786
                                        04 00012          RET                                                 ; 0790
```

```
; Routine Size:  19 bytes,    Routine Base:  DBG$CODE + 0142

;  668          0791  1
```

```
;  670          0792  1 GLOBAL ROUTINE DBG$NSAVE_LAST_VAL (VALUE_DESC) : NOVALUE =
;  671          0793  1
;  672          0794  1 !++
;  673          0795  1 ! FUNCTIONAL DESCRIPTION:
;  674          0796  1 !
;  675          0797  1 !       Saves the input parameter as the interpretation of
;  676          0798  1 !       last value. A copy of the value descriptor is constructed from non-listed
;  677          0799  1 !       storage.
;  678          0800  1 !
;  679          0801  1 !       Value descriptors may be volatile or non-volatile. This routine will
;  680          0802  1 !       store only non-volatile value descriptors.
;  681          0803  1 !
;  682          0804  1 ! FORMAL PARAMETERS:
;  683          0805  1 !
;  684          0806  1 !       VALUE_DESC      - A longword containing the address of a value descriptor
;  685          0807  1 !                         to be stored for last val. If 0, then last val is
;  686          0808  1 !                         canceled.
;  687          0809  1 !
;  688          0810  1 ! IMPLICIT INPUTS:
;  689          0811  1 !
;  690          0812  1 !       LAST_VAL        - A longword to contain the address of the stored value
;  691          0813  1 !                         descriptor
;  692          0814  1 !
;  693          0815  1 ! IMPLICIT OUTPUTS:
;  694          0816  1 !
;  695          0817  1 !       NONE
;  696          0818  1 !
;  697          0819  1 ! ROUTINE VALUE:
;  698          0820  1 !
;  699          0821  1 !       NOVALUE
;  700          0822  1 !
;  701          0823  1 ! COMPLETION CODES:
;  702          0824  1 !
;  703          0825  1 !       NONE
;  704          0826  1 !
;  705          0827  1 ! SIDE EFFECTS:
;  706          0828  1 !
;  707          0829  1 !       Redefines last value.
;  708          0830  1 !
;  709          0831  1 !--
;  710          0832  2     BEGIN
;  711          0833  2
;  712          0834  2     MAP VALUE_DESC : REF dbg$dhead;
;  713          0835  2
;  714          0836  2     LOCAL
;  715          0837  2         DUMMY,                      ! Dummy parameter
;  716          0838  2         OLD_SYMID_LIST,             ! Symid list for old '\'
;  717          0839  2         NEW_SYMID_LIST,             ! Symid list for new '\'
;  718          0840  2         MESSAGE_VECT,               ! Holds address of message argument vector
;  719          0841  2         VALUE_PTR;                  ! Temporary pointer to value descriptor.
;  720          0842  2
;  721          0843  2     old_symid_list = 0;
;  722          0844  2     new_symid_list = 0;
;  723          0845  2     value_ptr = 0;
;  724          0846  2
;  725          0847  2
;  726          0848  2     ! Check for non-null input
```

```
;  727          0849  2     !
;  728          0850  2     IF .value_desc NEQA 0
;  729          0851  2     THEN
;  730          0852  3         BEGIN
;  731          0853  3
;  732          0854  3         ! Check for a volatile value descriptor.
;  733          0855  3         !
;  734          0856  3         IF .value_desc [dbg$b_dhead_type] EQL dbg$k_v_value_desc
;  735          0857  3         THEN
;  736          0858  4             BEGIN
;  737          0859  4             dbg$nout_info (dbg$_nonewval);
;  738          0860  4             END
;  739          0861  3         ELSE
;  740          0862  4             BEGIN
;  741          0863  4
;  742          0864  4             ! We have a non-volatile value descriptor. Obtain the new symid_list.
;  743          0865  4             !
;  744          0866  4             IF NOT dbg$nget_symid (.value_desc, new_symid_list, dummy)
;  745          0867  4             THEN
;  746          0868  5                 BEGIN
;  747          0869  5
;  748          0870  5                 ! Once again, can't save new '\'.
;  749          0871  5                 !
;  750          0872  5                 dbg$nout_info (dbg$_nonewval);
;  751          0873  5                 END
;  752          0874  5
;  753          0875  4             ELSE
;  754          0876  5                 BEGIN
;  755          0877  5
;  756          0878  5                 ! Try to copy the value descriptor
;  757          0879  5                 !
;  758          0880  5                 IF NOT dbg$ncopy_desc (.value_desc, value_ptr, dummy)
;  759          0881  5                 THEN
;  760          0882  6                     BEGIN
;  761          0883  6
;  762          0884  6                     ! Can't save '\'
;  763          0885  6                     !
;  764          0886  6                     dbg$nout_info (dbg$_nonewval);
;  765          0887  6                     value_ptr = 0;
;  766          0888  6                     END
;  767          0889  5                 ELSE
;  768          0890  6                     BEGIN
;  769          0891  6
;  770          0892  6                     ! The descriptor has been copied. Lock the symids.
;  771          0893  6                     !
;  772          0894  6                     dbg$sta_lock_symid (.new_symid_list);
;  773          0895  5                     END;
;  774          0896  4                 END;
;  775          0897  3             END;
;  776          0898  2         END;
;  777          0899  2
;  778          0900  2
;  779          0901  2     ! The new descriptor has been copied and saved. Delete the old descriptor and
;  780          0902  2     ! unlock symids
;  781          0903  2     !
;  782          0904  2     IF .last_value NEQA 0
;  783          0905  2     THEN
```

```
;  784          0906  3          BEGIN
;  785          0907  3
;  786          0908  3          ! Obtain the symids for '\' and unlock them.
;  787          0909  3          !
;  788          0910  3          IF dbg$nget_symid (.last_value, old_symid_list, dummy)
;  789          0911  3          THEN
;  790          0912  3              dbg$sta_unlock_symid (.old_symid_list);
;  791          0913  3
;  792          0914  3
;  793          0915  3          ! Release the storage
;  794          0916  3          !
;  795          0917  3          dbg$nfree_desc (.last_value, dummy);
;  796          0918  2          END;
;  797          0919  2
;  798          0920  2      last_value = .value_ptr;
;  799          0921  2
;  800          0922  2      RETURN;
;  801          0923  2
;  802          0924  1      END;                ! End of dbg$nsave_last_val


