F 1

```
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
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  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     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     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
```

FILEID**DBGNSDATA

```
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  GGGGGG  NN    NNNN          SS  DD      DD  AAAAAAAAAA      TT      AAAAAAAAAA
DD      DD  BB      BB  GG  GGGGGG  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
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 MODULE DBGNSDATA (IDENT = 'V04-000') =
0
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:
1 !
1 !       DEBUG
1 !
1 ! ABSTRACT:
1 !
1 !       This module consists of data structures and data structure management
1 !       routines that manage the user's session specific data during the
1 !       processing of debugger commands while under the control of the new
1 !       Command Line Interpreter. Specifically, this module keeps track of
1 !       scopes, types, override types, radicies, current location, and
1 !       last value.
1 !
1 ! ENVIRONMENT:
1 !
1 !       VAX/VMS
1 !
1 ! AUTHOR:
1 !
1 !       David Plummer
1 !
1 ! CREATION DATE:
1 !
1 !       10-Jul-80
1 !
1 ! VERSION:
1 !
```

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

```
1 !
1 ! TABLE OF CONTENTS:
1 !
1
1 FORWARD ROUTINE
1         DBG$NGET_RADIX,                       ! Returns the user specified or default radix
1         DBG$NGET_TRANS_RADIX,                 ! Translates a default radix into the actual radix
1         DBG$NGET_DEFAULT_TYPE     : NOVALUE,  ! Returns user set default type
1         DBG$NGET_OVERRIDE_TYPE,               ! Returns user set override type
1         DBG$NGET_MODE,                        ! Returns true if 'symbolic', false
1                                               ! if 'nosymbolic'
1         DBG$NREPORT_LAST_VAL,                 ! Returns the last value given to the
1                                               ! user that was stored by the DEBUGGER
1         DBG$NREPORT_LAST_LOC,                 ! Returns the last location that was
1                                               ! shown to the user by the DEBUGGER
1         DBG$NSET_LAST_TYPLEN      : NOVALUE,  ! Stores potential last loc type and length
1         DBG$NGET_POTENTIAL_TYPE   : NOVALUE,  ! Returns the type associted with '.'
1                                               ! by the Address Expression Interpreter
1         DBG$NSAVE_LAST_VAL        : NOVALUE,  ! Stores a value descriptor for last val
1         DBG$NSAVE_LAST_LOC        : NOVALUE,  ! Stores an address expression desc for
1                                               ! current location
1         DBG$NCANCEL_LOC_AND_VAL   : NOVALUE,  ! Cancel last val and current loc for both
1                                               ! old and new debuggers
1         DBG$NPERM_SYM_INT;                    ! Permanent symbol Interpreter
1
1
1 !
1 ! REQUIRE FILES:
1 !
1
1 REQUIRE 'SRC$:DBGPROLOG.REQ';
1
1 LIBRARY 'LIB$:DBGGEN.L32';
1
1 !
1 ! OWN STORAGE:
1 !
1 OWN
1         LAST_VALUE              : REF dbg$dhead INITIAL (0),  ! Pointer to last value
1         LAST_LOC                : REF dbg$aed INITIAL (0),    ! Pointer to last location
1         POTENTIAL_LAST_TYPE     : INITIAL (-1),               ! Possible type for current loc
1         POTENTIAL_LAST_LENGTH   : INITIAL (0),                ! Possible length for current loc
1         LAST_TYPE               : INITIAL (-1),               ! Real type for current loc
1         LAST_LENGTH             : INITIAL (0);                ! Real length for current loc
1
1
1
1 !
1 ! EXTERNAL REFERENCES:
1 !
1
1 EXTERNAL ROUTINE
