```
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
```

**FILE**ID**DBGLANVEC

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  LL           AAAAAA    NN      NN  VV      VV  EEEEEEEEEE    CCCCCCCC
DDDDDDDD    BBBBBBBB      GGGGGGGG  LL           AAAAAA    NN      NN  VV      VV  EEEEEEEEEE    CCCCCCCC
DD      DD  BB      BB  GG          LL          AA    AA  NN      NN  VV      VV  EE            CC
DD      DD  BB      BB  GG          LL          AA    AA  NN      NN  VV      VV  EE            CC
DD      DD  BB      BB  GG          LL          AA    AA  NNNN    NN  VV      VV  EE            CC
DD      DD  BB      BB  GG          LL          AA    AA  NNNN    NN  VV      VV  EE            CC
DD      DD  BBBBBBBB    GG          LL          AA    AA  NN  NN  NN  VV      VV  EEEEEEEE      CC
DD      DD  BBBBBBBB    GG          LL          AA    AA  NN  NN  NN  VV      VV  EEEEEEEE      CC
DD      DD  BB      BB  GG  GGGGGG  LL          AAAAAAAAAA  NN    NNNN  VV      VV  EE            CC
DD      DD  BB      BB  GG  GGGGGG  LL          AAAAAAAAAA  NN    NNNN  VV      VV  EE            CC
DD      DD  BB      BB  GG      GG  LL          AA    AA  NN      NN    VV  VV    EE            CC                ....
DD      DD  BB      BB  GG      GG  LL          AA    AA  NN      NN    VV  VV    EE            CC                ....
DDDDDDDD    BBBBBBBB      GGGGGG    LLLLLLLLLL  AA    AA  NN      NN      VV      EEEEEEEEEE    CCCCCCCC      ....
DDDDDDDD    BBBBBBBB      GGGGGG    LLLLLLLLLL  AA    AA  NN      NN      VV      EEEEEEEEEE    CCCCCCCC      ....

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

```
;     1          0001  0 MODULE DBGLANVEC (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1 !
;     5          0005  1 !**************************************************************************
;     6          0006  1 !*                                                                        *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                  *
;    10          0010  1 !*                                                                        *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12          0012  1 !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE        *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    15          0015  1 !*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY      *
;    16          0016  1 !*  TRANSFERRED.                                                           *
;    17          0017  1 !*                                                                        *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT       *
;    20          0020  1 !*  CORPORATION.                                                           *
;    21          0021  1 !*                                                                        *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS   *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.               *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !**************************************************************************
;    27          0027  1 !
;    28          0028  1 !
;    29          0029  1 ! WRITTEN BY
;    30          0030  1 !       Bruce Olsen     July, 1980
;    31          0031  1 !
;    32          0032  1 ! REWRITTEN BY
;    33          0033  1 !       Rich Title      July, 1983
;    34          0034  1 !
;    35          0035  1 ! MODULE FUNCTION
;    36          0036  1 !       This module contains several miscellaneous routines for
;    37          0037  1 !       manipulating descriptors. The name of the module is a holdover
;    38          0038  1 !       from the days when each language had its own Primary and Value
;    39          0039  1 !       Descriptors. At that time, this module had routines which
;    40          0040  1 !       did a CASE on the language, and called the appropriate language
;    41          0041  1 !       routine. Now that we have common Primary and Value descriptors
;    42          0042  1 !       for all languages, this is no longer necessary. But the routines
;    43          0043  1 !       for copying descriptors, deleting descriptors, and so on,
;    44          0044  1 !       still reside in this module.
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY
;    48          0048  1 !       R. Title        Aug 1982        Put in code to check for implementationm
;    49          0049  1 !                                       level = 3, so that we can test new support
;    50          0050  1 !                                       for PASCAL, PLI, and COBOL.
;    51          0051  1 !       R. Title        Aug 1982        Added comments to each routine so that
;    52          0052  1 !                                       the description now says what the routine
;    53          0053  1 !                                       does, instead of just saying "see the
;    54          0054  1 !                                       language-specific routines".
;    55          0055  1 !       R. Title        Mar 1983        Removed all of the "level 2" PASCAL,
;    56          0056  1 !                                       PL/I and COBOL code.
;    57          0057  1 !
```

```
;    58          0058  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    59          0192  1
;    60          0193  1 FORWARD ROUTINE
;    61          0194  1     DBG$NGET_LVAL,
;    62          0195  1     DBG$NGET_TYPE,
;    63          0196  1     DBG$NMAKE_VAL_DESC,
;    64          0197  1     DBG$NTYPE_CONV,
;    65          0198  1     DBG$NSYMBOLIZE,
;    66          0199  1     DBG$NGET_PAGES,
;    67          0200  1     DBG$NGET_LENGTH,
;    68          0201  1     DBG$NCOPY_DESC,
;    69          0202  1     COPY_DESC_HANDLER,
;    70          0203  1     DBG$NFREE_DESC,
;    71          0204  1     DBG$NGET_SYMID,
;    72          0205  1     DBG$NINITIALIZE: NOVALUE;
```

```
;    74          0206  1 EXTERNAL ROUTINE
;    75          0207  1     DBG$DATA_LENGTH,                    ! Obtain length from VMS descriptor
;    76          0208  1     DBG$EVAL_LANG_OPERATOR,             ! Evaluate operator expressions in
;    77          0209  1                                         !     current language
;    78          0210  1     DBG$GET_MEMORY,                     ! Allocate permanent memory
;    79          0211  1     DBG$GET_TEMPMEM,                    ! Allocate temporary memory
;    80          0212  1     DBG$MAKE_VMS_DESC,                  ! Convert Primary Descriptor to
;    81          0213  1                                         !     VMS descriptor
;    82          0214  1     DBG$PRIM_TO_VAL,                    ! Convert Primary Descriptor to
;    83          0215  1                                         !     Value Descriptor
;    84          0216  1     DBG$PRINT_AGGREGATE: NOVALUE,       ! Output an aggregate object
;    85          0217  1     DBG$PRINT_IDENTIFIER,               ! Replacement for DBG$NSYMBOLIZE -
;    86          0218  1                                         !     prints an identifier.
;    87          0219  1     DBG$PRINT_VALUE: NOVALUE,           ! Print a value descriptor.
;    88          0220  1     DBG$REL_MEMORY: NOVALUE;            ! Release memory
;    89          0221  1
;    90          0222  1 EXTERNAL
;    91          0223  1     DBG$GB_LANGUAGE : BYTE,             ! Language code for current language
;    92          0224  1     DBG$GL_CONVERT_TOKEN,               ! Pointer to CONVERT token
;    93          0225  1     DBG$GL_DEPOSIT_TOKEN;               ! Pointer to DEPOSIT token
;    94          0226  1
;    95          0227  1 LITERAL
;    96          0228  1     MIN_LANGUAGE_CODE = MIN (DBG$K_PLI, DBG$K_PASCAL, DBG$K_COBOL),
;    97          0229  1     MAX_LANGUAGE_CODE = MAX (DBG$K_PLI, DBG$K_PASCAL, DBG$K_COBOL);
;    98          0230  1
;    99          0231  1 OWN
;   100          0232  1         COPY_DESC_HEAD;             ! Points to the header of a copied descriptor,
;   101          0233  1                                     ! if we are copying the descriptor into
;   102          0234  1                                     ! permanent memory. This is used by
;   103          0235  1                                     ! COPY_DESC_HANDLER.
;   104          0236  1
```

```
;  106          0237  1 GLOBAL ROUTINE DBG$NGET_LVAL (PRIM_DESC, PARAM2, PARAM3) =
;  107          0238  1 !
;  108          0239  1 ! FUNCTIONAL DESCRIPTION:
;  109          0240  1 !
;  110          0241  1 !       Obtains a symbol's lvalue using the primary descriptor for that
;  111          0242  1 !       symbol. Note that most types of named constants do not have an
;  112          0243  1 !       lvalue. The debugger gives special treatment to named constants
;  113          0244  1 !       which have read only memory allocated to contain their value.
;  114          0245  1 !
;  115          0246  1 !       This routine is still called from DBGEXC,
;  116          0247  1 !       in the process of displaying "old value", "new value" on watchpoints.
;  117          0248  1 !       This routine can thus go away when DBGEXC is replaced by DBGEVENT.
;  118          0249  1 !
;  119          0250  1 ! FORMAL PARAMETERS:
;  120          0251  1 !
;  121          0252  1 !       prim_desc       - A longword which contains the address of a primary descriptor
;  122          0253  1 !
;  123          0254  1 !       param2          - The address of a quadword to contain the lvalue of the
;  124          0255  1 !                         entity described by the primary descriptor and the bit
;  125          0256  1 !                         offset, if any. The byte address wi'l be contained in
;  126          0257  1 !                         in the first longword, the bit offset in the second
;  127          0258  1 !                         longword.
;  128          0259  1 !
;  129          0260  1 !       param3          - The address of a longword to contain the address of
;  130          0261  1 !                         a message argument vector as described on page 4-119
;  131          0262  1 !                         of the VAX/VMS system reference, volume 1A
;  132          0263  1 !
;  133          0264  1 ! IMPLICIT INPUTS:
;  134          0265  1 !
;  135          0266  1 !       NONE
;  136          0267  1 !
;  137          0268  1 ! IMPLICIT OUTPUTS:
;  138          0269  1 !
;  139          0270  1 !       NONE
;  140          0271  1 !
;  141          0272  1 ! ROUTINE VALUE:
;  142          0273  1 !
;  143          0274  1 !       An unsigned longword integer completion code
;  144          0275  1 !
;  145          0276  1 ! COMPLETION CODES:
;  146          0277  1 !
;  147          0278  1 !       STS$K_SUCCESS (1) - Success. The object described by the input primary
;  148          0279  1 !                           descriptor has an lvalue which is being returned.
;  149          0280  1 !
;  150          0281  1 !       STS$K_ERROR   (2) - Failure. Object does not have an lvalue.
;  151          0282  1 !
;  152          0283  1 ! SIDE EFFECTS:
;  153          0284  1 !
;  154          0285  1 !       NONE
;  155          0286  1 !
;  156          0287  1 !
;  157          0288  2   BEGIN
;  158          0289  2
;  159          0290  2   MAP
;  160          0291  2       PRIM_DESC : REF DBG$PRIMARY;    ! Points to a new style Primary
;  161          0292  2
;  162          0293  2                                       ! Descriptor.
```

```
;  163          0294  2     LOCAL
;  164          0295  2         VMS_DESC: REF DBG$STG_DESC,
;  165          0296  2         VMS_DESC_AREA: DBG$STG_DESC;
;  166          0297  2
;  167          0298  2
;  168          0299  2     IF .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC
;  169          0300  2     THEN
;  170          0301  3         BEGIN
;  171          0302  3
;  172          0303  3         ! Set up the VMS descriptor.
;  173          0304  3         !
;  174          0305  3         VM _DESC = VMS_DESC_AREA;
;  175          0306  3
;  176          0307  3         ! Call the routine that fills in the VMS descriptor.
;  177          0308  3         !
;  178          0309  3         DBG$MAKE_VMS_DESC (.PRIM_DESC, .VMS_DESC);
;  179          0310  3         END
;  180          0311  3
;  181          0312  3     ! Value descriptor or volatile value descriptor - we already
;  182          0313  3     ! have a VMS descriptor.
;  183          0314  3     !
;  184          0315  2     ELSE IF .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_VALUE_DESC
;  185          0316  2     OR .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC
;  186          0317  2     THEN
;  187          0318  2         VMS_DESC = PRIM_DESC [DBG$A_VALUE_VMSDESC]
;  188          0319  2
;  189          0320  2     ! We do not expect any other kind of descriptor.
;  190          0321  2     !
;  191          0322  2     ELSE
;  192          0323  2         $DBG_ERROR ('DBGLANVEC\DBG$NGET_LVAL unknown descriptor kind');
;  193          0324  2
;  194          0325  2     ! Fill in the output parameter to point to the
;  195          0326  2     ! (byte address, bit offset) quadword in the VMS descriptor.
;  196          0327  2     !
;  197          0328  2     .PARAM2     = .VMS_DESC[DSC$A_POINTER];
;  198          0329  2     .PARAM2 + 4 = .VMS_DESC[DSC$L_POS];
;  199          0330  2     RETURN STS$K_SUCCESS;
;  200          0331  1     END;


                                                                .TITLE  DBGLANVEC
                                                                .IDENT  \V04-000\

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

24 47 42 44 5C 43 45 56 4E 41 4C 47 42 44 2F  00000 P.AAA:     .ASCII  \/DBGLANVEC\<92>\DBG$NGET_LVAL unknown d\   ;
6F 6E 6B 6E 75 20 4C 41 56 4C 5F 54 45 47 4E  0000F                                                            ;
                              64 20 6E 77     0001E                                                            ;
   64 6E 69 6B 20 72 6F 74 70 69 72 63 73 65  00022             .ASCII  \escriptor kind\                       ;

                                                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                                                00000 COPY_DESC_HEAD:
                                                                .BLKB   4

                                                                .EXTRN  DBG$DATA_LENGTH
                                                                .EXTRN  DBG$EVAL_LANG_OPERATOR
```

```
                                                        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
                                                        .EXTRN  DBG$MAKE_VMS_DESC
                                                        .EXTRN  DBG$PRIM_TO_VAL
                                                        .EXTRN  DBG$PRINT_AGGREGATE
                                                        .EXTRN  DBG$PRINT_IDENTIFIER
                                                        .EXTRN  DBG$PRINT_VALUE
                                                        .EXTRN  DBG$REL_MEMORY, DBG$GB_LANGUAGE
                                                        .EXTRN  DBG$GL_CONVERT_TOKEN
                                                        .EXTRN  DBG$GL_DEPOSIT_TOKEN

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

                                          0004 00000    .ENTRY  DBG$NGET_LVAL, Save R2                  ; 0237
                                       5E               SUBL2   #12, SP                                 ;
00000079  8F     04   BC                08  0C  C2 00002    CMPZV   #16, #8, @PRIM_DESC, #121               ; 0299
                                       08  10  ED 00005    BNEQ    1$                                  ;
                                       52  11  12 0000F    MOVAB   VMS_DESC_AREA, VMS_DESC             ; 0305
                                           6E  9E 00011    PUSHL   VMS_DESC                            ; 0309
                                           52  DD 00014    PUSHL   PRIM_DESC                           ;
                                    04     AC  DD 00016    CALLS   #2, DBG$MAKE_VMS_DESC               ;
                 00000000G  00             02  FB 00019    BRB     4$                                  ; 0299
0000007A  8F     04   BC                08  10  ED 00022 1$:  CMPZV   #16, #8, @PRIM_DESC, #122          ; 0315
                                           0C  13 0002C    BEQL    2$                                  ;
00000083  8F     04   BC                08  10  ED 0002E    CMPZV   #16, #8, @PRIM_DESC, #131           ; 0316
                                           07  12 00038    BNEQ    3$                                  ;
                 52        04   AC          14  C1 0003A 2$:  ADDL3   #20, PRIM_DESC, VMS_DESC           ; 0318
                                           15  11 0003F    BRB     4$                                  ;
                                00000000'  EF  9F 00041 3$:  PUSHAB  P.AAA                               ; 0323
                                           01  DD 00047    PUSHL   #1                                  ;
                                00028362   8F  DD 00049    PUSHL   #164706                             ;
                 00000000G  00             03  FB 0004F    CALLS   #3, LIB$SIGNAL                      ;
                            50       08   AC  D0 00056 4$:  MOVL    PARAM2, R0                          ; 0328
                            60       04   A2  7D 0005A    MOVQ    4(VMS_DESC), (R0)                   ;
                            50            01  D0 0005E    MOVL    #1, R0                              ; 0330
                                          04 00061        RET                                         ; 0331

; Routine Size:  98 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  202     0332  1 GLOBAL ROUTINE DBG$NGET_TYPE (PRIM_DESC, PARAM2, PARAM3) =
;  203     0333  1 !
;  204     0334  1 ! FUNCTIONAL DESCRIPTION:
;  205     0335  1 !
;  206     0336  1 !       Uses a symbol's primary descriptor to return type information. The
;  207     0337  1 !       types recognized are limited to three:
;  208     0338  1 !
;  209     0339  1 !       1)      - type named constant and instruction
;  210     0340  1 !                 (lexical entities, labels)
;  211     0341  1 !
;  212     0342  1 !       2)      - type named constant and
;  213     0343  1 !                 noinstruction (symbolic literals)
;  214     0344  1 !
;  215     0345  1 !       3)      - type other
;  216     0346  1 !
;  217     0347  1 !       This routine is still called from DBGEXC.
;  218     0348  1 !       It can go away when we convert over to the new DBGEVENT.
;  219     0349  1 !
;  220     0350  1 ! FORMAL PARAMETERS:
;  221     0351  1 !
;  222     0352  1 !       prim_desc       - A longword containing the address of a primary descriptor
;  223     0353  1 !
;  224     0354  1 !       param2          - The address of a longword to contain an unsigned integer
;  225     0355  1 !                         encoding of the symbol's type as follows:
;  226     0356  1 !
;  227     0357  1 !               dbg$k_nc_instruction (125)      - named constant, instruction
;  228     0358  1 !
;  229     0359  1 !               dbg$k_nc_other (126)            - named constant, noinstruction
;  230     0360  1 !
;  231     0361  1 !               dbg$k_other (127)               - other
;  232     0362  1 !
;  233     0363  1 !       param3          - The address of a longword to contain the address of
;  234     0364  1 !                         a message argument vector as described on page 4-119
;  235     0365  1 !                         of the VAX/VMS system reference, volume 1A
;  236     0366  1 !
;  237     0367  1 ! IMPLICIT INPUTS:
;  238     0368  1 !
;  239     0369  1 !       NONE
;  240     0370  1 !
;  241     0371  1 ! IMPLICIT OUTPUTS:
;  242     0372  1 !
;  243     0373  1 !       In case of a severe error return, a message argument vector is constructed
;  244     0374  1 !       from dynamic storage and returned.
;  245     0375  1 !
;  246     0376  1 ! ROUTINE VALUE:
;  247     0377  1 !
;  248     0378  1 !       An unsigned integer longword completion code
;  249     0379  1 !
;  250     0380  1 ! COMPLETION CODES:
;  251     0381  1 !
;  252     0382  1 !       STS$K_SUCCESS (1) - Success. Type information recovered and returned.
;  253     0383  1 !
;  254     0384  1 !       STS$K_SEVERE  (4) - Failure. No type information recovered. Message
;  255     0385  1 !                           argument vector constructed and returned.
;  256     0386  1 !
;  257     0387  1 ! SIDE EFFECTS:
;  258     0388  1 !
```

```
;  259          0389  1 !       NONE
;  260          0390  1 !
;  261          0391  2     BEGIN
;  262          0392  2
;  263          0393  2     ! For now, always return "OTHER". This may not be completely
;  264          0394  2     ! correct - we will fix it up later.
;  265          0395  2     !
;  266          0396  2     .PARAM2 = DBG$K_OTHER;
;  267          0397  2     RETURN STS$K_SUCCESS;
;  268          0398  1     END;


                                        0000 00000          .ENTRY  DBG$NGET_TYPE, Save nothing                     ; 0332
                            08   BC     7F   8F   9A 00002  MOVZBL  #127, @PARAM2                                   ; 0396
                                 50          01   D0 00007  MOVL    #1, R0                                          ; 0397
                                                  04 0000A  RET                                                     ; 0398

; Routine Size:  11 bytes,    Routine Base:  DBG$CODE + 0062
```

```
;  270     0399  1 GLOBAL ROUTINE DBG$NMAKE_VAL_DESC (PRIM_DESC, PARAM2, PARAM3, PARAM4) =
;  271     0400  1 !
;  272     0401  1 !++
;  273     0402  1 ! FUNCTIONAL DESCRIPTION:
;  274     0403  1 !
;  275     0404  1 !       Translates language specific primary descriptors to language specific
;  276     0405  1 !       value descriptors. This routine should be able to use the symbol table
;  277     0406  1 !       access routines and the information contained within the primary descriptor
;  278     0407  1 !       to construct a descriptor which represents a 'value materialization' for
;  279     0408  1 !       the object represented by the input primary descriptor.
;  280     0409  1 !
;  281     0410  1 !       Note that this routine must be able to use life-time, invocation, and
;  282     0411  1 !       generation information to produce an accurate value descriptor of the
;  283     0412  1 !       input object, or to decide when the value of an object cannot be
;  284     0413  1 !       materialized (such as when the user's PC is not within the scope of
;  285     0414  1 !       a dynamic variable).
;  286     0415  1 !
;  287     0416  1 !       Value descriptors produced by this routine must be marked (within the
;  288     0417  1 !       type field of the language independent header block) as to whether
;  289     0418  1 !       they are non-volatile (dsc$k_value_desc) or volatile (dsc$k_v_value_desc).
;  290     0419  1 !       Volatile value descriptors will NOT be stored to represent '\', 'last value'.
;  291     0420  1 !
;  292     0421  1 !       Since value descriptors may be used as target descriptors ( as input to
;  293     0422  1 !       dbg$npli_type_conv ), some provision must be made for incorporating
;  294     0423  1 !       a value pointer field within the value descriptor. This type of value
;  295     0424  1 !       descriptor is loosely defined as a volatile type.
;  296     0425  1 !
;  297     0426  1 !       This routine is still called from DBGEXC in the process of giving
;  298     0427  1 !       watchpoint display. It can thus go away when DBGEXC is replaced
;  299     0428  1 !       by DBGEVENT.
;  300     0429  1 !
;  301     0430  1 !       This routine call a language-specific routine based on the language
;  302     0431  1 !       code in the descriptor header.
;  303     0432  1 !
;  304     0433  1 ! FORMAL PARAMETERS:
;  305     0434  1 !
;  306     0435  1 !       prim_desc       - A longword containing the address of a primary descriptor
;  307     0436  1 !
;  308     0437  1 !       param2          - A longword containing boolean true or false. When true,
;  309     0438  1 !                         the caller is requesting the construction of a value
;  310     0439  1 !                         descriptor that can be used as a target descriptor for
;  311     0440  1 !                         the type converter. The resulting value must therefore
;  312     0441  1 !                         contain a pointer to the value of the entity described
;  313     0442  1 !                         by the input primary descriptor. Presumably, such a
;  314     0443  1 !                         value descriptor will be of volatile type.
;  315     0444  1 !
;  316     0445  1 !       param3          - The address of a longword to contain the address of the
;  317     0446  1 !                         resulting value descriptor
;  318     0447  1 !
;  319     0448  1 !       param4          - The address of a longword to contain the address of a
;  320     0449  1 !                         message argument vector as described on page 4-119 of
;  321     0450  1 !                         the VAX/VMS system reference, volume 1A
;  322     0451  1 !
;  323     0452  1 ! IMPLICIT INPUTS:
;  324     0453  1 !
;  325     0454  1 !       Depends on the language-specific routine.
;  326     0455  1 !
```

```
;  327     0456  1 ! IMPLICIT OUTPUTS:
;  328     0457  1 !
;  329     0458  1 !       In case of a success return, the resulting value descriptor must be
;  330     0459  1 !       constructed from dynamic storage and returned.
;  331     0460  1 !
;  332     0461  1 !       In case of a severe error return, a message argument vector must be
;  333     0462  1 !       constructed from dynamic storage and returned.
;  334     0463  1 !
;  335     0464  1 ! ROUTINE VALUE:
;  336     0465  1 !
;  337     0466  1 !       An unsigned integer longword completion code
;  338     0467  1 !
;  339     0468  1 ! COMPLETION CODES:
;  340     0469  1 !
;  341     0470  1 !       STS$K_SUCCESS (1) - Success. Value descriptor constructed and returned.
;  342     0471  1 !
;  343     0472  1 !       STS$K_SEVERE  (4) - Failure. Value descriptor not constructed. Message
;  344     0473  1 !                           argument vector constructed and returned.
;  345     0474  1 !
;  346     0475  1 ! SIDE EFFECTS:
;  347     0476  1 !
;  348     0477  1 !       NONE
;  349     0478  1 !
;  350     0479  2     BEGIN
;  351     0480  2
;  352     0481  2     MAP
;  353     0482  2         PRIM_DESC: REF DBG$PRIMARY;
;  354     0483  2
;  355     0484  2     ! Don't convert to value desc if the primary is an aggregate.
;  356     0485  2     !
;  357     0486  2     IF .PRIM_DESC [DBG$V_DHDR_AGGR]
;  358     0487  2     THEN
;  359     0488  2         .PARAM3 = .PRIM_DESC
;  360     0489  2     ELSE
;  361     0490  2         IF NOT DBG$PRIM_TO_VAL (
;  362     0491  2                 .PRIM_DESC,
;  363     0492  2                 (IF .PARAM2 THEN DBG$K_V_VALUE_DESC ELSE DBG$K_VALUE_DESC),
;  364     0493  2                 .PARAM3)
;  365     0494  2         THEN
;  366     0495  2             $DBG_ERROR ('DBGLANVEC\DBG$NMAKE_VAL_DESC bad return code from PRIM_TO_VAL');
;  367     0496  2     RETURN STS$K_SUCCESS;
;  368     0497  1     END;


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

24 47 42 44 5C 43 45 56 4E 41 4C 47 42 44 3D  00030 P.AAB:  .ASCII  \=DBGLANVEC\<92>\DBG$NMAKE_VAL_DESC bad \   ;
20 43 53 45 44 5F 4C 41 56 5F 45 4B 41 4D 4E  0003F                                                           ;
                                  20 64 61 62  0004E                                                           ;
6F 72 66 20 65 64 6F 63 20 6E 72 75 74 65 72  00052          .ASCII  \return code from PRIM_TO_VAL\            ;
      4C 41 56 5F 4F 54 5F 4D 49 52 50 20 6D  00061                                                           ;


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