                                    003C 00000          .ENTRY  DBG$NSAVE_LAST_VAL, Save R2,R3,R4,R5        ; 0792
                55 00000000G   00   9E 00002          MOVAB   DBG$NGET_SYMID, R5                          ;
                54 00000000G   00   9E 00009          MOVAB   DBG$NOUT_INFO, R4                           ;
                53 00000000'   EF   9E 00010          MOVAB   LAST_VALUE, R3                              ;
                5E                 0C   C2 00017          SUBL2   #12, SP                                     ;
                         04        AE   D4 0001A          CLRL    OLD_SYMID_LIST                              ; 0843
                                   7E   D4 0001D          CLRL    NEW_SYMID_LIST                              ; 0844
                         04        AE   D4 0001F          CLRL    VALUE_PTR                                   ; 0845
                      52 04        AC   D0 00022          MOVL    VALUE_DESC, R2                              ; 0850
                                   49   13 00026          BEQL    4$                                          ;
                   83  8F 02        A2   91 00028          CMPB    2(R2), #131                                 ; 0856
                                   0E   13 0002D          BEQL    1$                                          ;
                         0C        AE   9F 0002F          PUSHAB  DUMMY                                       ; 0866
                         04        AE   9F 00032          PUSHAB  NEW_SYMID_LIST                              ;
                                   52   DD 00035          PUSHL   R2                                          ;
                      65           03   FB 00037          CALLS   #3, DBG$NGET_SYMID                          ;
                      0B           50   E8 0003A          BLBS    R0, 2$                                      ;
                         0002865B  8F   DD 0003D 1$:      PUSHL   #165467                                     ; 0872
                      64           01   FB 00043          CALLS   #1, DBG$NOUT_INFO                           ;
                                   29   11 00046          BRB     4$                                          ; 0866
                         0C        AE   9F 00048 2$:      PUSHAB  DUMMY                                       ; 0880
                         08        AE   9F 0004B          PUSHAB  VALUE_PTR                                   ;
                                   52   DD 0004E          PUSHL   R2                                          ;
00000000G             00           03   FB 00050          CALLS   #3, DBG$NCOPY_DESC                          ;
                      0E           50   E8 00057          BLBS    R0, 3$                                      ;
                         0002865B  8F   DD 0005A          PUSHL   #165467                                     ; 0886
                      64           01   FB 00060          CALLS   #1, DBG$NOUT_INFO                           ;
                         04        AE   D4 00063          CLRL    VALUE_PTR                                   ; 0887
                                   09   11 00066          BRB     4$                                          ; 0880
                                   6E   DD 00068 3$:      PUSHL   NEW_SYMID_LIST                              ; 0894
00000000G             00           01   FB 0006A          CALLS   #1, DBG$STA_LOCK_SYMID                      ;
                      50           63   D0 00071 4$:      MOVL    LAST_VALUE, R0                              ; 0904
                                   24   13 00074          BEQL    6$                                          ;
```

```
                        0C   AE   9F 00076          PUSHAB  DUMMY                           ; 0910
                        0C   AE   9F 00079          PUSHAB  OLD_SYMID_LIST                  ;
                             50   DD 0007C          PUSHL   R0                              ;
                       65    03   FB 0007E          CALLS   #3, DBG$NGET_SYMID              ;
                       0A    50   E9 00081          BLBC    R0, 5$                          ;
                        08   AE   DD 00084          PUSHL   OLD_SYMID_LIST                  ; 0912
             00000000G 00    01   FB 00087          CALLS   #1, DBG$STA_UNLOCK_SYMID        ;
                        0C   AE   9F 0008E 5$:      PUSHAB  DUMMY                           ; 0917
                             63   DD 00091          PUSHL   LAST_VALUE                      ;
             00000000G 00    02   FB 00093          CALLS   #2, DBG$NFREE_DESC              ;
                       63    04   AE   D0 0009A 6$: MOVL    VALUE_PTR, LAST_VALUE          ; 0920
                                  04 0009E          RET                                     ; 0924

; Routine Size:  159 bytes,    Routine Base:  DBG$CODE + 0155

;  803          0925  1
```

```
;  805     0926  1 GLOBAL ROUTINE DBG$NSAVE_LAST_LOC (ADDR_EXP_DESC, OVERRIDE_TYPE, OVERRIDE_LENGTH) : NOVALUE =
;  806     0927  1
;  807     0928  1 !++
;  808     0929  1 ! FUNCTIONAL DESCRIPTION:
;  809     0930  1 !
;  810     0931  1 !       Makes an unlisted copy of an address expression descriptor to store as
;  811     0932  1 !       the interpretation of current location. If override type and length are
;  812     0933  1 !       supplied, current location is typed accordingly.
;  813     0934  1 !
;  814     0935  1 ! FORMAL PARAMETERS:
;  815     0936  1 !
;  816     0937  1 !       ADDR_EXP_DESC   - A longword containing the address of an address expression
;  817     0938  1 !                         descriptor. If 0, then current loc is canceled.
;  818     0939  1 !
;  819     0940  1 !       OVERRIDE_TYPE   - A longword containing a vax standard type with which to
;  820     0941  1 !                         type current location.
;  821     0942  1 !
;  822     0943  1 !       OVERRIDE_LENGTH - A longword containg an integer representing the override
;  823     0944  1 !                         length to be associated with current location.
;  824     0945  1 !
;  825     0946  1 ! IMPLICIT INPUTS:
;  826     0947  1 !
;  827     0948  1 !       LAST_LOC        - A longword to contain the address of an addr exp desc
;  828     0949  1 !                         for current location.
;  829     0950  1 !
;  830     0951  1 !       LAST_TYPE       - A longword to contain the override type for current loc
;  831     0952  1 !
;  832     0953  1 !       POTENTIAL_LAST_TYPE - A longword containing a possible type for current loc.
;  833     0954  1 !
;  834     0955  1 !       LAST_LENGTH     - A longword to contain the override length for current loc
;  835     0956  1 !
;  836     0957  1 !       POTENTIAL_LAST_LENGTH - A longword containing a potential last length for curren loc
;  837     0958  1 !
;  838     0959  1 ! IMPLICIT OUTPUTS:
;  839     0960  1 !
;  840     0961  1 !       NONE
;  841     0962  1 !
;  842     0963  1 ! ROUTINE VALUE:
;  843     0964  1 !
;  844     0965  1 !       NOVALUE
;  845     0966  1 !
;  846     0967  1 ! COMPLETION CODES:
;  847     0968  1 !
;  848     0969  1 !       NONE
;  849     0970  1 !
;  850     0971  1 ! SIDE EFFECTS:
;  851     0972  1 !
;  852     0973  1 !       Redefines current location.
;  853     0974  1 !
;  854     0975  1 !--
;  855     0976  2     BEGIN
;  856     0977  2
;  857     0978  2     MAP
;  858     0979  2         ADDR_EXP_DESC       : REF dbg$aed;
;  859     0980  2
;  860     0981  2     LOCAL
;  861     0982  2         DUMMY,                                  ! Dummy parameter
```