1         DBG$COPY_MEMORY,                          ! Make a copy of a memory block
1         DBG$GET_TEMPMEM,                          ! Get temporary memory block
1         DBG$NCOPY_DESC,                           ! Copies primary and value descriptors
1         DBG$NFREE_DESC,                           ! Frees copied descriptors
1         DBG$NGET_SYMID,                           ! Obtains symid list for descriptors
```

```
1          DBG$NMAKE_ARG_VECT,                        ! Constructs a message arguemnt vector
1          DBG$NOUT_INFO,                             ! Outputs an informational message
1          DBG$SET_MOD_LVL,                           ! Sets mode pointer
1          DBG$STA_LOCK_SYMID   : NOVALUE,            ! Saves symids
1          DBG$STA_UNLOCK_SYMID : NOVALUE,            ! Releases symids
1          DBG$REL_MEMORY : NOVALUE;                  ! Release memory block
1
1 EXTERNAL
1          DBG$RUNFRAME            : BLOCK [,BYTE],          ! User runframe
1          DBG$GB_RADIX            : VECTOR[3, BYTE],        ! Radix settings
1          DBG$GL_LAST_VAL,                                  ! Old last value
1          DBG$GL_LAST_LOC,                                  ! Old last location
1          DBG$GL_NEXT_LOC,                                  ! Old next location
1          DBG$GL_GBLTYP,                                    ! Override type
1          DBG$GW_GBLLNGTH         : WORD,                   ! Override length
1          DBG$GL_DFLTTYP,                                   ! Default type
1          DBG$GW_DFLTLENG         : WORD,                   ! Default length
1          DBG$GB_MOD_PTR          : REF VECTOR [,BYTE],     ! Mode structure
1          DBG$GB_LANGUAGE         : BYTE;                   ! Language setting
```

```
1 GLOBAL ROUTINE DBG$NGET_RADIX =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Returns the current output radix. This is obtained by first checking
1 !       whether there is a RADIX/OVERRIDE in effect, and if so, returning
1 !       that. If not, return the current radix as specified by SET RADIX
1 !       or by the language default.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       DBG$GB_RADIX    - Global vector containing radix settings.
1 !
1 !--
2     BEGIN
2     LOCAL
2         RADIX;
2     RADIX = .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT_OVER];
2     IF .RADIX NEQ DBG$K_DEFAULT
2     THEN
2         RETURN .RADIX;
2     RETURN .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT];
1     END;                          ! End of dbg$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_LVL
                                    .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_DFLTTYP, 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
```

```
1 GLOBAL ROUTINE DBG$NGET_TRANS_RADIX (RADIX) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Returns a translation of the default radix if dbg$k_default was suppplied.
1 !       Otherwise, simply returns the input value.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       RADIX    - A longword containing the radix literal.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword corresponding to the translated radix literal.
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2       BEGIN
2       IF .radix EQL dbg$k_default
2       THEN
3           BEGIN
3           IF .dbg$gb_language NEQ dbg$k_macro
3                           AND
3              .dbg$gb_language NEQ dbg$k_bliss
3           THEN
3               RETURN dbg$k_decimal
3           ELSE
3               RETURN dbg$k_hex;
3           END
2      ELSE
3           BEGIN
3           RETURN .radix;
2           END;
1       END;                    ! End of dbg$nget_trans_radix


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