```
                                        0000 00000          .ENTRY  DBG$NMAKE_VAL_DESC, Save nothing                ; 0399
                          50        04  AC   D0 00002          MOVL    PRIM_DESC, R0                                   ; 0486
                          06        04  A0   E9 00006          BLBC    4(R0), 1$
                    0C    BC            50   D0 0000A          MOVL    R0, @PARAM3                                     ; 0488
                                        32   11 0000E          BRB     4$
                                  0C    AC   DD 00010 1$:      PUSHL   PARAM3                                          ; 0493
                          06        08  AC   E9 00013          BLBC    PARAM2, 2$                                      ; 0492
                          7E        83  8F   9A 00017          MOVZBL  #131, -(SP)
                                        04   11 0001B          BRB     3$
                          7E        7A  8F   9A 0001D 2$:      MOVZBL  #122, -(SP)
                                        50   DD 00021 3$:      PUSHL   R0                                              ; 0491
                00000000G 00            03   FB 00023          CALLS   #3, DBG$PRIM_TO_VAL
                          15            50   E8 0002A          BLBS    R0, 4$
                              00000000' EF   9F 0002D          PUSHAB  P.AAB                                           ; 0495
                                        01   DD 00033          PUSHL   #1
                              00028362  8F   DD 00035          PUSHL   #164706
                00000000G 00            03   FB 0003B          CALLS   #3, LIB$SIGNAL
                          50            01   D0 00042 4$:      MOVL    #1, R0                                          ; 0496
                                        04   00045          RET                                                     ; 0497