```
;  862          0983  2          OLD_SYMID_LIST,                         ! Pointer to old symid list
;  863          0984  2          NEW_SYMID_LIST,                         ! Pointer to new symid list
;  864          0985  2          SAVE_ADDR_EXP_DESC : REF dbg$aed;       ! The address expression descriptor to save
;  865          0986  2
;  866          0987  2      old_symid_list = 0;
;  867          0988  2      new_symid_list = 0;
;  868          0989  2      save_addr_exp_desc = 0;
;  869          0990  2
;  870          0991  2
;  871          0992  2      ! Check for null input
;  872          0993  2      !
;  873          0994  2      IF .addr_exp_desc NEQA 0
;  874          0995  2      THEN
;  875          0996  3          BEGIN
;  876          0997  3
;  877          0998  3          ! When this routine is called, an address expression descriptor is supplied
;  878          0999  3          ! to become the definition of current loc. If a type and length (not EQL -1)
;  879          1000  3          ! are supplied, we type current loc according to these parameters. Otherwise,
;  880          1001  3          ! we use potential_type and potential_loc to type current loc.
;  881          1002  3          !
;  882          1003  3          IF .override_type EQL -1
;  883          1004  3                      OR
;  884          1005  3             .override_length EQL -1
;  885          1006  3          THEN
;  886          1007  4              BEGIN
;  887          1008  4              last_type = .potential_last_type;
;  888          1009  4              last_length = .potential_last_length;
;  889          1010  4              END
;  890          1011  3          ELSE
;  891          1012  4              BEGIN
;  892          1013  4              last_type = .override_type;
;  893          1014  4              last_length = .override_length;
;  894          1015  3              END;
;  895          1016  3
;  896          1017  3
;  897          1018  3          ! Current location has been typed. Now a copy of the address expression
;  898          1019  3          ! descriptor, and any descriptors it points to must be made. First make a
;  899          1020  3          ! copy of the address expression descriptor.
;  900          1021  3          !
;  901          1022  3          save_addr_exp_desc = dbg$copy_memory(.addr_exp_desc);
;  902          1023  3
;  903          1024  3
;  904          1025  3          ! Make copies of any descriptors pointed to by the address expression descriptor
;  905          1026  3          !
;  906          1027  3          CASE .addr_exp_desc [dbg$b_aed_type] FROM dbg$k_primary_desc TO dbg$k_notype
;  907          1028  3              OF
;  908          1029  3              SET
;  909          1030  3
;  910          1031  3              [dbg$k_primary_desc] :
;  911          1032  4                  BEGIN
;  912          1033  4                  LOCAL
;  913          1034  4                      NEW_PRIM : REF dbg$dhead;
;  914          1035  4
;  915          1036  4
;  916          1037  4                  ! Have to copy the primary descriptor and notify the symbol table that
;  917          1038  4                  ! we're hanging on to a symid.
;  918          1039  4                  !
```

```
;  919          1040  4          IF NOT dbg$nget_symid (.addr_exp_desc [dbg$l_aed_value], new_symid_list, dummy)
;  920          1041  4          THEN
;  921          1042  5              BEGIN
;  922          1043  5
;  923          1044  5              ! Can't save new '.'
;  924          1045  5              !
;  925          1046  5              dbg$nout_info (dbg$_nonewcur);
;  926          1047  5              dbg$rel_memory (.save_addr_exp_desc);
;  927          1048  5              save_addr_exp_desc = 0;
;  928          1049  5              END
;  929          1050  5
;  930          1051  4          ELSE
;  931          1052  5              BEGIN
;  932          1053  5              IF NOT dbg$ncopy_desc (.addr_exp_desc [dbg$l_aed_value], new_prim, dummy)
;  933          1054  5              THEN
;  934          1055  6                  BEGIN
;  935          1056  6                  dbg$nout_info (dbg$_nonewcur);
;  936          1057  6                  dbg$rel_memory (.save_addr_exp_desc);
;  937          1058  6                  save_addr_exp_desc = 0;
;  938          1059  6                  END
;  939          1060  5              ELSE
;  940          1061  6                  BEGIN
;  941          1062  6
;  942          1063  6                  ! Notify the symbol table to hang on to the new symids
;  943          1064  6                  !
;  944          1065  6                  dbg$sta_lock_symid (.new_symid_list);
;  945          1066  6
;  946          1067  6
;  947          1068  6                  ! point the address expression to the permanent primary descriptor
;  948          1069  6                  !
;  949          1070  6                  save_addr_exp_desc [dbg$l_aed_value] = .new_prim;
;  950          1071  5                  END;
;  951          1072  4              END;
;  952          1073  3              END;
;  953          1074  3
;  954          1075  3              [dbg$k_perm_desc] :
;  955          1076  4                  BEGIN
;  956          1077  4
;  957          1078  4                  ! Have to make a non-listed copy of the permanent symbol descriptor.
;  958          1079  4                  !
;  959          1080  4                  save_addr_exp_desc [dbg$l_aed_value] =
;  960          1081  4                          dbg$copy_memory(.addr_exp_desc [dbg$l_aed_value]);
;  961          1082  3                  END;
;  962          1083  3
;  963          1084  3              [INRANGE,OUTRANGE] :
;  964          1085  4                  BEGIN
;  965          1086  4
;  966          1087  4                  ! Not pointing to any descriptors. All the work has been done.
;  967          1088  4                  !
;  968          1089  4                  0;
;  969          1090  4
;  970          1091  3                  END;
;  971          1092  3
;  972          1093  3              TES;
;  973          1094  3          END
;  974          1095  2      ELSE
;  975          1096  3          BEGIN
```