```
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
```

```
GLOBAL ROUTINE DBG$NGET_MODE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine returns an indication of the user specified mode: that is,
!       TRUE if 'SYMBOLIC', FALSE if 'NOSYMBOLIC'
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       DBG$GB_MOD_PTR  - Pointer to current mode level
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       TRUE    - If mode_symbols is true
!
!       FALSE   - If mode_symbols is false
!
! SIDE EFFECTS:
!
!       NONE
!
!--
   BEGIN

   LOCAL
       SAVE_LEVEL,                              ! Holds old pointer
       LEVEL_PTR : REF VECTOR [,BYTE];          ! mode structure


   ! Save the present mode level
   !
   save_level= .dbg$gb_mod_ptr;


   ! Set the mode pointer to the user default level
   !
   dbg$set_mod_lvl (user_def_mode);


   ! Pick up the new pointer
   !
   level_ptr = .dbg$gb_mod_ptr;


   ! Reset the mode level
   !
   dbg$gb_mod_ptr = .save_level;

```

```
;  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
```

```
1 GLOBAL ROUTINE DBG$NREPORT_LAST_VAL (VALUE_DESC, MESSAGE_VECT) =
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Returns the last value given to the user that was stored by the
1 !       DEBUGGER.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       value_desc      -       The address of a longword to contain the
1 !                               value descriptor last created to describe
1 !                               the output to the user.
1 !
1 !       msg_vec         -       The address of a longword to contain the
1 !                               address of a message argument vector
1 !                               as described on page 4-119 of the
1 !                               VAX/VMS system reference, volume 1A.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       LAST_VALUE      - Pointer to the stored last value descriptor
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       If successful returns a value descriptor describing the last value
1 !       that the user saw. Otherwise this routine returns a message argument
1 !       vector describing the error message.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code.
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS   (1)     - Successful return. The value descriptor
1 !                                 describing the last value that the user
1 !                                 saw was returned.
1 !
1 !       STS$K_SEVERE    (4)     - Failure. No value descriptor could be found.
1 !                                 The longword to contain the address of the
1 !                                 Standard VAX Message Argument Vector is
1 !                                 filled in to point to a fatal error message.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None.
1 !
1 !--
2     BEGIN
2
2     ! Check for not having a last value
2     !
2     IF .last_value EQLA 0
2     THEN
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
```

```
GLOBAL ROUTINE DBG$NREPORT_LAST_LOC( ADDR_EXP_DESC, TYPE, LENGTH, MESSAGE_VECT) =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Returns the last location that was returned shown to the user
!       by the DEBUGGER.
!
! FORMAL PARAMETERS:
!
!       add_exp_desc    -       The address of a longword to contain
!                               the address of an Address Expression
!                               Descriptor.
!
!       type            -       The address of a longword to contain
!                               the type of the last location shown
!                               to the user. This type may be one
!                               of the following types:
!
!                               dsc$k_dtype_b, dsc$k_dtype_w, dsc$k_dtype_l,
!                               dsc$k_dtype_lu, dsc$k_dtype_t, dsc$k_dtype_zi
!
!       length          -       The address of a longword to contain
!                               the length of the object output to the
!                               user.
!
!       msg_vec         -       The address of a longword to contain
!                               the address of a standard VAX Message
!                               Argument Vector as described on page
!                               4-119 of the VAX/VMS system reference
!                               manual, volume 1A.
!
! IMPLICIT INPUTS:
!
!       LAST_LOC        - Pointer to the stored address expression descriptor
!
!       LAST_TYPE       - The type in which current loc was last displayed.
!
!       LAST_LENGTH     - The last length used for curren loc.
!
! IMPLICIT OUTPUTS:
!
!       If successful, this routine returns an Address Expression Descriptor
!       describing the last location that the user looked at and the associated
!       type and length of the object previously looked at by the user in the
!       DEBUGGER.
!
!       If unsuccessful, the longword containing the address of the
!       Standard VAX Message Argument Vector for that buffer is updated
!       to contain the address of the error message buffer.
!
! ROUTINE VALUE:
!
!       STS$K_SUCCESS   (1)     - An Address Expression Descriptor describing
!                                 the last location looked at by the user was
!                                 found and is returned along with the
```

```
!                                        associated type and length of the object that
!                                        was displayed.
!
!        STS$K_SEVERE    (4)       - Failure. No Address Expression Descriptor
!                                        describing the last location that the user
!                                        looked at could be found. The longword
!                                        containing the address of the Standard VAX
!                                        Message Argument Vector is filled in to
!                                        point to a fatal error message.
!
! SIDE EFFECTS:
!
!        None.
!
!--

    BEGIN

    ! Check for no current location
    !
    IF .last_loc EQLA 0
    THEN
        BEGIN
        .message_vect = dbg$nmake_arg_vect (dbg$_nocurloc);
        RETURN sts$k_severe;
        END;

    ! Check to see if we have a permanent symbol descriptor. If we do, then
    ! we must get a new interpretation of the register
    !
    IF .last_loc [dbg$b_aed_type] EQL dbg$k_perm_desc
    THEN
        BEGIN
        LOCAL
            PSD_PTR,
            PERM_DESC : REF dbg$permsd;

        perm_desc = .last_loc [dbg$l_aed_value];