; Routine Size:  70 bytes,    Routine Base:  DBG$CODE + 006D
```

```
;  370     0498  1 GLOBAL ROUTINE DBG$NTYPE_CONV (VALUE_DESC, PARAM2, PARAM3, PARAM4, PARAM5) =
;  371     0499  1 !
;  372     0500  1 ! FUNCTIONAL DESCRIPTION:
;  373     0501  1 !
;  374     0502  1 !       Performs language specific and language independent type conversions.
;  375     0503  1 !       These will be both internal-to-internal and internal-to-external in
;  376     0504  1 !       nature. Target may be described by either language
;  377     0505  1 !       specific value descriptor or a subset of VAX standard descriptors.
;  378     0506  1 !       The latter category includes the following:
;  379     0507  1 !
;  380     0508  1 !       dsc$k_dtype_v
;  381     0509  1 !
;  382     0510  1 !       dsc$k_dtype_b, dsc$k_dtype_bu
;  383     0511  1 !
;  384     0512  1 !       dsc$k_dtype_w, dsc$k_dtype_wu
;  385     0513  1 !
;  386     0514  1 !       dsc$k_dtype_l, dsc$k_dtype_lu
;  387     0515  1 !
;  388     0516  1 !       dsc$k_dtype_q, dsc$k_dtype_qu
;  389     0517  1 !
;  390     0518  1 !       dsc$k_dtype_f, dsc$k_dtype_d
;  391     0519  1 !
;  392     0520  1 !       dsc$k_dtype_t
;  393     0521  1 !
;  394     0522  1 !       The source descriptor must be a language specific value descriptor.
;  395     0523  1 !
;  396     0524  1 !       Note that this routine will be used to obtain the 'printable' (external)
;  397     0525  1 !       value of the source as the result of EXAMINE commands.
;  398     0526  1 !
;  399     0527  1 !       This routine is still called from a couple of places; one is to
;  400     0528  1 !       convert the expression in an IF or a WHILE command to boolean;
;  401     0529  1 !       another is to display the value of watchpoints in "old value",
;  402     0530  1 !       "new value" displays. (This second use of this routine will go
;  403     0531  1 !       away when DBGEVENT replaces DBGEXC.)
;  404     0532  1 !
;  405     0533  1 ! FORMAL PARAMETERS:
;  406     0534  1 !
;  407     0535  1 !       value_desc      - A longword which contains the address of
;  408     0536  1 !                         a language specific value descriptor
;  409     0537  1 !
;  410     0538  1 !       param2          - A longword containing an integer encoding of the radix
;  411     0539  1 !                         to be used when converting to a 'printable' value:
;  412     0540  1 !
;  413     0541  1 !               dbg$k_default (1)       - source language default radix
;  414     0542  1 !
;  415     0543  1 !               dbg$k_binary (2)        - binary radix
;  416     0544  1 !
;  417     0545  1 !               dbg$k_octal (8)         - octal radix
;  418     0546  1 !
;  419     0547  1 !               dbg$k_decimal (10)      - decimal radix
;  420     0548  1 !
;  421     0549  1 !               dbg$k_hex (16)          - hexadecimal radix
;  422     0550  1 !
;  423     0551  1 !                       Note that this parameter is significant ONLY when the
;  424     0552  1 !                       object described by the source descriptor is to be
;  425     0553  1 !                       converted to external format. A request for a binary,
;  426     0554  1 !                       octal, or hex 'printable' value means to consider the
```

```
;  427     0555  1 !                value of source as a bit pattern to be translated to
;  428     0556  1 !                special characters. In this sense, the type of the source
;  429     0557  1 !                value is not significant - only the length. Values will
;  430     0558  1 !                therefore be displayed as unsigned integers within the
;  431     0559  1 !                specified radix. Values will be left-extended to nibble
;  432     0560  1 !                boundaries.
;  433     0561  1 !
;  434     0562  1 !        param3  - A longword containing an unsigned integer encoding of the
;  435     0563  1 !                  type of information contained within the target parameter:
;  436     0564  1 !
;  437     0565  1 !                dbg$k_vax_desc (130)      - target contains the address of a
;  438     0566  1 !                                            VAX standard descriptor
;  439     0567  1 !
;  440     0568  1 !                                            Note: The caller of dbg$nxxx_type_conv
;  441     0569  1 !                                            must assure that the dsc$a_pointer
;  442     0570  1 !                                            field of the descriptor contains the
;  443     0571  1 !                                            address of an appropriately large
;  444     0572  1 !                                            block of storage.
;  445     0573  1 !
;  446     0574  1 !                dbg$k_value_desc (122)    - target contains the address of a
;  447     0575  1 !                                            language specific value descriptor.
;  448     0576  1 !                                            The type convertor deposits the
;  449     0577  1 !                                            value of Source into the address of
;  450     0578  1 !                                            the value in Target.
;  451     0579  1 !
;  452     0580  1 !                dbg$k_external_desc (129) - target contains the address of
;  453     0581  1 !                                            a VAX standard string descriptor.
;  454     0582  1 !                                            This is a request to convert to
;  455     0583  1 !                                            'printable' format. Conversion must
;  456     0584  1 !                                            include check for unprintable characters.
;  457     0585  1 !
;  458     0586  1 !        param4  - A longword which contains the address of either a VAX
;  459     0587  1 !                  standard descriptor, or a language specific value descriptor
;  460     0588  1 !
;  461     0589  1 !        param5  - The address of a longword to contain the address of
;  462     0590  1 !                  a message argument vector as described on page 4-119 of
;  463     0591  1 !                  the VAX/VMS system reference, volume 1A
;  464     0592  1 !
;  465     0593  1 ! IMPLICIT INPUTS:
;  466     0594  1 !
;  467     0595  1 !        NONE
;  468     0596  1 !
;  469     0597  1 ! IMPLICIT OUTPUTS:
;  470     0598  1 !
;  471     0599  1 !        When this routine is called to obtain the 'printable' (external) value
;  472     0600  1 !        of the source object, the target will contain the address of a VAX
;  473     0601  1 !        standard string descriptor with length and pointer fields set to 0.
;  474     0602  1 !        Dynamic storage must be obtained to contain the resulting ascii string.
;  475     0603  1 !
;  476     0604  1 !        In all other cases, this routine is not required to allocate storage to
;  477     0605  1 !        contain the resulting value of a conversion request. Targets which are
;  478     0606  1 !        described by VAX standard descriptors MUST contain the address of a
;  479     0607  1 !        block of storage (the dsc$a_pointer field)
;  480     0608  1 !        in which the resulting value of the conversion will be stored.
;  481     0609  1 !
;  482     0610  1 !        Dynamic storage must be used to construct the message argument vector
;  483     0611  1 !        upon a severe error return.
```

```
;  484          0612  1 !
;  485          0613  1 ! ROUTINE VALUE:
;  486          0614  1 !
;  487          0615  1 !       unsigned integer longword completion code
;  488          0616  1 !
;  489          0617  1 ! COMPLETION CODES:
;  490          0618  1 !
;  491          0619  1 !       STS$K_SUCCESS (1) - Success. Conversion performed.
;  492          0620  1 !
;  493          0621  1 !       STS$K_SEVERE  (4) - Failure. No conversion. Message argument vector
;  494          0622  1 !                           constructed and returned.
;  495          0623  1 !
;  496          0624  1 ! SIDE EFFECTS:
;  497          0625  1 !
;  498          0626  1 !       Informational messages such as string and number truncation may be
;  499          0627  1 !       issued during processing.
;  500          0628  1 !
;  501          0629  1
;  502          0630  2     BEGIN
;  503          0631  2
;  504          0632  2     SELECTONE .PARAM3 OF
;  505          0633  2         SET
;  506          0634  2
;  507          0635  2         ! One place this routine is called is in the processing of the
;  508          0636  2         ! IF, WHILE, and INCR commands, in order to convert the given
;  509          0637  2         ! value to a type understood by the command.
;  510          0638  2         ! In these cases, the third parameter is DBG$K_VAX_DESC and
;  511          0639  2         ! the fourth parameter is a pointer to a VAX standard descriptor.
;  512          0640  2         !
;  513          0641  2         [DBG$K_VAX_DESC] :
;  514          0642  3             BEGIN
;  515          0643  3             LOCAL
;  516          0644  3                 V_VAL_DESC: REF DBG$VALDESC;
;  517          0645  3
;  518          0646  3             ! Build a volatile value descriptor around the given VAX
;  519          0647  3             ! standard descriptor.
;  520          0648  3             !
;  521          0649  3             V_VAL_DESC = DBG$GET_TEMPMEM (DBG$K_VALDESC_BASE_SIZE+4);
;  522          0650  3             CA$MOVE (12, .VALUE_DESC, .V_VAL_DESC);
;  523          0651  3             V_VAL_DESC[DBG$B_DHDR_TYPE] = DBG$K_V_VALUE_DESC;
;  524          0652  3             V_VAL_DESC[DBG$W_DHDR_LENGTH] = 4 * (DBG$K_VALDESC_BASE_SIZE+4);
;  525          0653  3             CA$MOVE (12, .PARAM4, V_VAL_DESC[DBG$A_VALUE_VMSDESC]);
;  526          0654  3
;  527          0655  3             ! Call the EVAL_LANG_OPERATOR routine to do the conversion.
;  528          0656  3             !
;  529          0657  3             DBG$EVAL_LANG_OPERATOR (
;  530          0658  3                         DBG$GL_CONVERT_TOKEN,
;  531          0659  3                         .VALUE_DESC,
;  532          0660  3                         .V_VAL_DESC);
;  533          0661  2             END;
;  534          0662  2
;  535          0663  2         ! Another case is during the output of watchpoints in
;  536          0664  2         ! "old value", "new value".
;  537          0665  2         !
;  538          0666  2         [DBG$K_EXTERNAL_DESC] :
;  539          0667  3             BEGIN
;  540          0668  3             MAP
```

```
;  541          0669  3                VALUE_DESC: REF DBG$PRIMARY;
;  542          0670  3
;  543          0671  3            ! Check for aggregate.
;  544          0672  3            !
;  545          0673  3            IF .VALUE_DESC [DBG$V_DHDR_AGGR]
;  546          0674  3            THEN
;  547          0675  3                DBG$PRINT_AGGREGATE (.VALUE_DESC, .PARAM2)
;  548          0676  3
;  549          0677  3            ELSE
;  550          0678  3
;  551          0679  3                ! Call the PRINT_VALUE routine
;  552          0680  3                !
;  553          0681  3                DBG$PRINT_VALUE (.VALUE_DESC, .PARAM2, FALSE, FALSE);
;  554          0682  3
;  555          0683  3            ! This is kind of a kludge. We fill in a -1 to PARAM5
;  556          0684  3            ! and this indicates to the caller in DBGEXC that the
;  557          0685  3            ! value has already been displayed.
;  558          0686  3            !
;  559          0687  3            .PARAM5 = -1;
;  560          0688  2            END;
;  561          0689  2
;  562          0690  2        ! I don't think there are any other cases where this routine
;  563          0691  2        ! is still used, so signal an internal DEBUG error.
;  564          0692  2        !
;  565          0693  2        [OTHERWISE] :
;  566          0694  2            $DBG_ERROR ('DBGLANVEC\DBG$NTYPE_CONV');
;  567          0695  2
;  568          0696  2        TES;
;  569          0697  2    RETURN STS$K_SUCCESS;
;  570          0698  1    END;


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

24 47 42 44 5C 43 45 56 4E 41 4C 47 42 44 18  0006E P.AAC:    .ASCII  <24>\DBGLANVEC\<92>\DBG$NTYPE_CONV\                  ;
                  56 4E 4F 43 5F 45 50 59 54 4E  0007D                                                                  ;


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

                                                  007C 00000            .ENTRY  DBG$NTYPE_CONV, Save R2,R3,R4,R5,R6           ; 0498
                                  50      0C    AC  D0 00002            MOVL    PARAM3, R0                                    ; 0632
                          00000082    8F        50  D1 00006            CMPL    R0, #150                                      ; 0641
                                                33  12 0000D            BNEQ    1$                                            ;
                                                0C  DD 0000F            PUSHL   #12                                           ; 0649
                          00000000G   00        01  FB 00011            CALLS   #1, DBG$GET_TEMPMEM                           ;
                                      56        50  D0 00018            MOVL    R0, V_VAL_DESC                                ;
                              66        04    BC  0C  28 0001B            MOVC3   #12, @VALUE_DESC, (V_VAL_DESC)                ; 0650
                                        02    A6  8F  90 00020            MOVB    #-125, 2(V_VAL_DESC)                          ; 0651
                                      66        30  B0 00025            MOVW    #48, (V_VAL_DESC)                             ; 0652
                      14  A6              10  BC  0C  28 00028            MOVC3   #12, @PARAM4, 20(V_VAL_DESC)                  ; 0653
                                                56  DD 0002E            PUSHL   V_VAL_DESC                                    ; 0660
                                                04  AC  DD 00030            PUSHL   VALUE_DESC                                    ; 0659
                                  000000000G    00  9F 00033            PUSHAB  DBG$GL_CONVERT_TOKEN                          ; 0657
                          00000000G   00        03  FB 00039            CALLS   #3, DBG$EVAL_LANG_OPERATOR                    ;
```

```
                                    48  11 00040          BRB     5$                                  ; 0632
                00000081    8F      50  D1 00042 1$:      CMPL    R0, #129                            ; 0666
                                    2A  12 00049          BNEQ    4$                                  ;
                            52  04  AC  D0 0004B          MOVL    VALUE_DESC, R2                      ; 0673
                            0E  04  A2  E9 0004F          BLBC    4(R2), 2$                           ;
                                08  AC  DD 00053          PUSHL   PARAM2                              ; 0675
                                52  DD 00056          PUSHL   R2                                  ;
                00000000G   00      02  FB 00058          CALLS   #2, DBG$PRINT_AGGREGATE             ;
                                    0E  11 0005F          BRB     3$                                  ;
                                    7E  7C 00061 2$:      CLRQ    -(SP)                               ; 0681
                                08  AC  DD 00063          PUSHL   PARAM2                              ;
                                52  DD 00066          PUSHL   R2                                  ;
                00000000G   00      04  FB 00068          CALLS   #4, DBG$PRINT_VALUE                 ;
                      14    BC      01  CE 0006F 3$:      MNEGL   #1, @PARAM5                         ; 0687
                                    15  11 00073          BRB     5$                                  ; 0632
                        00000000'   EF  9F 00075 4$:      PUSHAB  P.AAC                               ; 0694
                                    01  DD 00078          PUSHL   #1                                  ;
                        00028362    8F  DD 0007D          PUSHL   #164706                             ;
                00000000G   00      03  FB 00083          CALLS   #3, LIB$SIGNAL                      ;
                            50      01  D0 0008A 5$:      MOVL    #1, R0                              ; 0697
                                        04 0008D          RET                                         ; 0698