```
;  976          1097  3          save_addr_exp_desc = 0;
;  977          1098  2          END;
;  978          1099  2
;  979          1100  2
;  980          1101  2      ! Release storage and symids for the old address expression descriptor
;  981          1102  2      !
;  982          1103  2      IF .last_loc NEQA 0
;  983          1104  2      THEN
;  984          1105  3          BEGIN
;  985          1106  3          CASE .last_loc [dbg$b_aed_type] FROM dbg$k_primary_desc TO dbg$k_notype
;  986          1107  3              OF
;  987          1108  3              SET
;  988          1109  3
;  989          1110  3              [dbg$k_primary_desc] :
;  990          1111  4                  BEGIN
;  991          1112  4                  LOCAL
;  992          1113  4                      OLD_PRIM_DESC : REF dbg$dhead;
;  993          1114  4
;  994          1115  4                  old_prim_desc = .last_loc [dbg$l_aed_value];  ! Get the primary desc
;  995          1116  4
;  996          1117  4
;  997          1118  4                  ! Extract the symids
;  998          1119  4                  !
;  999          1120  4                  IF dbg$nget_symid (.old_prim_desc, old_symid_list, dummy)
; 1000          1121  4                  THEN
; 1001          1122  4                      dbg$sta_unlock_symid (.old_symid_list);
; 1002          1123  4
; 1003          1124  4                  ! Release the storage for the primary descriptor
; 1004          1125  4                  !
; 1005          1126  4                  dbg$nfree_desc (.old_prim_desc, dummy);
; 1006          1127  4
; 1007          1128  3                  END;
; 1008          1129  3
; 1009          1130  3              [dbg$k_perm_desc] :
; 1010          1131  4                  BEGIN
; 1011          1132  4
; 1012          1133  4                  ! Just release the storage for the permanent symbol descriptor
; 1013          1134  4                  !
; 1014          1135  4                  dbg$rel_memory (.last_loc [dbg$l_aed_value]);
; 1015          1136  4
; 1016          1137  3                  END;
; 1017          1138  3
; 1018          1139  3              [INRANGE,OUTRANGE] :
; 1019          1140  4                  BEGIN
; 1020          1141  4
; 1021          1142  4                  ! No extra storage to release
; 1022          1143  4                  !
; 1023          1144  4                  0;
; 1024          1145  4
; 1025          1146  3                  END;
; 1026          1147  3
; 1027          1148  3              TES;
; 1028          1149  3
; 1029          1150  3
; 1030          1151  3      ! Release the storage for the address expression descriptor itself and
; 1031          1152  3      ! set current loc to nil
; 1032          1153  3      !
```

```
;  1033          1154  3          dbg$rel_memory (.last_loc);
;  1034          1155  3
;  1035          1156  2          END;
;  1036          1157  2
;  1037          1158  2
;  1038          1159  2      ! Save the new address expression descriptor
;  1039          1160  2      !
;  1040          1161  2      IF .save_addr_exp_desc NEQA 0
;  1041          1162  2      THEN
;  1042          1163  2          last_loc = .save_addr_exp_desc
;  1043          1164  2      ELSE
;  1044          1165  2          last_loc = 0;
;  1045          1166  2
;  1046          1167  2      RETURNR;
;  1047          1168  2
;  1048          1169  1      END;        ! End of dbg$nsave_last_loc


                                                00FC 00000          .ENTRY  DBG$NSAVE_LAST_LOC, Save R2,R3,R4,R5,R6,R7  ; 0926
                                    57 00000000G  00   9E 00002     MOVAB   DBG$NGET_SYMID, R7                          ;
                                    56 00000000G  00   9E 00009     MOVAB   DBG$COPY_MEMORY, R6                         ;
                                    55 00000000G  00   9E 00010     MOVAB   DBG$REL_MEMORY, R5                          ;
                                    54 00000000'  EF   9E 00017     MOVAB   LAST_LOC, R4                                ;
                                    5E            0C   C2 0001E     SUBL2   #12, SP                                     ;
                                             04   AE   D4 00021     CLRL    OLD_SYMID_LIST                              ; 0987
                                                  7E   D4 00024     CLRL    NEW_SYMID_LIST                              ; 0988
                                                  53   D4 00026     CLRL    SAVE_ADDR_EXP_DESC                          ; 0989
                                    52            04   AC   D0 00028     MOVL    ADDR_EXP_DESC, R2                   ; 0994
                                                  73   13 0002C     BEQL    7$                                          ;
                                FFFFFFFF    8F        08   AC   D1 0002E     CMPL    OVERRIDE_TYPE, #-1                  ; 1003
                                                  0A   13 00036     BEQL    1$                                          ;
                                FFFFFFFF    8F        0C   AC   D1 00038     CMPL    OVERRIDE_LENGTH, #-1                ; 1005
                                                  07   12 00040     BNEQ    2$                                          ;
                                      0C    A4        04   A4   7D 00042  1$:  MOVQ    POTENTIAL_LAST_TYPE, LAST_TYPE      ; 1008
                                                  05   11 00047     BRB     3$                                          ; 1003
                                      0C    A4        08   AC   7D 00049  2$:  MOVQ    OVERRIDE_TYPE, LAST_TYPE            ; 1013
                                                  52   DD 0004E  3$:  PUSHL   R2                                      ; 1022
                                            66          01   FB 00050     CALLS   #1, DBG$COPY_MEMORY                 ;
                                            53          50   D0 00053     MOVL    R0, SAVE_ADDR_EXP_DESC              ;
                                      07    79    8F    62   8F 00056     CASEB   (R2), #121, #7                      ; 1027
                          0066        0058        0066      0012      0005B  4$:  .WORD   5$-4$,-                        ;
                          0066        0066        0066      0066      00063          11$-4$,-                         ;
                                                                                   9$-4$,-                          ;
                                                                                   11$-4$,-                         ;
                                                                                   11$-4$,-                         ;
                                                                                   11$-4$,-                         ;
                                                                                   11$-4$,-                         ;
                                                                                   11$-4$                           ;
                                                  54   11 0006B     BRB     11$                                         ;
                                             0C   AE   9F 0006D  5$:  PUSHAB  DUMMY                                   ; 1040
                                             04   AE   9F 00070     PUSHAB  NEW_SYMID_LIST                              ;
                                             04   A2   DD 00073     PUSHL   4(R2)                                       ;
                                            67          03   FB 00076     CALLS   #3, DBG$NGET_SYMID                  ;
                                            13          50   E9 00079     BLBC    R0, 6$                              ;
```