        IF NOT dbg$nperm_sym_int (.perm_desc [dbg$b_permsd_id], psd_ptr, 0, .message_vect)
        THEN
            RETURN sts$k_severe;

        ! Release the old permanent symbol descriptor and set up the address
        ! expression descriptor to point to the new one.
        !
        dbg$rel_memory (.perm_desc);

        ! Make a permanent copy of the new permanent symbol descriptor
        !
        perm_desc = dbg$copy_memory (.psd_ptr);
        last_loc [dbg$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
```

```
1 GLOBAL ROUTINE DBG$NSET_LAST_TYPLEN (TYPE, LENGTH) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Stores the values of the input parameters for possible use of typing
1 !       current location. The type for curren loc is returned by the AEI
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       TYPE    -       A longword containing the value of a vax standard type.
1 !
1 !       LENGTH  -       A longword containing the length in bytes to be stored.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       POTENTIAL_LAST_TYPE     - A longword to contain the value of type
1 !
1 !       POTENTIAL_LAST_LENGTH   - A longword to contain the length
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2    BEGIN
2
2    ! Just tuck away the information
2    !
2    potential_last_type = .type;
2    potential_last_length = .length;
2
2    RETURN;
2
1    END;        ! End of dbg$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
```

```
1 GLOBAL ROUTINE DBG$NGET_POTENTIAL_TYPE (TYPE, LENGTH) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine returns the AEI set type to be associated with current location
1 !       in lieu of any overrides. This information is used to type raw addresses.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       TYPE    - The address of a longword to contain type information
1 !
1 !       LENGTH  - The address of a longword to contain the length (in bytes)
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       potential_last_type     - a longword containing the type set by AEI
1 !
1 !       potential_last_length - a longword containing the length set by AEI
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NOVALUE
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2    BEGIN
2
2
2    ! Just return the requested information
2    !
2    .type = .potential_last_type;
2    .length = .potential_last_length;
2
2    RETURN;
2
1    END;          ! End of dbg$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
```

```
1 GLOBAL ROUTINE DBG$NSAVE_LAST_VAL (VALUE_DESC) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !      Saves the input parameter as the interpretation of
1 !      last value. A copy of the value descriptor is constructed from non-listed
1 !      storage.
1 !
1 !      Value descriptors may be volatile or non-volatile. This routine will
1 !      store only non-volatile value descriptors.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !      VALUE_DESC       - A longword containing the address of a value descriptor
1 !                         to be stored for last val. If 0, then last val is
1 !                         canceled.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !      LAST_VAL         - A longword to contain the address of the stored value
1 !                         descriptor
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !      NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !      NOVALUE
1 !
1 ! COMPLETION CODES:
1 !
1 !      NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !      Redefines last value.
1 !
1 !--
2     BEGIN
2
2     MAP VALUE_DESC : REF dbg$dhead;
2
2     LOCAL
2          DUMMY,                       ! Dummy parameter
2          OLD_SYMID_LIST,              ! Symid list for old '\'
2          NEW_SYMID_LIST,              ! Symid list for new '\'
2          MESSAGE_VECT,                ! Holds address of message argument vector
2          VALUE_PTR;                   ! Temporary pointer to value descriptor.
2
2     old_symid_list = 0;
2     new_symid_list = 0;
2     value_ptr = 0;
2
2
2     ! Check for non-null input
```

```
!
IF .value_desc NEQA 0
THEN
    BEGIN

    ! Check for a volatile value descriptor.
    !
    IF .value_desc [dbg$b_dhead_type] EQL dbg$k_v_value_desc
    THEN
        BEGIN
        dbg$nout_info (dbg$_nonewval);
        END
    ELSE
        BEGIN

        ! We have a non-volatile value descriptor. Obtain the new symid_list.
        !
        IF NOT dbg$nget_symid (.value_desc, new_symid_list, dummy)
        THEN
            BEGIN

            ! Once again, can't save new '\'.
            !
            dbg$nout_info (dbg$_nonewval);
            END