; Routine Size:  142 bytes,    Routine Base:  DBG$CODE + 00B3
```

```
;  572          0699  1 GLOBAL ROUTINE DBG$NSYMBOLIZE (PRIM_DESC, PARAM2, PARAM3) =
;  573          0700  1 !
;  574          0701  1 !  FUNCTION
;  575          0702  1 !        Prints the name given by the primary descriptor, in the
;  576          0703  1 !        appropriate language format. This routine actually just
;  577          0704  1 !        passes the descriptor along to the new routine
;  578          0705  1 !        DBG$PRINT_IDENTIFIER.
;  579          0706  1 !
;  580          0707  1 !  FORMAL PARAMETERS:
;  581          0708  1 !
;  582          0709  1 !        PRIM_DESC       - A longword containing the address of a language specific
;  583          0710  1 !                          primary descriptor
;  584          0711  1 !
;  585          0712  1 !        PARAM2, PARAM3  - Unknown to this routine
;  586          0713  1 !
;  587          0714  1 !  IMPLICIT INPUTS:
;  588          0715  1 !
;  589          0716  1 !        NONE
;  590          0717  1 !
;  591          0718  1 !  IMPLICIT OUTPUTS:
;  592          0719  1 !
;  593          0720  1 !        Same as the invoked routine
;  594          0721  1 !
;  595          0722  1 !  ROUTINE VALUE:
;  596          0723  1 !
;  597          0724  1 !        Same as the invoked routine
;  598          0725  1 !
;  599          0726  1 !  COMPLETION CODES:
;  600          0727  1 !
;  601          0728  1 !        Same as the invoked routine
;  602          0729  1 !
;  603          0730  1 !  SIDE EFFECTS:
;  604          0731  1 !
;  605          0732  1 !        Same as the invoked routine.
;  606          0733  1 !
;  607          0734  1 !        This routine will generate a SIGNAL upon detection of a foreign
;  608          0735  1 !        language value within the primary descriptor.
;  609          0736  1 !
;  610          0737  2     BEGIN
;  611          0738  2     DBG$PRINT_IDENTIFIER (.PRIM_DESC);
;  612          0739  2     RETURN STS$K_SUCCESS;
;  613          0740  1     END;


                                        0000 00000          .ENTRY  DBG$NSYMBOLIZE, Save nothing                    ; 0699
                                  04    AC  DD 00002        PUSHL   PRIM_DESC                                       ; 0738
                        00000000G   00      01  FB 00005   CALLS   #1, DBG$PRINT_IDENTIFIER                        ;
                                    50      01  D0 0000C   MOVL    #1, R0                                          ; 0739
                                            04 0000F        RET                                                     ; 0740

; Routine Size:  16 bytes,    Routine Base:  DBG$CODE + 0141
```

```
;  615          0741  1 GLOBAL ROUTINE DBG$NGET_PAGES (PRIM_DESC, PARAM2, PARAM3) =
;  616          0742  1
;  617          0743  1 !++
;  618          0744  1 ! FUNCTIONAL DESCRIPTION:
;  619          0745  1 !
;  620          0746  1 !       Uses a symbol's primary descriptor to construct a linked list of page
;  621          0747  1 !       numbers which reflect those pages of storage in which the symbol's
;  622          0748  1 !       rvalue is contained. Note that the pages may be non-contiguous.
;  623          0749  1 !
;  624          0750  1 !       A page number is represented by the high order 23 bits of a virtual
;  625          0751  1 !       address, with the low order 9 bits set to 0:
;  626          0752  1 !
;  627          0753  1 !       page = (virtual__address AND B'11111111111111111111111000000000')
;  628          0754  1 !
;  629          0755  1 !       At implementation level 2,
;  630          0756  1 !       This routine calls a language-specific routine depending on the language
;  631          0757  1 !       code in the header of the descriptor.
;  632          0758  1 !
;  633          0759  1 !       At implementation level 3, the descriptors are the same so the
;  634          0760  1 !       work is done right here.
;  635          0761  1 !
;  636          0762  1 ! FORMAL PARAMETERS:
;  637          0763  1 !
;  638          0764  1 !       prim_desc       - A longword containing the address of a primary descriptor
;  639          0765  1 !
;  640          0766  1 !       param2          - The address of a longword to contain the address of the
;  641          0767  1 !                         head node in the page list. Nodes in the page list
;  642          0768  1 !                         consist of blocks of two longwords each. The second
;  643          0769  1 !                         longword of the node block contains a page number on
;  644          0770  1 !                         which some portion of the symbol's rvalue resides. The
;  645          0771  1 !                         first longword of the node block contains the address
;  646          0772  1 !                         of the next node in the list. The last node in the list
;  647          0773  1 !                         should contain a 0 in this link field.
;  648          0774  1 !
;  649          0775  1 !       param3          - The address of a longword to contain the address of
;  650          0776  1 !                         a message argument vector as described on page 4-119
;  651          0777  1 !                         of the VAX/VMS system reference, volume 1A
;  652          0778  1 !
;  653          0779  1 ! IMPLICIT INPUTS:
;  654          0780  1 !
;  655          0781  1 !       NONE
;  656          0782  1 !
;  657          0783  1 ! IMPLICIT OUTPUTS:
;  658          0784  1 !
;  659          0785  1 !       In case of a success return, the page list is constructed from dynamic
;  660          0786  1 !       storage and returned.
;  661          0787  1 !
;  662          0788  1 !       In case of a severe error return, a message arguement vector is constructed
;  663          0789  1 !       and returned.
;  664          0790  1 !
;  665          0791  1 ! ROUTINE VALUE:
;  666          0792  1 !
;  667          0793  1 !       An unsigned integer longword completion code
;  668          0794  1 !
;  669          0795  1 ! COMPLETION CODES:
;  670          0796  1 !
;  671          0797  1 !       STS$K_SUCCESS (1) - Success. Page list constructed and returned.
```

```
;  672          0798  1 !
;  673          0799  1 !       STS$K_SEVERE  (4) - Failure. Page list not constructed. Message argument
;  674          0800  1 !                           vector constructed and returned.
;  675          0801  1 !
;  676          0802  1 ! SIDE EFFECTS:
;  677          0803  1 !
;  678          0804  1 !       NONE
;  679          0805  1 !
;  680          0806  2     BEGIN
;  681          0807  2
;  682          0808  2     MAP
;  683          0809  2         PRIM_DESC: REF DBG$PRIMARY;
;  684          0810  2
;  685          0811  2     LOCAL
;  686          0812  2         BIT_LENGTH,                             ! Length of data in bits
;  687          0813  2         CURRENT_BLOCK: REF DBG$LINK_NODE,       ! Pointer to current page
;  688          0814  2                                                 !    number block
;  689          0815  2         CURRENT_PAGE_ADDRESS,                   ! A page address
;  690          0816  2         END_ADDRESS,                            ! Last page address
;  691          0817  2         NEXT_BLOCK: REF DBG$LINK_NODE,          ! Pointer to the next page
;  692          0818  2                                                 !    number block
;  693          0819  2         POS,
;  694          0820  2         VMS_DESC: REF DBG$STG_DESC,
;  695          0821  2         VMS_DESC_AREA: DBG$STG_DESC;
;  696          0822  2
;  697          0823  2     ! For volatile value descriptors we already have a vms desc.
;  698          0824  2     !
;  699          0825  2     IF .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC
;  700          0826  2     OR .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_VALUE_DESC
;  701          0827  2     THEN
;  702          0828  2         VMS_DESC = PRIM_DESC [DBG$A_VALUE_VMSDESC]
;  703          0829  2     ELSE IF .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC
;  704          0830  2     THEN
;  705          0831  3         BEGIN
;  706          0832  3
;  707          0833  3         ! Turn the primary descriptor into a VMS descriptor.
;  708          0834  3         !
;  709          0835  3         VMS_DESC = VMS_DESC_AREA;
;  710          0836  3         IF NOT DBG$MAKE_VMS_DESC (.PRIM_DESC, .VMS_DESC)
;  711          0837  3         THEN
;  712          0838  3             $DBG_ERROR ('DBGLANVEC\DBG$NGET_PAGES');
;  713          0839  3         END
;  714          0840  2     ELSE
;  715          0841  2         $DBG_ERROR ('DBGLANVEC\DBG$NGET_PAGES');
;  716          0842  2
;  717          0843  2     ! The first address is given in the VMS descriptor. The end address
;  718          0844  2     ! must be computed from the bit length and the bit offset.
;  719          0845  2     !
;  720          0846  2     CURRENT_PAGE_ADDRESS = .VMS_DESC[DSC$A_POINTER] AND %X'FFFFFE00';
;  721          0847  2     BIT_LENGTH = DBG$DATA_LENGTH (.VMS_DESC);
;  722          0848  2     IF .VMS_DESC[DSC$B_CLASS] EQL DSC$K_CLASS_UBS
;  723          0849  2     THEN
;  724          0850  2         POS = .VMS_DESC[DSC$L_POS]
;  725          0851  2     ELSE
;  726          0852  2         POS = 0;
;  727          0853  2     END_ADDRESS = .VMS_DESC[DSC$A_POINTER] + (.BIT_LENGTH + .POS - 1)/8;
;  728          0854  2
```

```
;  729          0855  2
;  730          0856  2      ! Loop through the pages.
;  731          0857  2      !
;  732          0858  2      CURRENT_BLOCK = 0;
;  733          0859  2      WHILE .CURRENT_PAGE_ADDRESS LEQ .END_ADDRESS DO
;  734          0860  3          BEGIN
;  735          0861  3
;  736          0862  3          ! Allocate space for a new node. Fill in the value field
;  737          0863  3          ! and link it in to the list (the list is actually being
;  738          0864  3          ! constructed backwards). Increment CURRENT_PAGE_ADDRESS to
;  739          0865  3          ! the next page and loop.
;  740          0866  3          !
;  741          0867  3          NEXT_BLOCK = DBG$GET_TEMPMEM (DBG$K_LINK_NODE_SIZE);
;  742          0868  3          NEXT_BLOCK[DBG$L_LINK_NODE_LINK] = .CURRENT_BLOCK;
;  743          0869  3          NEXT_BLOCK[DBG$L_LINK_NODE_VALUE] = .CURRENT_PAGE_ADDRESS;
;  744          0870  3          CURRENT_BLOCK = .NEXT_BLOCK;
;  745          0871  3          CURRENT_PAGE_ADDRESS = .CURRENT_PAGE_ADDRESS + 512;
;  746          0872  2          END;
;  747          0873  2
;  748          0874  2      ! Return the address of the last block.
;  749          0875  2      !
;  750          0876  2      .PARAM2 = .CURRENT_BLOCK;
;  751          0877  2      RETURN STS$K_SUCCESS;
;  752          0878  1      END;


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

24 47 42 44 5C 43 45 56 4E 41 4C 47 42 44 18  00087 P.AAD:  .ASCII  <24>\DBGLANVEC\<92>\DBG$NGET_PAGES\          ;
                53 45 47 41 50 5F 54 45 47 4E  00096
24 47 42 44 5C 43 45 56 4E 41 4C 47 42 44 18  000A0 P.AAE:  .ASCII  <24>\DBGLANVEC\<92>\DBG$NGET_PAGES\          ;
                53 45 47 41 50 5F 54 45 47 4E  000AF


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

                                                  001C 00000    .ENTRY  DBG$NGET_PAGES, Save R2,R3,R4                ; 0741
                                              5E  0C   C2 00002         SUBL2   #12, SP                            ;
00000083    8F      04    BC                08  10   ED 00005         CMPZV   #16, #8, @PRIM_DESC, #131          ; 0825
                                                  0C   13 0000F         BEQL    1$                                 ;
0000007A    8F      04    BC                08  10   ED 00011         CMPZV   #16, #8, @PRIM_DESC, #122          ; 0826
                                                  07   12 0001B         BNEQ    2$                                 ;
                    52          04        AC  14   C1 0001D 1$:       ADDL3   #20, PRIM_DESC, VMS_DESC           ; 0828
                                                  3B   11 00022         BRB     5$                                 ;
00000079    8F      04    BC                08  10   ED 00024 2$:       CMPZV   #16, #8, @PRIM_DESC, #121          ; 0829
                                                  1A   12 0002E         BNEQ    3$                                 ;
                                              52  6E   9E 00030         MOVAB   VMS_DESC_AREA, VMS_DESC            ; 0835
                                                  52   DD 00033         PUSHL   VMS_DESC                           ; 0836
                                                  04   AC DD 00035      PUSHL   PRIM_DESC                          ;
                          00000000G           00  02   FB 00038         CALLS   #2, DBG$MAKE_VMS_DESC              ;
                                              1D  50   E8 0003F         BLBS    R0, 5$                             ;
                                  00000000'       EF   9F 00042         PUSHAB  P.AAD                              ; 0838
                                                  06   11 00048         BRB     4$                                 ;
                                  00000000'       EF   9F 0004A 3$:       PUSHAB  P.AAE                            ; 0841
                                                  01   DD 00050 4$:       PUSHL   #1                               ;
```