```
                                                0C   AE   9F 0007C          PUSHAB  DUMMY                                   ;1053
                                                08   AE   9F 0007F          PUSHAB  NEW_PRIM                                ;
                                                04   A2   DD 00082          PUSHL   4(R2)                                   ;
                                 00000000G      00        03   FB 00085    CALLS   #3, DBG$NCOPY_DESC                      ;
                                                14        50   E8 0008C    BLBS    R0, 8$                                  ;
                                          00028663        8F   DD 0008F 6$: PUSHL   #165475                                ;1056
                                 00000000G      00        01   FB 00095    CALLS   #1, DBG$NOUT_INFO                       ;
                                                          53   DD 0009C    PUSHL   SAVE_ADDR_EXP_DESC                      ;1057
                                                65        01   FB 0009E    CALLS   #1, DBG$REL_MEMORY                      ;
                                                          1C   11 000A1 7$: BRB     10$                                     ;1058
                                                          6E   DD 000A3 8$: PUSHL   NEW_SYMID_LIST                          ;1065
                                 00000000G      00        01   FB 000A5    CALLS   #1, DBG$STA_LOCK_SYMID                  ;
                                       04       A3   04   AE   D0 000AC    MOVL    NEW_PRIM, 4(SAVE_ADDR_EXP_DESC)         ;1070
                                                          0E   11 000B1    BRB     11$                                     ;1027
                                                04        A2   DD 000B3 9$: PUSHL   4(R2)                                   ;1081
                                                66        01   FB 000B6    CALLS   #1, DBG$COPY_MEMORY                     ;
                                       04       A3        50   D0 000B9    MOVL    R0, 4(SAVE_ADDR_EXP_DESC)               ;
                                                          02   11 000BD    BRB     11$                                     ;0994
                                                          53   D4 000BF 10$: CLRL    SAVE_ADDR_EXP_DESC                      ;1097
                                                50        64   D0 000C1 11$: MOVL    LAST_LOC, R0                            ;1103
                                                          4C   13 000C4    BEQL    17$                                     ;
                                    07          79   8F        60   8F 000C6    CASEB   (R0), #121, #7                          ;1106
                0042                003C             0042        0012      000CB 12$: .WORD   13$-12$,-                       ;
                0042                0042             0042        0042      000D3      16$-12$,-                               ;
                                                                                      15$-12$,-                               ;
                                                                                      16$-12$,-                               ;
                                                                                      16$-12$,-                               ;
                                                                                      16$-12$,-                               ;
                                                                                      16$-12$,-                               ;
                                                                                      16$-12$                                 ;
                                                          30   11 000DB    BRB     16$                                     ;
                                                52        04   A0   D0 000DD 13$: MOVL    4(R0), OLD_PRIM_DESC            ;1115
                                                0C        AE   9F 000E1    PUSHAB  DUMMY                                   ;1120
                                                0C        AE   9F 000E4    PUSHAB  OLD_SYMID_LIST                          ;
                                                          52   DD 000E7    PUSHL   OLD_PRIM_DESC                           ;
                                                67        03   FB 000E9    CALLS   #3, DBG$NGET_SYMID                      ;
                                                0A        50   E9 000EC    BLBC    R0, 14$                                 ;
                                                08        AE   DD 000EF    PUSHL   OLD_SYMID_LIST                          ;1122
                                 00000000G      00        01   FB 000F2    CALLS   #1, DBG$STA_UNLOCK_SYMID                ;
                                                0C        AE   9F 000F9 14$: PUSHAB  DUMMY                                  ;1126
                                                          52   DD 000FC    PUSHL   OLD_PRIM_DESC                           ;
                                 00000000G      00        02   FB 000FE    CALLS   #2, DBG$NFREE_DESC                      ;
                                                          06   11 00105    BRB     16$                                     ;1106
                                                04        A0   DD 00107 15$: PUSHL   4(R0)                                  ;1135
                                                65        01   FB 0010A    CALLS   #1, DBG$REL_MEMORY                      ;
                                                          64   DD 0010D 16$: PUSHL   LAST_LOC                               ;1154
                                                65        01   FB 0010F    CALLS   #1, DBG$REL_MEMORY                      ;
                                                          53   D5 00112 17$: TSTL    SAVE_ADDR_EXP_DESC                      ;1161
                                                          04   13 00114    BEQL    18$                                     ;
                                                64        53   D0 00116    MOVL    SAVE_ADDR_EXP_DESC, LAST_LOC            ;1163
                                                          04   00119       RET                                             ;
                                                          64   D4 0011A 18$: CLRL    LAST_LOC                                ;1165
                                                          04   0011C       RET                                             ;1169