        ELSE
            BEGIN

            ! Try to copy the value descriptor
            !
            IF NOT dbg$ncopy_desc (.value_desc, value_ptr, dummy)
            THEN
                BEGIN

                ! Can't save '\'
                !
                dbg$nout_info (dbg$_nonewval);
                value_ptr = 0;
                END
            ELSE
                BEGIN

                ! The descriptor has been copied. Lock the symids.
                !
                dbg$sta_lock_symid (.new_symid_list);
                END;
            END;
        END;
    END;


! The new descriptor has been copied and saved. Delete the old descriptor and
! unlock symids
!
IF .last_value NEQA 0
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
```

```
GLOBAL ROUTINE DBG$NSAVE_LAST_LOC (ADDR_EXP_DESC, OVERRIDE_TYPE, OVERRIDE_LENGTH) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Makes an unlisted copy of an address expression descriptor to store as
!       the interpretation of current location. If override type and length are
!       supplied, current location is typed accordingly.
!
! FORMAL PARAMETERS:
!
!       ADDR_EXP_DESC     - A longword containing the address of an address expression
!                           descriptor. If 0, then current loc is canceled.
!
!       OVERRIDE_TYPE     - A longword containing a vax standard type with which to
!                           type current location.
!
!       OVERRIDE_LENGTH - A longword containg an integer representing the override
!                           length to be associated with current location.
!
! IMPLICIT INPUTS:
!
!       LAST_LOC          - A longword to contain the address of an addr exp desc
!                           for current location.
!
!       LAST_TYPE         - A longword to contain the override type for current loc
!
!       POTENTIAL_LAST_TYPE - A longword containing a possible type for current loc.
!
!       LAST_LENGTH       - A longword to contain the override length for current loc
!
!       POTENTIAL_LAST_LENGTH - A longword containing a potential last length for curren loc
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       NONE
!
!       NOVALUE
!
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Redefines current location.
!
!--
    BEGIN

    MAP
        ADDR_EXP_DESC       : REF dbg$aed;

    LOCAL
        DUMMY,                                  ! Dummy parameter
```

```
;       2           OLD_SYMID_LIST,                         ! Pointer to old symid list
;       2           NEW_SYMID_LIST,                         ! Pointer to new symid list
;       2           SAVE_ADDR_EXP_DESC : REF dbg$aed;       ! The address expression descriptor to save
;       2
;       2   old_symid_list = 0;
;       2   new_symid_list = 0;
;       2   save_addr_exp_desc = 0;
;       2
;       2
;       2   ! Check for null input
;       2   !
;       2   IF .addr_exp_desc NEQA 0
;       2   THEN
;       3       BEGIN
;       3
;       3       ! When this routine is called, an address expression descriptor is supplied
;       3       ! to become the definition of current loc. If a type and length (not EQL -1)
;       3       ! are supplied, we type current loc according to these parameters. Otherwise,
;       3       ! we use potential_type and potential_loc to type current loc.
;       3       !
;       3       IF .override_type EQL -1
;       3               OR
;       3          .override_length EQL -1
;       3       THEN
;       4           BEGIN
;       4           last_type = .potential_last_type;
;       4           last_length = .potential_last_length;
;       4           END
;       3       ELSE
;       4           BEGIN
;       4           last_type = .override_type;
;       4           last_length = .override_length;
;       3           END;
;       3
;       3
;       3       ! Current location has been typed. Now a copy of the address expression
;       3       ! descriptor, and any descriptors it points to must be made. First make a
;       3       ! copy of the address expression descriptor.
;       3       !
;       3       save_addr_exp_desc = dbg$copy_memory(.addr_exp_desc);
;       3
;       3
;       3       ! Make copies of any descriptors pointed to by the address expression descriptor
;       3       !
;       3       CASE .addr_exp_desc [dbg$b_aed_type] FROM dbg$k_primary_desc TO dbg$k_notype
;       3           OF
;       3           SET
;       3
;       3           [dbg$k_primary_desc] :
;       4               BEGIN
;       4               LOCAL
;       4                   NEW_PRIM : REF dbg$dhead;
;       4
;       4
;       4                   ! Have to copy the primary descriptor and notify the symbol table that
;       4                   ! we're hanging on to a symid.
;       4                   !
```