```
                          00028362   8F   DD 00052          PUSHL   #164706                                          ;
                00000000G   00       03   FB 00058          CALLS   #3, LIB$SIGNAL                                   ;
            54        04    A2 000001FF  8F   CB 0005F 5$:    BICL3   #511, 4(VMS_DESC), CURRENT_PAGE_ADDRESS          ; 0846
                                       52   DD 00068          PUSHL   VMS_DESC                                         ; 0847
                00000000G   00          01   FB 0006A          CALLS   #1, DBG$DATA_LENGTH                              ;
                            0D      03  A2   91 00071          CMPB    3(VMS_DESC), #13                                 ; 0848
                                        06   12 00075          BNEQ    6$                                               ;
                            51      08  A2   D0 00077          MOVL    8(VMS_DESC), POS                                 ; 0850
                                        02   11 0007B          BRB     7$                                               ;
                                        51   D4 0007D 6$:      CLRL    POS                                              ; 0852
                            50      FF A140  9E 0007F 7$:      MOVAB   -1(POS)[BIT_LENGTH], R0                          ; 0853
                            50          08   C6 00084          DIVL2   #8, R0                                           ;
            53              50      04  A2   C1 00087          ADDL3   4(VMS_DESC), R0, END_ADDRESS                     ;
                                        52   D4 0008C          CLRL    CURRENT_BLOCK                                    ; 0858
                            53          54   D1 0008E 8$:      CMPL    CURRENT_PAGE_ADDRESS, END_ADDRESS                ; 0859
                                        1A   14 00091          BGTR    9$                                               ;
                                        02   DD 00093          PUSHL   #2                                               ; 0867
                00000000G   00          01   FB 00095          CALLS   #1, DBG$GET_TEMPMEM                              ;
                            60          52   D0 0009C          MOVL    CURRENT_BLOCK, (NEXT_BLOCK)                      ; 0868
                    04      A0          84   7E 0009F          MOVAQ   (CURRENT_PAGE_ADDRESS)+, 4(NEXT_BLOCK)           ; 0869
                            52          50   D0 000A3          MOVL    NEXT_BLOCK, CURRENT_BLOCK                        ; 0870
                            54    01F8  C4   9E 000A6          MOVAB   504(R4), CURRENT_PAGE_ADDRESS                    ; 0871
                                        E1   11 000AB          BRB     8$                                               ; 0859
                    08      BC          52   D0 000AD 9$:      MOVL    CURRENT_BLOCK, @PARAM2                           ; 0876
                            50          01   D0 000B1          MOVL    #1, R0                                           ; 0877
                                             04 000B4          RET                                                      ; 0878

; Routine Size:  181 bytes,    Routine Base:  DBG$CODE + 0151
```

```
;  754     0879  1 GLOBAL ROUTINE DBG$NGET_LENGTH (PRIM_DESC, PARAM2, PARAM3) =
;  755     0880  1 !++
;  756     0881  1 ! FUNCTIONAL DESCRIPTION:
;  757     0882  1 !
;  758     0883  1 !       Uses a symbol's primary descriptor to obtain the length of the symbol's
;  759     0884  1 !       rvalue. The length is to be given in bits. Lengths longer than 2 ** 32
;  760     0885  1 !       must be truncated to this length.
;  761     0886  1 !
;  762     0887  1 !       The debugger assumes that rvalues refer to contiguous blocks of storage.
;  763     0888  1 !       If this is not true for a given variable, this routine fails.
;  764     0889  1 !
;  765     0890  1 !       Length should reflect the maximum length for entities that may vary in
;  766     0891  1 !       size, and include the length of a control word, if one is present.
;  767     0892  1 !
;  768     0893  1 !       If the value of the object can not be materialized by the Type Convertor
;  769     0894  1 !       (DBG$NTYPE_CONV), this routine should return STS$K_INFO. This is
;  770     0895  1 !       generally true for objects of aggregate type, e.g., PASCAL arrays and
;  771     0896  1 !       record, PL/I structures.
;  772     0897  1 !
;  773     0898  1 !       This routine calls a language-specific routine based on the language
;  774     0899  1 !       code in the descriptor header.
;  775     0900  1 !
;  776     0901  1 ! FORMAL PARAMETERS:
;  777     0902  1 !
;  778     0903  1 !       prim_desc       - A longword containing the address of a primary descriptor
;  779     0904  1 !
;  780     0905  1 !       param2          - The address of a longword to contain an unsigned integer
;  781     0906  1 !                         longword representing the symbol's rvalue length in bits
;  782     0907  1 !
;  783     0908  1 !       param3          - The address of a longword to contain the address of a
;  784     0909  1 !                         message argument vector as described on page 4-119 of
;  785     0910  1 !                         the VAX/VMS system reference, volume 1A
;  786     0911  1 !
;  787     0912  1 ! IMPLICIT INPUTS:
;  788     0913  1 !
;  789     0914  1 !       NONE
;  790     0915  1 !
;  791     0916  1 ! IMPLICIT OUTPUTS:
;  792     0917  1 !
;  793     0918  1 !       In case of a severe error return, a message argument vector is constructed
;  794     0919  1 !       from dynamic storage and returned.
;  795     0920  1 !
;  796     0921  1 ! ROUTINE VALUE:
;  797     0922  1 !
;  798     0923  1 !       An unsigned integer longword completion code
;  799     0924  1 !
;  800     0925  1 ! COMPLETION CODES:
;  801     0926  1 !
;  802     0927  1 !       STS$K_SUCCESS (1) - Success. Length of symbol's rvalue returned.
;  803     0928  1 !
;  804     0929  1 !       STS$K_INFO    (3) - Success. Length of the symbol's rvalue returned but
;  805     0930  1 !                           the symbol refers to a value that the Type Convertor
;  806     0931  1 !                           cannot materialize.
;  807     0932  1 !
;  808     0933  1 !       STS$K_SEVERE  (4) - Failure. No length returned. Message argument vector
;  809     0934  1 !                           constructed and returned.
;  810     0935  1 !
```

```
;  811          0936  1 ! SIDE EFFECTS:
;  812          0937  1 !
;  813          0938  1 !       NONE
;  814          0939  1 !
;  815          0940  1 !
;  816          0941  2     BEGIN
;  817          0942  2
;  818          0943  2     MAP
;  819          0944  2         PRIM_DESC: REF DBG$VALDESC;
;  820          0945  2     LOCAL
;  821          0946  2         VMS_DESC,
;  822          0947  2         VMS_DESC_AREA: DBG$STG_DESC;
;  823          0948  2
;  824          0949  2     ! Primary Descriptors.
;  825          0950  2     !
;  826          0951  2     IF .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC
;  827          0952  2     THEN
;  828          0953  3         BEGIN
;  829          0954  3
;  830          0955  3         ! Call a routine to construct the VMS descriptor.
;  831          0956  3         !
;  832          0957  3         VMS_DESC = VMS_DESC_AREA;
;  833          0958  3         IF NOT DBG$MAKE_VMS_DESC (.PRIM_DESC, .VMS_DESC)
;  834          0959  3         THEN
;  835          0960  3             $DBG_ERROR ('DBGLANVEC\DBG$NGET_LENGTH');
;  836          0961  3         END
;  837          0962  3
;  838          0963  3     ! Volatile Value Descriptors or Value Descriptors.
;  839          0964  3     !
;  840          0965  2     ELSE IF .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC
;  841          0966  2     OR .PRIM_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_VALUE_DESC
;  842          0967  2     THEN
;  843          0968  2
;  844          0969  2         ! In this case just get the VMS descriptor out of the
;  845          0970  2         ! volatile value descriptor.
;  846          0971  2         !
;  847          0972  2         VMS_DESC = PRIM_DESC [DBG$A_VALUE_VMSDESC]
;  848          0973  2
;  849          0974  2     ! We do not expect any other kind of descriptor.
;  850          0975  2     !
;  851          0976  2     ELSE
;  852          0977  2         $DBG_ERROR ('DBGLANVEC\DBG$NGET_LENGTH unknown descriptor type');
;  853          0978  2
;  854          0979  2
;  855          0980  2     ! Call the routine in DBGVALUES that extracts a bit length from
;  856          0981  2     ! a VMS descriptor.
;  857          0982  2     !
;  858          0983  2     .PARAM2 = DBG$DATA_LENGTH (.VMS_DESC);
;  859          0984  2     RETURN STS$K_SUCCESS;
;  860          0985  1     END;


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

24  47  42  44  5C  43  45  56  4E  41  4C  47  42  44  19  000B9 P.AAF:  .ASCII  <25>\DBGLANVEC\<92>\DBG$NGET_LENGTH\        ;
                48  54  47  4E  45  4C  5F  54  45  47  4E  000C8                                                            ;
```

```
24 47 42 44 5C 43 45 56 4E 41 4C 47 42 44 31  000D3 P.AAG:  .ASCII  \1DBGLANVEC\<92>\DBG$NGET_LENGTH unknown\      ;
6B 6E 75 20 48 54 47 4E 45 4C 5F 54 45 47 4E  000E2
                                 6E 77 6F 6E  000F1
70 79 74 20 72 6F 74 70 69 72 63 73 65 64 20  000F5          .ASCII  \ descriptor type\                            ;
                                          65  00104                                                              ;


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

                                             0004 00000           .ENTRY  DBG$NGET_LENGTH, Save R2                 ; 0879
                                          5E   0C   C2 00002      SUBL2   #12, SP                                  ;
00000079   8F     04   BC                08   10   ED 00005      CMPZV   #16, #8, @PRIM_DESC, #121                ; 0951
                                          1A   12 0000F          BNEQ    1$                                       ;
                                          52   6E   9E 00011     MOVAB   VMS_DESC_AREA, VMS_DESC                  ; 0957
                                               52   DD 00014     PUSHL   VMS_DESC                                 ; 0958
                                                    04   AC DD 00016     PUSHL   PRIM_DESC                        ;
                                 00000000G     00        02   FB 00019     CALLS   #2, DBG$MAKE_VMS_DESC            ;
                                               3C        50   E8 00020     BLBS    R0, 5$                           ;
                                    00000000'          EF   9F 00023     PUSHAB  P.AAF                            ; 0960
                                               25      11   00029     BRB     4$                                   ;
00000083   8F     04   BC                08   10   ED 0002B 1$:    CMPZV   #16, #8, @PRIM_DESC, #131                ; 0965
                                          0C   13 00035          BEQL    2$                                       ;
0000007A   8F     04   BC                08   10   ED 00037      CMPZV   #16, #8, @PRIM_DESC, #122                ; 0966
                                          07   12 00041          BNEQ    3$                                       ;
                52        04   AC                14   C1 00043 2$:     ADDL3   #20, PRIM_DESC, VMS_DESC         ; 0972
                                          15   11 00048          BRB     5$                                       ;
                                    00000000'          EF   9F 0004A 3$:     PUSHAB  P.AAG                        ; 0977
                                               01   DD 00050 4$:     PUSHL   #1                                   ;
                                    00028362   8F   DD 00052          PUSHL   #164706                              ;
                                 00000000G     00   03   FB 00058     CALLS   #3, LIB$SIGNAL                       ;
                                               52   DD 0005F 5$:     PUSHL   VMS_DESC                             ; 0983
                                 00000000G     00   01   FB 00061     CALLS   #1, DBG$DATA_LENGTH                  ;
                                       08      BC   50   D0 00068     MOVL    R0, @PARAM2                          ;
                                               50   01   D0 0006C     MOVL    #1, R0                               ; 0984
                                                         04 0006F     RET                                          ; 0985

; Routine Size:  112 bytes,    Routine Base:  DBG$CODE + 0206
```

```
;  862     0986  1 GLOBAL ROUTINE DBG$NCOPY_DESC (DESC, PARAM2, PARAM3, PARAM4) =
;  863     0987  1 !
;  864     0988  1 ! FUNCTIONAL DESCRIPTION:
;  865     0989  1 !
;  866     0990  1 !       Accepts as input a language specific primary or value descriptor
;  867     0991  1 !       (constructed from listed storage)
;  868     0992  1 !       and makes a copy of the descriptor out of no -listed storage. This
;  869     0993  1 !       non-volatile copy will be stored in conjunction with x-points and
;  870     0994  1 !       current location.
;  871     0995  1 !
;  872     0996  1 !       This routine may use DBG$NCOPY to copy each portion of the
;  873     0997  1 !       descriptor that has been created from listed dynamic storage.
;  874     0998  1 !
;  875     0999  1 ! FORMAL PARAMETERS:
;  876     1000  1 !
;  877     1001  1 !       desc                    - The address of a language specifc primary or
;  878     1002  1 !                                 value descriptor
;  879     1003  1 !
;  880     1004  1 !       param2                  - The address of a longword to contain the address
;  881     1005  1 !                                 of the non-volatile copy of the descriptor
;  882     1006  1 !
;  883     1007  1 !       param3                  - The address of a longword to contain the address
;  884     1008  1 !                                 of a message argument vector for errors
;  885     1009  1 !
;  886     1010  1 !       param4                  - A flag saying whether to copy into permanent
;  887     1011  1 !                                 memory or temporary memory. Only used in
;  888     1012  1 !                                 implementation level 3.
;  889     1013  1 ! IMPLICIT INPUTS:
;  890     1014  1 !
;  891     1015  1 !       NONE
;  892     1016  1 !
;  893     1017  1 ! IMPLICIT OUTPUTS:
;  894     1018  1 !
;  895     1019  1 !       On success, the non-volatile copy of a primary descriptor.
;  896     1020  1 !
;  897     1021  1 !       On failure, a message argument vector.
;  898     1022  1 !
;  899     1023  1 ! ROUTINE VALUE:
;  900     1024  1 !
;  901     1025  1 !       An unsigned integer longword completion code
;  902     1026  1 !
;  903     1027  1 ! COMPLETION CODES:
;  904     1028  1 !
;  905     1029  1 !       STS$K_SUCCESS   (1)     - Success. Copy constructed and returned.
;  906     1030  1 !
;  907     1031  1 !       STS$K_SEVERE    (4)     - Failure. Copy not produced. Message argument
;  908     1032  1 !                                 vector constructed and returned.
;  909     1033  1 !
;  910     1034  1 ! SIDE EFFECTS:
;  911     1035  1 !
;  912     1036  1 !       NONE
;  913     1037  1 !
;  914     1038  2     BEGIN
;  915     1039  2
;  916     1040  2     MAP
;  917     1041  2         DESC: REF DBG$VALDESC;
;  918     1042  2
```