; Routine Size:  285 bytes,    Routine Base:  DBG$CODE + 01F4
```

; 1049 1170 1

```
;  1051          1171  1 GLOBAL ROUTINE DBG$NCANCEL_LOC_AND_VAL : NOVALUE =
;  1052          1172  1
;  1053          1173  1 !++
;  1054          1174  1 ! FUNCTIONAL DESCRIPTION:
;  1055          1175  1 !
;  1056          1176  1 !       When changing languages, last val and current location go away. This
;  1057          1177  1 !       routine does this.
;  1058          1178  1 !
;  1059          1179  1 ! FORMAL PARAMETERS:
;  1060          1180  1 !
;  1061          1181  1 !       NONE
;  1062          1182  1 !
;  1063          1183  1 ! IMPLICIT INPUTS:
;  1064          1184  1 !
;  1065          1185  1 !       dbg$gl_last_val         - Old debugger last value
;  1066          1186  1 !
;  1067          1187  1 !       dbg$gl_last_loc         - Old debugger last location (current loc)
;  1068          1188  1 !
;  1069          1189  1 !       dbg$gl_next_loc         - Old debugger next location ( <CR> )
;  1070          1190  1 !
;  1071          1191  1 !       last_value              - Pointer to value descriptor for last val
;  1072          1192  1 !
;  1073          1193  1 !       last_loc                - Pointer to addr exp desc for current loc
;  1074          1194  1 !
;  1075          1195  1 ! IMPLICIT OUTPUTS:
;  1076          1196  1 !
;  1077          1197  1 !       NONE
;  1078          1198  1 !
;  1079          1199  1 ! ROUTINE VALUE:
;  1080          1200  1 !
;  1081          1201  1 !       NOVALUE
;  1082          1202  1 !
;  1083          1203  1 ! COMPLETION CODES:
;  1084          1204  1 !
;  1085          1205  1 !       NONE
;  1086          1206  1 !
;  1087          1207  1 ! SIDE EFFECTS:
;  1088          1208  1 !
;  1089          1209  1 !       The meaning of current location and last value is erased for the old and
;  1090          1210  1 !       new debuggers.
;  1091          1211  1 !
;  1092          1212  1 !--
;  1093          1213  2     BEGIN
;  1094          1214  2
;  1095          1215  2
;  1096          1216  2     ! Wipe out the pseudos for the old debugger
;  1097          1217  2     !
;  1098          1218  2     dbg$gl_last_val = 0;
;  1099          1219  2     dbg$gl_last_loc = 0;
;  1100          1220  2     dbg$gl_next_loc = 0;
;  1101          1221  2
;  1102          1222  2
;  1103          1223  2     ! Wipe out the pseudos for the new debugger
;  1104          1224  2     !
;  1105          1225  2     dbg$nsave_last_val (0);
;  1106          1226  2     dbg$nsave_last_loc (0, dbg$k_notype, 0);
;  1107          1227  2
```

```
; 1108          1228  2     RETURN;
; 1109          1229  1     END;        ! End to dbg$ncancel_loc_and_val


                                                0000 00000          .ENTRY  DBG$NCANCEL_LOC_AND_VAL, Save nothing      ; 1171
                                    00000000G   00   D4 00002      CLRL    DBG$GL_LAST_VAL                            ; 1218
                                    00000000G   00   D4 00008      CLRL    DBG$GL_LAST_LOC                            ; 1219
                                    00000000G   00   D4 0000E      CLRL    DBG$GL_NEXT_LOC                            ; 1220
                                                7E   D4 00014      CLRL    -(SP)                                      ; 1225
                                FE29    CF          01   FB 00016      CALLS   #1, DBG$NSAVE_LAST_VAL                     ;
                                                7E   D4 0001B      CLRL    -(SP)                                      ; 1226
                                        7E      80  8F   9A 0001D      MOVZBL  #128, -(SP)                                ;
                                                7E   D4 00021      CLRL    -(SP)                                      ;
                                FEBB    CF          03   FB 00023      CALLS   #3, DBG$NSAVE_LAST_LOC                     ;
                                                    04 00028       RET                                                ; 1229

; Routine Size:  41 bytes,    Routine Base:  DBG$CODE + 0311

; 1110          1230  1
```

```
;  1112          1231  1 GLOBAL ROUTINE DBG$NPERM_SYM_INT (REG_NUM, PERM_SYM_PTR, PATH_NAME_VECT, MESSAGE_VECT) =
;  1113          1232  1
;  1114          1233  1 !++
;  1115          1234  1 ! FUNCTIONAL DESCRIPTION:
;  1116          1235  1 !
;  1117          1236  1 !       A kernel DEBUG routine to obtain a permanent symbol descriptor for the
;  1118          1237  1 !       DEBUG permanent symbols R0-R11, AP, PC, FP, and PSL.
;  1119          1238  1 !
;  1120          1239  1 !       A permanent symbol descriptor contains information which:
;  1121          1240  1 !
;  1122          1241  1 !       1) - identifies the symbol by register number
;  1123          1242  1 !
;  1124          1243  1 !       2) - contains a virtual address which represents the location at which
;  1125          1244  1 !            the debugger has stored the value of the register
;  1126          1245  1 !
;  1127          1246  1 ! FORMAL PARAMETERS:
;  1128          1247  1 !
;  1129          1248  1 !       reg_num         - A longword containing an unsigned integer between 0 and 16,
;  1130          1249  1 !                         inclusive. This integer identifies the permanent symbol
;  1131          1250  1 !                         to be found according to the following code:
;  1132          1251  1 !
;  1133          1252  1 !                           dbg$k_r0 (200)       - R0
;  1134          1253  1 !
;  1135          1254  1 !                           dbg$k_r1 (201)       - R1
;  1136          1255  1 !
;  1137          1256  1 !                           dbg$k_r2 (202)       - R2
;  1138          1257  1 !
;  1139          1258  1 !                           dbg$k_r3 (203)       - R3
;  1140          1259  1 !
;  1141          1260  1 !                           dbg$k_r3 (204)       - R4
;  1142          1261  1 !
;  1143          1262  1 !                           dbg$k_r5 (205)       - R5
;  1144          1263  1 !
;  1145          1264  1 !                           dbg$k_r6 (206)       - R6
;  1146          1265  1 !
;  1147          1266  1 !                           dbg$k_r7 (207)       - R7
;  1148          1267  1 !
;  1149          1268  1 !                           dbg$k_r8 (208)       - R8
;  1150          1269  1 !
;  1151          1270  1 !                           dbg$k_r9 (209)       - R9
;  1152          1271  1 !
;  1153          1272  1 !                           dbg$k_r10 (210)      - R10
;  1154          1273  1 !
;  1155          1274  1 !                           dbg$k_r11 (211)      - R11
;  1156          1275  1 !
;  1157          1276  1 !                           dbg$k_ap (212)       - R12 (AP)
;  1158          1277  1 !
;  1159          1278  1 !                           dbg$k_fp (213)       - R13 (FP)
;  1160          1279  1 !
;  1161          1280  1 !                           dbg$k_sp (214)       - R14 (SP)
;  1162          1281  1 !
;  1163          1282  1 !                           dbg$k_pc (215)       - R15 (PC)
;  1164          1283  1 !
;  1165          1284  1 !                           dbg$k_psl (216)      - PSL
;  1166          1285  1 !
;  1167          1286  1 !       perm_sym_ptr    - The address of a longword to contain the address of a
;  1168          1287  1 !                         permanent symbol descriptor upon return. This permanent
```