```
;  4                  IF NOT dbg$nget_symid (.addr_exp_desc [dbg$l_aed_value], new_symid_list, dummy)
;  4                  THEN
;  5                      BEGIN
;  5
;  5                      ! Can't save new '.'
;  5                      !
;  5                      dbg$nout_info (dbg$_nonewcur);
;  5                      dbg$rel_memory (.save_addr_exp_desc);
;  5                      save_addr_exp_desc = 0;
;  5                      END
;  5
;  4                  ELSE
;  5                      BEGIN
;  5                      IF NOT dbg$ncopy_desc (.addr_exp_desc [dbg$l_aed_value], new_prim, dummy)
;  5                      THEN
;  6                          BEGIN
;  6                          dbg$nout_info (dbg$_nonewcur);
;  6                          dbg$rel_memory (.save_addr_exp_desc);
;  6                          save_addr_exp_desc = 0;
;  6                          END
;  5                      ELSE
;  6                          BEGIN
;  6
;  6                          ! Notify the symbol table to hang on to the new symids
;  6                          !
;  6                          dbg$sta_lock_symid (.new_symid_list);
;  6
;  6
;  6                          ! point the address expression to the permanent primary descriptor
;  6                          !
;  6                          save_addr_exp_desc [dbg$l_aed_value] = .new_prim;
;  5                          END;
;  4                      END;
;  3                  END;
;  3
;  3              [dbg$k_perm_desc] :
;  4                  BEGIN
;  4
;  4                  ! Have to make a non-listed copy of the permanent symbol descriptor.
;  4                  !
;  4                  save_addr_exp_desc [dbg$l_aed_value] =
;  4                          dbg$copy_memory(.addr_exp_desc [dbg$l_aed_value]);
;  3                  END;
;  3
;  3              [INRANGE,OUTRANGE] :
;  4                  BEGIN
;  4
;  4                  ! Not pointing to any descriptors. All the work has been done.
;  4                  !
;  4                  0;
;  4
;  3                  END;
;  3
;  3              TES;
;  3          END
;  2      ELSE
;  3          BEGIN
```

```
3          save_addr_exp_desc = 0;
2          END;
2
2
2      ! Release storage and symids for the old address expression descriptor
2      !
2      IF .last_loc NEQA 0
2      THEN
3          BEGIN
3          CASE .last_loc [dbg$b_aed_type] FROM dbg$k_primary_desc TO dbg$k_notype
3              OF
3              SET
3
3              [dbg$k_primary_desc] :
4                  BEGIN
4                  LOCAL
4                      OLD_PRIM_DESC : REF dbg$dhead;
4
4                  old_prim_desc = .last_loc [dbg$l_aed_value];  ! Get the primary desc
4
4
4                  ! Extract the symids
4                  !
4                  IF dbg$nget_symid (.old_prim_desc, old_symid_list, dummy)
4                  THEN
4                      dbg$sta_unlock_symid (.old_symid_list);
4
4                  ! Release the storage for the primary descriptor
4                  !
4                  dbg$nfree_desc (.old_prim_desc, dummy);
4
3                  END;
3
3              [dbg$k_perm_desc] :
4                  BEGIN
4
4                  ! Just release the storage for the permanent symbol descriptor
4                  !
4                  dbg$rel_memory (.last_loc [dbg$l_aed_value]);
4
3                  END;
3
3              [INRANGE,OUTRANGE] :
4                  BEGIN
4
4                  ! No extra storage to release
4                  !
4                  0;
4
3                  END;
3
3              TES;
3
3
3          ! Release the storage for the address expression descriptor itself and
3          ! set current loc to nil
3          !
```

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

```
GLOBAL ROUTINE DBG$NCANCEL_LOC_AND_VAL : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       When changing languages, last val and current location go away. This
!       routine does this.
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       dbg$gl_last_val         - Old debugger last value
!
!       dbg$gl_last_loc         - Old debugger last location (current loc)
!
!       dbg$gl_next_loc         - Old debugger next location ( <CR> )
!
!       last_value              - Pointer to value descriptor for last val
!
!       last_loc                - Pointer to addr exp desc for current loc
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       NONE
!
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       The meaning of current location and last value is erased for the old and
!       new debuggers.
!
!--
    BEGIN