```
;  919          1043  2     BUILTIN
;  920          1044  2         ACTUALCOUNT;                ! Count of acual parameters.
;  921          1045  2
;  922          1046  2     LOCAL
;  923          1047  2         LENGTH,                     ! Length in bytes of copy
;  924          1048  2         PERM_FLAG;                  ! Flag saying whether to copy into permanent
;  925          1049  2                                     !    or temporary memory.
;  926          1050  2
;  927          1051  2     ! Enable a handler which will take care of NOFREE error messages.
;  928          1052  2     ! The reason for this is, if we run out of memory part way through
;  929          1053  2     ! copying the descriptor, then we want to release the memory we
;  930          1054  2     ! have allocated so far, so that it does not get lost forever.
;  931          1055  2     !
;  932          1056  2     ENABLE
;  933          1057  2         COPY_DESC_HANDLER;
;  934          1058  2
;  935          1059  2     ! Default the fourth parameter to TRUE.
;  936          1060  2     ! Also initialize the pointer to the new descriptor header.
;  937          1061  2     !
;  938          1062  2     IF ACTUALCOUNT() LSS 4
;  939          1063  2     THEN
;  940          1064  2         PERM_FLAG = TRUE
;  941          1065  2     ELSE
;  942          1066  2         PERM_FLAG = .PARAM4;
;  943          1067  2     COPY_DESC_HEAD = 0;
;  944          1068  2
;  945          1069  2     ! Compute the number of bytes to allocate. Always allocate
;  946          1070  2     ! at least 16 + base size of value descriptor.
;  947          1071  2     !
;  948          1072  2     LENGTH = .DESC[DBG$W_DHDR_LENGTH];
;  949          1073  2     IF .LENGTH LSS 16 + 4*DBG$K_VALDESC_BASE_SIZE
;  950          1074  2     THEN
;  951          1075  2         LENGTH = 16 + 4*DBG$K_VALDESC_BASE_SIZE;
;  952          1076  2
;  953          1077  2     CASE .DESC [DBG$B_DHDR_TYPE] FROM DBG$K_LITERAL TO DBG$K_V_VALUE_DESC OF
;  954          1078  2         SET
;  955          1079  2
;  956          1080  2         ! Ordinary value descriptors. These have the actual value embedded
;  957          1081  2         ! inside them. Copy the descriptor and fix up the pointer field
;  958          1082  2         ! so it points to the right place.
;  959          1083  2         !
;  960          1084  2         [DBG$K_VALUE_DESC]:
;  961          1085  3             BEGIN
;  962          1086  3             MAP
;  963          1087  3                 DESC: REF DBG$VALDESC;  ! Pointer to a new style value
;  964          1088  3                                         !     descriptor (the original)
;  965          1089  3             LOCAL
;  966          1090  3                 DESC_COPY: REF DBG$VALDESC;     ! Pointer to a new style value
;  967          1091  3                                                 !     descriptor (the copy).
;  968          1092  3
;  969          1093  3             IF .PERM_FLAG
;  970          1094  3             THEN
;  971          1095  3                 DESC_COPY = DBG$GET_MEMORY ((3+.LENGTH)/4)
;  972          1096  3             ELSE
;  973          1097  3                 DESC_COPY = DBG$GET_TEMPMEM ((3+.LENGTH)/4);
;  974          1098  3             CH$MOVE (.DESC[DBG$W_DHDR_LENGTH], .DESC, .DESC_COPY);
;  975          1099  3             DESC_COPY [DBG$L_VALUE_POINTER] = DESC_COPY [DBG$A_VALUE_ADDRESS];
```

```
;  976          1100  3            .PARAM2 = .DESC_COPY;
;  977          1101  2            END;
;  978          1102  2
;  979          1103  2        ! Volatile value descriptors. These point to a region of user
;  980          1104  2        ! memory containing the value. We do the same as above except
;  981          1105  2        ! that we do not fix up the pointer field.
;  982          1106  2        !
;  983          1107  2        [DBG$K_V_VALUE_DESC]:
;  984          1108  3            BEGIN
;  985          1109  3            MAP
;  986          1110  3                DESC: REF DBG$VALDESC;  ! Pointer to a new style value
;  987          1111  3                                       !     descriptor (the original)
;  988          1112  3            LOCAL
;  989          1113  3                DESC_COPY: REF DBG$VALDESC;     ! Pointer to a new style value
;  990          1114  3                                                !     descriptor (the copy).
;  991          1115  3
;  992          1116  3            IF .PERM_FLAG
;  993          1117  3            THEN
;  994          1118  3                DESC_COPY = DBG$GET_MEMORY ((3+.LENGTH)/4)
;  995          1119  3            ELSE
;  996          1120  3                DESC_COPY = DBG$GET_TEMPMEM ((3+.LENGTH)/4);
;  997          1121  3            CH$MOVE (7,.DESC[DBG$W_DHDR_LENGTH], .DESC, .DESC_COPY);
;  998          1122  3            .PARAM2 = .DESC_COPY;
;  999          1123  3            END;
; 1000          1124  2
; 1001          1125  2        ! New style Primary Descriptors. Here we have to copy the root
; 1002          1126  2        ! node and all sub-nodes. Note that we have to do this carefully,
; 1003          1127  2        ! in such as way that at any time we call GET_MEMORY, we must
; 1004          1128  2        ! have a valid (though partially constructed) Primary. This is
; 1005          1129  2        ! in case GET_MEMORY signals a NOFREE error message - we want
; 1006          1130  2        ! to be able to release the storage we have allocated up
; 1007          1131  2        ! to the point of running out of memory.
; 1008          1132  2        !
; 1009          1133  2        [DBG$K_PRIMARY_DESC]:
; 1010          1134  3            BEGIN
; 1011          1135  3
; 1012          1136  3            MAP
; 1013          1137  3                DESC: REF DBG$PRIMARY;                  ! Pointer to the Primary
; 1014          1138  3                                                        !     Descriptor to
; 1015          1139  3                                                        !     be copied.
; 1016          1140  3            LOCAL
; 1017          1141  3                DESC_COPY : REF DBG$PRIMARY,            ! Pointer to the copy
; 1018          1142  3                                                        !     of the Primary
; 1019          1143  3                                                        !     Descriptor.
; 1020          1144  3                DIMCNT,
; 1021          1145  3                NEW_SUBNODE: REF DBG$PRIM_NODE,         ! Pointer to a copy of
; 1022          1146  3                                                        !    a subnode
; 1023          1147  3                NUMBLKS,
; 1024          1148  3                PREV_SUBNODE: REF DBG$PRIM_NODE,        ! Pointer to a copy of
; 1025          1149  3                                                        !    a subnode
; 1026          1150  3                SIZE,                                   ! Size of a subnode
; 1027          1151  3                SUBCNT,
; 1028          1152  3                SUBNODE: REF DBG$PRIM_NODE;             ! Pointer to the original
; 1029          1153  3                                                        !    subnode.
; 1030          1154  3
; 1031          1155  3            ! Allocate memory for a new root node and copy the
; 1032          1156  3            ! values into it. We will fix up forward and back
```

```
;  1033          1157  3          ! Links later.
;  1034          1158  3          !
;  1035          1159  3          IF .PERM_FLAG
;  1036          1160  3          THEN
;  1037          1161  4              BEGIN
;  1038          1162  4              DESC_COPY = DBG$GET_MEMORY (DBG$K_PRIMARY_SIZE);
;  1039          1163  4
;  1040          1164  4              ! Put a pointer to the Primary in this own variable so
;  1041          1165  4              ! COPY_DESC_HANDLER can later free up the storage.
;  1042          1166  4              !
;  1043          1167  4              COPY_DESC_HEAD = .DESC_COPY;
;  1044          1168  4              END
;  1045          1169  3          ELSE
;  1046          1170  3              DESC_COPY = DBG$GET_TEMPMEM (DBG$K_PRIMARY_SIZE);
;  1047          1171  3          CH$MOVE (4*DBG$K_PRIMARY_SIZE, .DESC, .DESC_COPY);
;  1048          1172  3
;  1049          1173  3          ! Fix up the forward and back links so we have a valid partially
;  1050          1174  3          ! constructed Primary - i.e., we do not want to leave them pointing
;  1051          1175  3          ! to the original Primary.
;  1052          1176  3          !
;  1053          1177  3          DESC_COPY[DBG$L_PRIM_FLINK] = DESC_COPY[DBG$L_PRIM_FLINK];
;  1054          1178  3          DESC_COPY[DBG$L_PRIM_BLINK] = DESC_COPY[DBG$L_PRIM_FLINK];
;  1055          1179  3
;  1056          1180  3          ! Loop through each of the subnodes.
;  1057          1181  3          !
;  1058          1182  3          SUBNODE = .DESC [DBG$L_PRIM_FLINK];
;  1059          1183  3          PREV_SUBNODE = 0;
;  1060          1184  3          WHILE .SUBNODE NEQ DESC[DBG$L_PRIM_FLINK] DO
;  1061          1185  4              BEGIN
;  1062          1186  4
;  1063          1187  4              ! Allocate space for the new subnode.
;  1064          1188  4              !
;  1065          1189  4              IF .SUBNODE [DBG$B_PNODE_FCODE] EQL RST$K_TYPE_ARRAY
;  1066          1190  4              THEN
;  1067          1191  5                  BEGIN
;  1068          1192  5                  ! Use larger of SUBCNT, DIMCNT.
;  1069          1193  5                  SUBCNT = .SUBNODE[DBG$B_PNARR_SUBCNT];
;  1070          1194  5                  DIMCNT = .SUBNODE[DBG$B_PNARR_DIMCNT];
;  1071          1195  5                  IF .SUBCNT GTR .DIMCNT
;  1072          1196  5                  THEN
;  1073          1197  5                      NUMBLKS = .SUBCNT
;  1074          1198  5                  ELSE
;  1075          1199  5                      NUMBLKS = .DIMCNT;
;  1076          1200  5                  SIZE = DBG$K_PRIM_SIZE_ARRAY +
;  1077          1201  5                         DBG$K_PRIM_SIZE_SUBS*.NUMBLKS;
;  1078          1202  5                  END
;  1079          1203  4              ELSE IF .SUBNODE [DBG$B_PNODE_FCODE] EQL RST$K_TYPE_RECORD
;  1080          1204  4              THEN
;  1081          1205  4                  SIZE = DBG$K_PRIM_SIZE_RECORD
;  1082          1206  4              ELSE IF .SUBNODE [DBG$B_PNODE_FCODE] EQL RST$K_TYPE_VARIANT
;  1083          1207  4              THEN
;  1084          1208  4                  SIZE = DBG$K_PRIM_SIZE_VARIANT
;  1085          1209  4              ELSE
;  1086          1210  4                  SIZE = DBG$K_PRIM_SIZE_NORMAL;
;  1087          1211  4              IF .PERM_FLAG
;  1088          1212  4              THEN
;  1089          1213  4                  NEW_SUBNODE = DBG$GET_MEMORY(.SIZE)
```

```
;  1090          1214  4          ELSE
;  1091          1215  4              NEW_SUBNODE = DBG$GET_TEMPMEM(.SIZE);
;  1092          1216  4
;  1093          1217  4          ! Copy the values.
;  1094          1218  4          !
;  1095          1219  4          CH$MOVE (4*.SIZE, .SUBNODE, .NEW_SUBNODE);
;  1096          1220  4
;  1097          1221  5          IF .PERM_FLAG AND (.SUBNODE [DBG$B_PNODE_FCODE] EQL RST$K_TYPE_VARIANT)
;  1098          1222  4          THEN
;  1099          1223  4              NEW_SUBNODE[DBG$V_PNVAR_VALID] = FALSE;
;  1100          1224  4
;  1101          1225  4          ! Fill in the forward and back links.
;  1102          1226  4          !
;  1103          1227  4          IF .PREV_SUBNODE EQL 0
;  1104          1228  4          THEN
;  1105          1229  5              BEGIN
;  1106          1230  5              DESC_COPY [DBG$L_PRIM_FLINK] = .NEW_SUBNODE;
;  1107          1231  5              DESC_COPY [DBG$L_PRIM_BLINK] = .NEW_SUBNODE;
;  1108          1232  5              NEW_SUBNODE [DBG$L_PNODE_FLINK] = DESC_COPY [DBG$A_PRIM_FLINK];
;  1109          1233  5              NEW_SUBNODE [DBG$L_PNODE_BLINK] = DESC_COPY [DBG$A_PRIM_FLINK];
;  1110          1234  5              END
;  1111          1235  4          ELSE
;  1112          1236  5              BEGIN
;  1113          1237  5              PREV_SUBNODE [DBG$L_PNODE_FLINK] = .NEW_SUBNODE;
;  1114          1238  5              DESC_COPY [DBG$L_PRIM_BLINK] = .NEW_SUBNODE;
;  1115          1239  5              NEW_SUBNODE [DBG$L_PNODE_FLINK] = DESC_COPY [DBG$A_PRIM_FLINK];
;  1116          1240  5              NEW_SUBNODE [DBG$L_PNODE_BLINK] = .PREV_SUBNODE;
;  1117          1241  4              END;
;  1118          1242  4          PREV_SUBNODE = .NEW_SUBNODE;
;  1119          1243  4          SUBNODE = .SUBNODE [DBG$L_PNODE_FLINK];
;  1120          1244  3          END;
;  1121          1245  3
;  1122          1246  3              .PARAM2 = .DESC_COPY;
;  1123          1247  2               END;
;  1124          1248  2
;  1125          1249  2          ! At implementation level 3, we do not expect any other kind
;  1126          1250  2          ! of descriptor.
;  1127          1251  2          !
;  1128          1252  2          [INRANGE, OUTRANGE]:
;  1129          1253  2              $DBG_ERROR ('DBGLANVEC\DBG$NCOPY_DESC');
;  1130          1254  2
;  1131          1255  2          TES;
;  1132          1256  2
;  1133          1257  2      ! The copying has been done. Return success.
;  1134          1258  2      !
;  1135          1259  2      RETURN STS$K_SUCCESS;
;  1136          1260  1      END;