```
;1169     1288  1 !                       symbol descriptor consists of nine byte block. The first
;1170     1289  1 !                       byte contains the register id number as input in reg_num.
;1171     1290  1 !                       The next four bytes contain the address of the stored
;1172     1291  1 !                       value of the register. The next four bytes are reserved
;1173     1292  1 !                       for future use and will be set to 0 in this release.
;1174     1293  1 !
;1175     1294  1 !       path_name_vect - Not used in this release. Should be a longword containing
;1176     1295  1 !                       0. A future release of DEBUG will allow pathname qual-
;1177     1296  1 !                       ification of registers.
;1178     1297  1 !
;1179     1298  1 !       message_vect   - The address of a longword to contain the add ess of a
;1180     1299  1 !                       message argument vector as described on page 4-119 of
;1181     1300  1 !                       the VAX/VMS system reference, volume 1A.
;1182     1301  1 !
;1183     1302  1 ! IMPLICIT INPUTS:
;1184     1303  1 !
;1185     1304  1 !       NONE
;1186     1305  1 !
;1187     1306  1 ! IMPLICIT OUTPUTS:
;1188     1307  1 !
;1189     1308  1 !       In case of a success return, a permanent symbol descriptor is constructed
;1190     1309  1 !       and returned.
;1191     1310  1 !
;1192     1311  1 !       In case of a severe error return, a message argument vector is constructed
;1193     1312  1 !       and returned.
;1194     1313  1 !
;1195     1314  1 ! ROUTINE VALUE:
;1196     1315  1 !
;1197     1316  1 !       An usigned integer longword completion code
;1198     1317  1 !
;1199     1318  1 ! COMPLETION CODES:
;1200     1319  1 !
;1201     1320  1 !       STS$K_SUCCESS (1) - Success. Permanent symbol descriptor constructed
;1202     1321  1 !                           and returned.
;1203     1322  1 !
;1204     1323  1 !       STS$K_SEVERE  (4) - Failure. Permanent symbol not interpreted. Message
;1205     1324  1 !                           argument vector constructed and returned.
;1206     1325  1 !
;1207     1326  1 ! SIDE EFFECTS:
;1208     1327  1 !
;1209     1328  1 !       NONE
;1210     1329  1 !
;1211     1330  1 !--
;1212     1331  2     BEGIN
;1213     1332  2
;1214     1333  2     LOCAL
;1215     1334  2         PERM_SYM_DESC : REF dbg$permsd;
;1216     1335  2
;1217     1336  2
;1218     1337  2     ! Create storage for the permanent symbol descriptor
;1219     1338  2     !
;1220     1339  2     perm_sym_desc = dbg$get_tempmem(dbg$k_permsd_size);
;1221     1340  2
;1222     1341  2
;1223     1342  2     ! copy the register id over to the permanent symbols descriptor and make sure
;1224     1343  2     ! that the pathname pointer field is 0.
;1225     1344  2     !
```

```
;  1226          1345  2     perm_sym_desc [dbg$b_permsd_id] = .reg_num;
;  1227          1346  2     perm_sym_desc [dbg$l_permsd_path] = 0;
;  1228          1347  2
;  1229          1348  2
;  1230          1349  2     ! Obtain the address of the register, keying off the id number.
;  1231          1350  2     !
;  1232          1351  2     perm_sym_desc [dbg$l_permsd_addr] =
;  1233          1352  2
;  1234          1353  3         (CASE .reg_num FROM dbg$k_r0 TO dbg$k_psl
;  1235          1354  3             OF
;  1236          1355  3             SET
;  1237          1356  3
;  1238          1357  3             [dbg$k_r0]          : dbg$runframe [dbg$l_user_r0];
;  1239          1358  3             [dbg$k_r1]          : dbg$runframe [dbg$l_user_r1];
;  1240          1359  3             [dbg$k_r2]          : dbg$runframe [dbg$l_user_r2];
;  1241          1360  3             [dbg$k_r3]          : dbg$runframe [dbg$l_user_r3];
;  1242          1361  3             [dbg$k_r4]          : dbg$runframe [dbg$l_user_r4];
;  1243          1362  3             [dbg$k_r5]          : dbg$runframe [dbg$l_user_r5];
;  1244          1363  3             [dbg$k_r6]          : dbg$runframe [dbg$l_user_r6];
;  1245          1364  3             [dbg$k_r7]          : dbg$runframe [dbg$l_user_r7];
;  1246          1365  3             [dbg$k_r8]          : dbg$runframe [dbg$l_user_r8];
;  1247          1366  3             [dbg$k_r9]          : dbg$runframe [dbg$l_user_r9];
;  1248          1367  3             [dbg$k_r10]         : dbg$runframe [dbg$l_user_r10];
;  1249          1368  3             [dbg$k_r11]         : dbg$runframe [dbg$l_user_r11];
;  1250          1369  3             [dbg$k_ap]          : dbg$runframe [dbg$l_user_ap];
;  1251          1370  3             [dbg$k_fp]          : dbg$runframe [dbg$l_user_fp];
;  1252          1371  3             [dbg$k_sp]          : dbg$runframe [dbg$l_user_sp];
;  1253          1372  3             [dbg$k_pc]          : dbg$runframe [dbg$l_user_pc];
;  1254          1373  3             [dbg$k_psl]         : dbg$runframe [dbg$l_user_psl];
;  1255          1374  3
;  1256          1375  3             [INRANGE,OUTRANGE]  : $DBG_ERROR('DBGNSDATA\DBG$NPERM_SYM_INT');
;  1257          1376  2             TES);
;  1258          1377  2
;  1259          1378  2     .perm_sym_ptr = .perm_sym_desc;
;  1260          1379  2
;  1261          1380  2     RETURN sts$k_success;
;  1262          1381  2
;  1263          1382  1     END;                    ! End of dbg$nperm_sym_int


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

24 47 42 44 5C 41 54 41 44 53 4E 47 42 44 1B 00000 P.AAA:  .ASCII  <27>\DBGNSDATA\<92>\DBG$NPERM_SYM_INT\      ;
      54 4E 49 5F 4D 59 53 5F 4D 52 45 50 4E 0000F                                                            ;