    ! Wipe out the pseudos for the old debugger
    !
    dbg$gl_last_val = 0;
    dbg$gl_last_loc = 0;
    dbg$gl_next_loc = 0;


    ! Wipe out the pseudos for the new debugger
    !
    dbg$nsave_last_val (0);
    dbg$nsave_last_loc (0, dbg$k_notype, 0);

```

```
; 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
```

```
GLOBAL ROUTINE DBG$NPERM_SYM_INT (REG_NUM, PERM_SYM_PTR, PATH_NAME_VECT, MESSAGE_VECT) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       A kernel DEBUG routine to obtain a permanent symbol descriptor for the
!       DEBUG permanent symbols R0-R11, AP, PC, FP, and PSL.
!
!       A permanent symbol descriptor contains information which:
!
!       1) - identifies the symbol by register number
!
!       2) - contains a virtual address which represents the location at which
!            the debugger has stored the value of the register
!
! FORMAL PARAMETERS:
!
!       reg_num         - A longword containing an unsigned integer between 0 and 16,
!                         inclusive. This integer identifies the permanent symbol
!                         to be found according to the following code:
!
!                          dbg$k_r0 (200)        - R0
!
!                          dbg$k_r1 (201)        - R1
!
!                          dbg$k_r2 (202)        - R2
!
!                          dbg$k_r3 (203)        - R3
!
!                          dbg$k_r3 (204)        - R4
!
!                          dbg$k_r5 (205)        - R5
!
!                          dbg$k_r6 (206)        - R6
!
!                          dbg$k_r7 (207)        - R7
!
!                          dbg$k_r8 (208)        - R8
!
!                          dbg$k_r9 (209)        - R9
!
!                          dbg$k_r10 (210)       - R10
!
!                          dbg$k_r11 (211)       - R11
!
!                          dbg$k_ap (212)        - R12 (AP)
!
!                          dbg$k_fp (213)        - R13 (FP)
!
!                          dbg$k_sp (214)        - R14 (SP)
!
!                          dbg$k_pc (215)        - R15 (PC)
!
!                          dbg$k_psl (216)       - PSL
!
!       perm_sym_ptr    - The address of a longword to contain the address of a
!                         permanent symbol descriptor upon return. This permanent
```

```
1 !                         symbol descriptor consists of nine byte block. The first
1 !                         byte contains the register id number as input in reg_num.
1 !                         The next four bytes contain the address of the stored
1 !                         value of the register. The next four bytes are reserved
1 !                         for future use and will be set to 0 in this release.
1 !
1 !       path_name_vect - Not used in this release. Should be a longword containing
1 !                         0. A future release of DEBUG will allow pathname qual-
1 !                         ification of registers.
1 !
1 !       message_vect    - The address of a longword to contain the address of a
1 !                         message argument vector as described on page 4-119 of
1 !                         the VAX/VMS system reference, volume 1A.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       In case of a success return, a permanent symbol descriptor is constructed
1 !       and returned.
1 !
1 !       In case of a severe error return, a message argument vector is constructed
1 !       and returned.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An usigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS (1) - Success. Permanent symbol descriptor constructed
1 !                           and returned.
1 !
1 !       STS$K_SEVERE  (4) - Failure. Permanent symbol not interpreted. Message
1 !                           argument vector constructed and returned.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2     BEGIN
2
2     LOCAL
2         PERM_SYM_DESC : REF dbg$permsd;
2
2
2     ! Create storage for the permanent symbol descriptor
2     !
2     perm_sym_desc = dbg$get_tempmem(dbg$k_permsd_size);
2
2
2     ! copy the register id over to the permanent symbols descriptor and make sure
2     ! that the pathname pointer field is 0.
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  00          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$:    .WORD   2$-1$,-                                     :
     0062           005C       0056           0050     0002D                3$-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