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

24  47  42  44  5C  43  45  56  4E  41  4C  47  42  44  18  00105 P.AAH:  .ASCII  <24>\DBGLANVEC\<92>\DBG$NCOPY_DESC\          ;
                    43  53  45  44  5F  59  50  4F  43  4E  00114                                                              ;
```

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

                                                OFFC 00000      .ENTRY  DBG$NCOPY_DESC, Save R2,R3,R4,R5,R6,R7,R8,- ; 0986
                                                                        R9,R10,R11
                                          5E        14  C2 00002      SUBL2   #20, SP
                                          6D  01A7  CF  DE 00005      MOVAL   32$, (FP)                           ; 1038
                                          04        6C  91 0000A      CMPB    (AP), #4                            ; 1062
                                                    05  1E 0000D      BGEQU   1$
                                          6E        01  D0 0000F      MOVL    #1, PERM_FLAG                       ; 1064
                                                    04  11 00012      BRB     2$
                                          6E    10  AC  D0 00014 1$:  MOVL    PARAM4, PERM_FLAG                   ; 1066
                                        00000000' EF  D4 00018 2$:  CLRL    COPY_DESC_HEAD                      ; 1067
                                          5B    04  AC  D0 0001E      MOVL    DESC, R11                           ; 1072
                                          50        6B  3C 00022      MOVZWL  (R11), LENGTH
                                          30        50  D1 00025      CMPL    LENGTH, #48                         ; 1073
                                                    03  18 00028      BGEQ    3$
                                          50        30  D0 0002A      MOVL    #48, LENGTH                         ; 1075
                                    0B  78  8F      02  AB  8F 0002D 3$:  CASEB   2(R11), #120, #11              ; 1077
                              0018  002F    0085      0018    00033 4$:  .WORD   5$-4$,-
                              0018  0018    0018      0018    0003B      14$-4$,-
                              005A  0018    0018      0018    00043      6$-4$,-
                                                                         5$-4$,-
                                                                         5$-4$,-
                                                                         5$-4$,-
                                                                         5$-4$,-
                                                                         5$-4$,-
                                                                         5$-4$,-
                                                                         5$-4$,-
                                                                         5$-4$,-
                                                                         9$-4$
                                        00000000' EF  9F 0004B 5$:  PUSHAB  P.AAH                               ; 1253
                                                  01  DD 00051      PUSHL   #1
                                        00028362  8F  DD 00053      PUSHL   #164706
                              00000000G 00        03  FB 00059      CALLS   #3, LIB$SIGNAL
                                                  53  11 00060      BRB     13$
                                          50        03  C0 00062 6$:  ADDL2   #3, R0                           ; 1095
                                          50        04  C6 00065      DIVL2   #4, R0
                                          0B        6E  E9 00068      BLBC    PERM_FLAG, 7$                       ; 1093
                                                    50  DD 0006B      PUSHL   R0                                  ; 1095
                              00000000G 00        01  FB 0006D      CALLS   #1, DBG$GET_MEMORY
                                                  09  11 00074      BRB     8$
                                                    50  DD 00076 7$:  PUSHL   R0                                  ; 1097
                              00000000G 00        01  FB 00078      CALLS   #1, DBG$GET_TEMPMEM
                                          56        50  D0 0007F 8$:  MOVL    R0, DESC_COPY
                                66        6B        6B  28 00082      MOVC3   (R11), (R11), (DESC_COPY)           ; 1098
                                    18    A6    20  A6  9E 00086      MOVAB   32(R6), 24(DESC_COPY)               ; 1099
                                                  24  11 0008B      BRB     12$                                 ; 1100
                                          50        03  C0 0008D 9$:  ADDL2   #3, R0                           ; 1118
                                          50        04  C6 00090      DIVL2   #4, R0
                                          0B        6E  E9 00093      BLBC    PERM_FLAG, 10$                      ; 1116
                                                    50  DD 00096      PUSHL   R0                                  ; 1118
                              00000000G 00        01  FB 00098      CALLS   #1, DBG$GET_MEMORY
                                                  09  11 0009F      BRB     11$
                                                    50  DD 000A1 10$:  PUSHL   R0                                  ; 1120
                              00000000G 00        01  FB 000A3      CALLS   #1, DBG$GET_TEMPMEM
                                          56        50  D0 000AA 11$:  MOVL    R0, DESC_COPY
                                66        6B        6B  28 000AD      MOVC3   (R11), (R11), (DESC_COPY)           ; 1121
```

```
                08   BC            56  D0 000B1 12$:   MOVL    DESC_COPY, @PARAM2                      ; 1122
                                 00F4  31 000B5 13$:   BRW     31$                                     ; 1077
                     15              6E  E9 000B8 14$:   BLBC    PERM_FLAG, 15$                          ; 1159
                                     09  DD 000BB        PUSHL   #9                                      ; 1162
          00000000G  00              01  FB 000BD        CALLS   #1, DBG$GET_MEMORY                      ;
                     5A              50  D0 000C4        MOVL    R0, DESC_COPY                           ;
          00000000'  EF              5A  D0 000C7        MOVL    DESC_COPY, COPY_DESC_HEAD               ; 1167
                                      0C  11 000CE        BRB     16$                                     ; 1159
                                     09  DD 000D0 15$:   PUSHL   #9                                      ; 1170
          00000000G  00              01  FB 000D2        CALLS   #1, DBG$GET_TEMPMEM                     ;
                     5A              50  D0 000D9        MOVL    R0, DESC_COPY                           ;
     6A              6B              24  28 000DC 16$:   MOVC3   #36, (R11), (DESC_COPY)                ; 1171
                08   AE        14    AA  9E 000E0        MOVAB   20(DESC_COPY), 8(SP)                    ; 1177
                08   BE        08    AE  D0 000E5        MOVL    8(SP), @8(SP)                           ;
                18   AA        08    AE  D0 000EA        MOVL    8(SP), 24(DESC_COPY)                    ; 1178
                     59        14    AB  D0 000EF        MOVL    20(R11), SUBNODE                        ; 1182
                               04    AE  D4 000F3        CLRL    PREV_SUBNODE                            ; 1183
                     50        14    AB  9E 000F6 17$:   MOVAB   20(R11), R0                             ; 1184
                     50              59  D1 000FA        CMPL    SUBNODE, R0                             ;
                                     03  12 000FD        BNEQ    18$                                     ;
                                   00A6  31 000FF        BRW     30$                                     ;
                     01        09    A9  91 00102 18$:   CMPB    9(SUBNODE), #1                          ; 1189
                                     25  12 00106        BNEQ    21$                                     ;
                0C   AE        1F    A9  9A 00108        MOVZBL  31(SUBNODE), SUBCNT                     ; 1193
                10   AE        1B    A9  9A 0010D        MOVZBL  27(SUBNODE), DIMCNT                     ; 1194
                10   AE        0C    AE  D1 00112        CMPL    SUBCNT, DIMCNT                          ; 1195
                                     06  15 00117        BLEQ    19$                                     ;
                     57        0C    AE  D0 00119        MOVL    SUBCNT, NUMBLKS                         ; 1197
                                     04  11 0011D        BRB     20$                                     ;
                     57        10    AE  D0 0011F 19$:   MOVL    DIMCNT, NUMBLKS                         ; 1199
     50              57              05  C5 00123 20$:   MULL3   #5, NUMBLKS, R0                         ; 1201
                     56        0A    A0  9E 00127        MOVAB   10(R0), SIZE                            ; 1200
                                     19  11 0012B        BRB     24$                                     ; 1189
                     07        09    A9  91 0012D 21$:   CMPB    9(SUBNODE), #7                          ; 1203
                                     05  12 00131        BNEQ    22$                                     ;
                     56              07  D0 00133        MOVL    #7, SIZE                                ; 1205
                                     0E  11 00136        BRB     24$                                     ;
                     13        09    A9  91 00138 22$:   CMPB    9(SUBNODE), #19                         ; 1206
                                     05  12 0013C        BNEQ    23$                                     ;
                     56              0A  D0 0013E        MOVL    #10, SIZE                               ; 1208
                                     03  11 00141        BRB     24$                                     ;
                     56              06  D0 00143 23$:   MOVL    #6, SIZE                                ; 1210
                     0B              6E  E9 00146 24$:   BLBC    PERM_FLAG, 25$                          ; 1213
                                     56  DD 00149        PUSHL   SIZE                                    ;
          00000000G  00              01  FB 0014B        CALLS   #1, DBG$GET_MEMORY                      ;
                                     09  11 00152        BRB     26$                                     ;
                                     56  DD 00154 25$:   PUSHL   SIZE                                    ; 1215
          00000000G  00              01  FB 00156        CALLS   #1, DBG$GET_TEMPMEM                     ;
                     58              50  D0 0015D 26$:   MOVL    R0, NEW_SUBNODE                         ;
     50              56              02  78 00160        ASHL    #2, SIZE, R0                            ; 1219
     68              69              50  28 00164        MOVC3   R0, (SUBNODE), (NEW_SUBNODE)            ;
                     0A              6E  E9 00168        BLBC    PERM_FLAG, 27$                          ; 1221
                     13        09    A9  91 0016B        CMPB    9(SUBNODE), #19                         ;
                                     04  12 0016F        BNEQ    27$                                     ;
                0A   A8              10  8A 00171        BICB2   #16, 10(NEW_SUBNODE)                    ; 1223
                               04    AE  D5 00175 27$:   TSTL    PREV_SUBNODE                            ; 1227
                                     13  12 00178        BNEQ    28$                                     ;
```

```
                08   BE            58   D0 0017A          MOVL    NEW_SUBNODE, @8(SP)                    ; 1230
                18   AA            58   D0 0017E          MOVL    @NEW_SUBNODE, 24(DESC_COPY)            ; 1231
                     68      08    AE   D0 00182          MOVL    @8(SP), (NEW_SUBNODE)                  ; 1232
                04   A8      08    AE   D0 00186          MOVL    8(SP), 4(NEW_SUBNODE)                  ; 1233
                                   11   11 0018B          BRB     29$                                    ; 1227
                04   BE            58   D0 0018D 28$:     MOVL    NEW_SUBNODE, @PREV_SUBNODE             ; 1237
                18   AA            58   D0 00191          MOVL    NEW_SUBNODE, 24(DESC_COPY)             ; 1238
                     68      08    AE   D0 00195          MOVL    8(SP), (NEW_SUBNODE)                   ; 1239
                04   A8      04    AE   D0 00199          MOVL    PREV_SUBNODE, 4(NEW_SUBNODE)           ; 1240
                04   AE            58   D0 0019E 29$:     MOVL    NEW_SUBNODE, PREV_SUBNODE              ; 1242
                     59            69   D0 001A2          MOVL    (SUBNODE), SUBNODE                     ; 1243
                                FF4E   31 001A5          BRW     17$                                    ; 1184
                08   BC            5A   D0 001A8 30$:     MOVL    DESC_COPY, @PARAM2                     ; 1246
                     50            01   D0 001AC 31$:     MOVL    #1, R0                                 ; 1259
                                      04 001AF          RET                                            ; 1260
                                    0000 001B0 32$:     .WORD   Save nothing                           ; 1038
                                   7E   D4 001B2          CLRL    -(SP)                                  ;
                                   5E   DD 001B4          PUSHL   SP                                     ;
                     7E      04    AC   7D 001B6          MOVQ    4(AP), -(SP)                           ;
              0000V  CF            03   FB 001BA          CALLS   #3, COPY_DESC_HANDLER                  ;
                                      04 001BF          RET                                            ;

; Routine Size:  448 bytes,    Routine Base:  DBG$CODE + 0276
```

```
;  1138          1261  1 ROUTINE COPY_DESC_HANDLER (SIG, MECH) =
;  1139          1262  1 !
;  1140          1263  1 ! FUNCTION
;  1141          1264  1 !       This is the error hander for DBG$NCOPY_DESC. This routine is
;  1142          1265  1 !       responsible for freeing up the memory we have allocated so
;  1143          1266  1 !       far, if we get a NOFREE error message while copying the descriptor.
;  1144          1267  1 !
;  1145          1268  1 ! INPUTS
;  1146          1269  1 !       SIG     - Signal argument vector
;  1147          1270  1 !       MECH    - not used
;  1148          1271  1 !
;  1149          1272  1 ! IMPLICIT INPUT
;  1150          1273  1 !       COPY_DESC_HEAD - An own variable that points to the head of
;  1151          1274  1 !                        the descriptor copy.
;  1152          1275  1 !
;  1153          1276  1 ! OUTPUTS
;  1154          1277  1 !       This routine resignals the error.
;  1155          1278  1 !
;  1156          1279  2     BEGIN
;  1157          1280  2     MAP
;  1158          1281  2         SIG: REF VECTOR;
;  1159          1282  2
;  1160          1283  2     ! Only do something if the error is "no free storage" and if the own
;  1161          1284  2     ! variable COPY_DESC_HEAD is not zero (meaning that some storage has
;  1162          1285  2     ! been allocated before the NOFREE).
;  1163          1286  2     !
;  1164          1287  2     IF .SIG[1] EQL DBG$_NOFREE
;  1165          1288  2     THEN
;  1166          1289  2         IF .COPY_DESC_HEAD NEQ 0
;  1167          1290  2         THEN
;  1168          1291  3             BEGIN
;  1169          1292  3             DBG$NFREE_DESC(.COPY_DESC_HEAD);
;  1170          1293  3             COPY_DESC_HEAD = 0;
;  1171          1294  2             END;
;  1172          1295  2
;  1173          1296  2     ! Having freed the storage, resignal the error.
;  1174          1297  2     !
;  1175          1298  2     RETURN SS$_RESIGNAL;
;  1176          1299  1     END;


                                0004 00000 COPY_DESC_HANDLER:
                                           .WORD   Save R2                                  ; 1261
                52 00000000'    EF  9E 00002           MOVAB   COPY_DESC_HEAD, R2
                50        04    AC  D0 00009           MOVL    SIG, R0                      ; 1287
00028332        8F        04    A0  D1 0000D           CMPL    4(R0), #164658
                                0E  12 00015           BNEQ    1$
                50              62  D0 00017           MOVL    COPY_DESC_HEAD, R0           ; 1289
                                09  13 0001A           BEQL    1$
                                50  DD 0001C           PUSHL   R0                           ; 1292
            0000V   CF          01  FB 0001E           CALLS   #1, DBG$NFREE_DESC
                                62  D4 00023           CLRL    COPY_DESC_HEAD               ; 1293
                50      0918    8F  3C 00025 1$:       MOVZWL  #2328, R0                    ; 1298
                                    04 0002A           RET                                  ; 1299
```