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

                                                  000C 00000    .ENTRY  DBG$NPERM_SYM_INT, Save R2,R3           ; 1231
                                53 00000000G    00   9E 00002  MOVAB   DBG$RUNFRAME+4, R3                     ;
                                                03   DD 00009  PUSHL   #3                                      ; 1339
                           00000000G            01   FB 0000B  CALLS   #1, DBG$GET_TEMPMEM                    ;
                                52              50   D0 00012  MOVL    R0, PERM_SYM_DESC                     ;
                                62         04   AC   90 00015  MOVB    REG_NUM, (PERM_SYM_DESC)              ; 1345
                                           08   A2   D4 00019  CLRL    8(PERM_SYM_DESC)                      ; 1346
```

```
                     10 000000C8   8F      04   AC  CF 0001C          CASEL   REG_NUM, #200, #16                      ; 1353
          004A       0044          003E         0039     00025 1$:     .WORD   2$-T$,-                                ;
          0062       005C          0056         0050     0002D         3$-1$,-                                        ;
          007A       0074          006E         0068     00035         4$-1$,-                                        ;
          0092       008C          0086         0080     0003D         5$-1$,-                                        ;
                                                0098     00045         6$-1$,-                                        ;
                                                                       7$-1$,-                                        ;
                                                                       8$-1$,-                                        ;
                                                                       9$-1$,-                                        ;
                                                                       10$-1$,-                                       ;
                                                                       11$-1$,-                                       ;
                                                                       12$-1$,-                                       ;
                                                                       13$-1$,-                                       ;
                                                                       14$-1$,-                                       ;
                                                                       15$-1$,-                                       ;
                                                                       16$-1$,-                                       ;
                                                                       17$-1$,-                                       ;
                                                                       18$-1$                                         ;
                          00000000'    EF  9F 00047          PUSHAB  P.AAA                                          ; 1375
                                       01  DD 0004D          PUSHL   #1                                             ;
                          00028362     8F  DD 0004F          PUSHL   #164706                                        ;
                00000000G  00          03  FB 00055          CALLS   #3, LIB$SIGNAL                                 ;
                                       63  11 0005C          BRB     19$                                            ;
                           50          63  9E 0005E 2$:      MOVAB   DBG$RUNFRAME+4, R0                             ; 1357
                                       5E  11 00061          BRB     19$                                            ;
                           50      04  A3  9E 00063 3$:      MOVAB   DBG$RUNFRAME+8, R0                             ; 1358
                                       58  11 00067          BRB     19$                                            ;
                           50      08  A3  9E 00069 4$:      MOVAB   DBG$RUNFRAME+12, R0                            ; 1359
                                       52  11 0006D          BRB     19$                                            ;
                           50      0C  A3  9E 0006F 5$:      MOVAB   DBG$RUNFRAME+16, R0                            ; 1360
                                       4C  11 00073          BRB     19$                                            ;
                           50      10  A3  9E 00075 6$:      MOVAB   DBG$RUNFRAME+20, R0                            ; 1361
                                       46  11 00079          BRB     19$                                            ;
                           50      14  A3  9E 0007B 7$:      MOVAB   DBG$RUNFRAME+24, R0                            ; 1362
                                       40  11 0007F          BRB     19$                                            ;
                           50      18  A3  9E 00081 8$:      MOVAB   DBG$RUNFRAME+28, R0                            ; 1363
                                       3A  11 00085          BRB     19$                                            ;
                           50      1C  A3  9E 00087 9$:      MOVAB   DBG$RUNFRAME+32, R0                            ; 1364
                                       34  11 0008B          BRB     19$                                            ;
                           50      20  A3  9E 0008D 10$:     MOVAB   DBG$RUNFRAME+36, R0                            ; 1365
                                       2E  11 00091          BRB     19$                                            ;
                           50      24  A3  9E 00093 11$:     MOVAB   DBG$RUNFRAME+40, R0                            ; 1366
                                       28  11 00097          BRB     19$                                            ;
                           50      28  A3  9E 00099 12$:     MOVAB   DBG$RUNFRAME+44, R0                            ; 1367
                                       22  11 0009D          BRB     19$                                            ;
                           50      2C  A3  9E 0009F 13$:     MOVAB   DBG$RUNFRAME+48, R0                            ; 1368
                                       1C  11 000A3          BRB     19$                                            ;
                           50      30  A3  9E 000A5 14$:     MOVAB   DBG$RUNFRAME+52, R0                            ; 1369
                                       16  11 000A9          BRB     19$                                            ;
                           50      34  A3  9E 000AB 15$:     MOVAB   DBG$RUNFRAME+56, R0                            ; 1370
                                       10  11 000AF          BRB     19$                                            ;
                           50      38  A3  9E 000B1 16$:     MOVAB   DBG$RUNFRAME+60, R0                            ; 1371
                                       0A  11 000B5          BRB     19$                                            ;
                           50      3C  A3  9E 000B7 17$:     MOVAB   DBG$RUNFRAME+64, R0                            ; 1372
                                       04  11 000BB          BRB     19$                                            ;
                           50      40  A3  9E 000BD 18$:     MOVAB   DBG$RUNFRAME+68, R0                            ; 1373
                    04     A2              50  D0 000C1 19$:     MOVL    R0, 4(PERM_SYM_DESC)                       ; 1353
```

```
                08    BC              52  D0 000C5          MOVL    PERM_SYM_DESC, @PERM_SYM_PTR             ; 1378
                      50              01  D0 000C9          MOVL    #1, R0                                   ; 1380
                                          04 000CC          RET                                              ; 1382

; Routine Size:  205 bytes,    Routine Base:  DBG$CODE + 033A
```

```
; 1265          1383  1 END                           !End of module
; 1266          1384  0 ELUDOM


                                                      .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;   DBG$OWN                       24  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;   DBG$CODE                    1031  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;   DBG$PLIT                      28  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         3         0        1000        00:01.9
;   _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1
;                                        32         0         0           7        00:00.1
;   _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1
;                                      1545        48         3          97        00:02.0
;   _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                       418         2         0          31        00:00.4
;   _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1
;                                       386        22         5          22        00:00.3
;   _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1
;                                       150         2         1          12        00:00.3



;                               COMMAND QUALIFIERS

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

; Size:          1031 code + 52 data bytes
; Run Time:         00:23.8
; Elapsed Time:     01:19.7
; Lines/CPU Min:     3487
; Lexemes/CPU-Min:   8469
; Memory Used:   132 pages
; Compilation Complete
```

DBGNSET
LIS

DBGNSDATA
LIS

DBGNSEARC
LIS