; Routine Size:  43 bytes,    Routine Base:  DBG$CODE + 0436

```
;  1178          1300  1 GLOBAL ROUTINE DBG$NFREE_DESC (DESC, PARAM2, PARAM3) =
;  1179          1301  1 !
;  1180          1302  1 !++
;  1181          1303  1 ! FUNCTIONAL DESCRIPTION:
;  1182          1304  1 !
;  1183          1305  1 !       Releases dynamic storage associated with a non-volatile copy of a
;  1184          1306  1 !       language specific value or primary descriptor.
;  1185          1307  1 !       This routine accepts as input a copy of a primary or value
;  1186          1308  1 !       descriptor produced by DBG$NCOPY_DESC and calls the
;  1187          1309  1 !       routine DBG$REL_MEMORY to release each block of non-listed dynamic
;  1188          1310  1 !       storage contained within the descriptor.
;  1189          1311  1 !
;  1190          1312  1 !       This routine calls a language-specific routine based on the
;  1191          1313  1 !       language code in the descriptor header.
;  1192          1314  1 !
;  1193          1315  1 ! FORMAL PARAMETERS:
;  1194          1316  1 !
;  1195          1317  1 !       desc                    - The address of a non-volatile primary or
;  1196          1318  1 !                                 value descriptor
;  1197          1319  1 !
;  1198          1320  1 !       param2                  - The address of a longword to contain the address
;  1199          1321  1 !                                 of a message argument vector for errors
;  1200          1322  1 !
;  1201          1323  1 ! IMPLICIT INPUTS:
;  1202          1324  1 !
;  1203          1325  1 !       NONE
;  1204          1326  1 !
;  1205          1327  1 ! IMPLICIT OUTPUTS:
;  1206          1328  1 !
;  1207          1329  1 !       On failure, a message argument vector.
;  1208          1330  1 !
;  1209          1331  1 ! ROUTINE VALUE:
;  1210          1332  1 !
;  1211          1333  1 !       An unsigned integer longword completion code
;  1212          1334  1 !
;  1213          1335  1 ! COMPLETION CODES:
;  1214          1336  1 !
;  1215          1337  1 !       STS$K_SUCCESS   (1)     - Success. Storage for  descriptor released.
;  1216          1338  1 !
;  1217          1339  1 !       STS$K_SEVERE    (4)     - Failure. Storage for descriptor not released. Message
;  1218          1340  1 !                                 argument vector constructed and returned.
;  1219          1341  1 !
;  1220          1342  1 ! SIDE EFFECTS:
;  1221          1343  1 !
;  1222          1344  1 !       Dynamic memory is returned to the free storage pool.
;  1223          1345  1 !
;  1224          1346  1 !--
;  1225          1347  2     BEGIN
;  1226          1348  2     MAP
;  1227          1349  2         DESC: REF DBG$VALDESC;
;  1228          1350  2
;  1229          1351  2     ! Handle value descriptors separately from primary descriptors.
;  1230          1352  2     !
;  1231          1353  2     SELECTONE .DESC [DBG$B_DHDR_TYPE] OF
;  1232          1354  2         SET
;  1233          1355  2
;  1234          1356  2         ! Ordinary value descriptors. These are allocated in one contiguous
```

```
: 1235          1357  2          ! block so we can just release that block.
: 1236          1358  2          !
: 1237          1359  2          [DBG$K_VALUE_DESC, DBG$K_V_VALUE_DESC]:
: 1238          1360  3              BEGIN
: 1239          1361  3              DBG$REL_MEMORY (.DESC);
: 1240          1362  2              END;
: 1241          1363  2
: 1242          1364  2          ! New style Primary Descriptors. Here we have to release storage
: 1243          1365  2          ! for the root node and all the subnodes.
: 1244          1366  2          !
: 1245          1367  2          [DBG$K_PRIMARY_DESC]:
: 1246          1368  3              BEGIN
: 1247          1369  3              MAP
: 1248          1370  3                  DESC: REF DBG$PRIMARY;                  ! Pointer to the Primary
: 1249          1371  3                                                          !     Descriptor for which
: 1250          1372  3                                                          !     a symid list is to
: 1251          1373  3                                                          !     be constructed.
: 1252          1374  3              LOCAL
: 1253          1375  3                  NEW_SUBNODE,                            ! Pointer to the next
: 1254          1376  3                                                          !    subnode
: 1255          1377  3                  SAVED_PTR,                              ! Position in root node
: 1256          1378  3                                                          !    that is pointed to
: 1257          1379  3                                                          !    by the flink in the
: 1258          1380  3                                                          !    last subnode.
: 1259          1381  3                  SUBNODE: REF DBG$PRIM_NODE;             ! Pointer to a subnode.
: 1260          1382  3
: 1261          1383  3                  ! First save away a pointer to the subnode and a pointer
: 1262          1384  3                  ! which will identify when we have looped through all the
: 1263          1385  3                  ! subnodes. Then release the storage associated with the
: 1264          1386  3                  ! root node.
: 1265          1387  3                  !
: 1266          1388  3                  SAVED_PTR = DESC [DBG$L_PRIM_FLINK];
: 1267          1389  3                  SUBNODE = .DESC [DBG$L_PRIM_FLINK];
: 1268          1390  3                  DBG$REL_MEMORY (.DESC);
: 1269          1391  3
: 1270          1392  3                  ! Loop through the subnodes. After saving a pointer to the
: 1271          1393  3                  ! next subnode, release the storage for the current subnode.
: 1272          1394  3                  !
: 1273          1395  3                  WHILE .SUBNODE NEQ .SAVED_PTR DO
: 1274          1396  4                      BEGIN
: 1275          1397  4                      NEW_SUBNODE = .SUBNODE [DBG$L_PNODE_FLINK];
: 1276          1398  4                      DBG$REL_MEMORY (.SUBNODE);
: 1277          1399  4                      SUBNODE = .NEW_SUBNODE;
: 1278          1400  3                      END;
: 1279          1401  2                  END;
: 1280          1402  2
: 1281          1403  2              ! At implementation level 3, we do not expect any other kind
: 1282          1404  2              ! of descriptor.
: 1283          1405  2              !
: 1284          1406  2              [OTHERWISE]:
: 1285          1407  2                  $DBG_ERROR ('DBGLANVEC\DBG$NFREE_DESC');
: 1286          1408  2
: 1287          1409  2              TES;
: 1288          1410  2
: 1289          1411  2          ! The storage has been freed. Return success.
: 1290          1412  2          !
: 1291          1413  2          RETURN STS$K_SUCCESS;
```

```
; 1292          1414  1     END;


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

24 47 42 44 5C 43 45 56 4E 41 4C 47 42 44 18  0011E P.AAI:  .ASCII  <24>\DBGLANVEC\<92>\DBG$NFREE_DESC\          ;
                  43 53 45 44 5F 45 45 52 46 4E  0012D                                                        ;


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

                                          003C 00000            .ENTRY  DBG$NFREE_DESC, Save R2,R3,R4,R5      ; 1300
                     55 00000000G      00  9E 00002            MOVAB   DBG$REL_MEMORY, R5                    ;
                     52          04    AC  D0 00009            MOVL    DESC, R2                              ; 1353
                     50          02    A2  9A 0000D            MOVZBL  2(R2), R0                             ;
                  7A 8F                50  91 00011            CMPB    R0, #122                              ; 1359
                                       06  13 00015            BEQL    1$                                    ;
                  83 8F                50  91 00017            CMPB    R0, #131                              ;
                                       07  12 0001B            BNEQ    2$                                    ;
                                       52  DD 0001D 1$:        PUSHL   R2                                    ; 1361
                     65                01  FB 0001F            CALLS   #1, DBG$REL_MEMORY                    ;
                                       3A  11 00022            BRB     5$                                    ; 1353
                  79 8F                50  91 00024 2$:        CMPB    R0, #121                              ; 1367
                                       1F  12 00028            BNEQ    4$                                    ;
                     54          14    A2  9E 0002A            MOVAB   20(R2), SAVED_PTR                     ; 1388
                     53          14    A2  D0 0002E            MOVL    20(R2), SUBNODE                       ; 1389
                                       52  DD 00032            PUSHL   R2                                    ; 1390
                     65                01  FB 00034            CALLS   #1, DBG$REL_MEMORY                    ;
                     54                53  D1 00037 3$:        CMPL    SUBNODE, SAVED_PTR                    ; 1395
                                       22  13 0003A            BEQL    5$                                    ;
                     52                63  D0 0003C            MOVL    (SUBNODE), NEW_SUBNODE                ; 1397
                                       53  DD 0003F            PUSHL   SUBNODE                               ; 1398
                     65                01  FB 00041            CALLS   #1, DBG$REL_MEMORY                    ;
                     53                52  D0 00044            MOVL    NEW_SUBNODE, SUBNODE                  ; 1399
                                       EE  11 00047            BRB     3$                                    ; 1395
                        00000000'      EF  9F 00049 4$:        PUSHAB  P.AAI                                 ; 1407
                                       01  DD 0004F            PUSHL   #1                                    ;
                        00028362       8F  DD 00051            PUSHL   #164706                               ;
                00000000G  00          03  FB 00057            CALLS   #3, LIB$SIGNAL                        ;
                     50                01  D0 0005E 5$:        MOVL    #1, R0                                ; 1413
                                           04 00061            RET                                           ; 1414

; Routine Size:  98 bytes,    Routine Base:  DBG$CODE + 0461
```

```
;  1294          1415  1 GLOBAL ROUTINE DBG$NGET_SYMID (DESC, PARAM2, PARAM3) =
;  1295          1416  1 !
;  1296          1417  1 !++
;  1297          1418  1 !
;  1298          1419  1 !       Returns a list of symids contained within a language specific primary
;  1299          1420  1 !       or value descriptor.
;  1300          1421  1 !
;  1301          1422  1 !       This routine calls a language-specific routine based on the language
;  1302          1423  1 !       code in the descriptor header.
;  1303          1424  1 !
;  1304          1425  1 !--
;  1305          1426  1 !
;  1306          1427  1 !
;  1307          1428  1 !
;  1308          1429  1 !
;  1309          1430  1 !
;  1310          1431  1 !
;  1311          1432  1 !
;  1312          1433  1 !
;  1313          1434  1 !
;  1314          1435  1 !
;  1315          1436  1 !
;  1316          1437  1 !
;  1317          1438  1 !
;  1318          1439  1 !
;  1319          1440  1 !
;  1320          1441  1 !
;  1321          1442  1 !
;  1322          1443  1 !
;  1323          1444  1 !
;  1324          1445  1 !
;  1325          1446  1 !
;  1326          1447  1 !
;  1327          1448  1 !
;  1328          1449  1 !
;  1329          1450  1 !
;  1330          1451  1 !
;  1331          1452  1 !
;  1332          1453  1 !
;  1333          1454  1 !
;  1334          1455  1 !
;  1335          1456  1 !
;  1336          1457  1 !
;  1337          1458  1 !
;  1338          1459  1 !
;  1339          1460  1 !
;  1340          1461  1 !
;  1341          1462  1 !
;  1342          1463  1 !
;  1343          1464  1 !
;  1344          1465  1 !
;  1345          1466  1 !
;  1346          1467  1 !
;  1347          1468  2     BEGIN
;  1348          1469  2     MAP
;  1349          1470  2         DESC: REF DBG$VALDESC;
;  1350          1471  2     LOCAL
```

FUNCTIONAL DESCRIPTION:

Returns a list of symids contained within a language specific primary or value descriptor.

This routine calls a language-specific routine based on the language code in the descriptor header.

FORMAL PARAMETERS:

desc - A longword containing the address of a language specific primary or value descriptor.

param2 - The address of a longword to contain the address of the first node in the symid list. Each node in the consists of a two longword block. The first longword is the link field and contains the address of the next node in the list. This field is 0 for the last node in the list. The second longword contains the value of a symid. Each symid that appears in a descriptor should appear once and only once in the symid list.

param3 - The address of a longword to contain the address of a message argument vector as described on page 4-119 of the VAX/VMS system reference, volume 1A

IMPLICIT INPUTS:

NONE

IMPLICIT OUTPUTS:

In case of a severe error return, a message argument vector is constructed from dynamic storage and returned.

ROUTINE VALUE:

An unsigned integer longword completion code

COMPLETION CODES:

STS$K_SUCCESS (1) - Success. Symid list constructed and returned.

STS$K_SEVERE (4) - Failure. No symid list returned. Message argument vector constructed and returned.

SIDE EFFECTS:

NONE

```
;  1351          1472  2          DIMCNT,
;  1352          1473  2          NUMBLK$,
;  1353          1474  2          SUBCNT,
;  1354          1475  2          SYMID_LIST;                         ! Pointer to the head of
;  1355          1476  2                                              !    the symid list.
;  1356          1477  2
;  1357          1478  2      ! Implementation level 3 - all languages at this level have common
;  1358          1479  2      ! descriptors so we construct the symid list here.
;  1359          1480  2      !
;  1360          1481  2
;  1361          1482  2          ROUTINE APPEND_TO_LIST (SYMID, SYMID_LIST) : NOVALUE =
;  1362          1483  2          !
;  1363          1484  2          ! FUNCTION
;  1364          1485  2          !       This subroutine is used below to append a new symid
;  1365          1486  2          !       to the symid list under construction.
;  1366          1487  2          !
;  1367          1488  2          ! INPUTS
;  1368          1489  2          !       SYMID -         The symid to be added to the list.
;  1369          1490  2          !       SYMID_LIST -    Points to a longword containing a pointer
;  1370          1491  2          !                       to the head of the symid list.
;  1371          1492  2          !
;  1372          1493  2          ! OUTPUTS
;  1373          1494  2          !       If the symid list was empty, a one-node list will be
;  1374          1495  2          !       created and the SYMID_LIST parameter will contain
;  1375          1496  2          !       a pointer to this one-node list.
;  1376          1497  2          !       Otherwise, the SYMID_LIST parameter is left unchanged
;  1377          1498  2          !       but a node may be added to the list it points to.
;  1378          1499  2          !
;  1379          1500  3              BEGIN
;  1380          1501  3              LOCAL
;  1381          1502  3                  LINK_NODE: REF DBG$LINK_NODE,   ! Pointer to a node in the
;  1382          1503  3                                                  !    symid list.
;  1383          1504  3                  PREV_NODE: REF DBG$LINK_NODE;   ! Pointer to a node in the
;  1384          1505  3                                                  !    symid list.
;  1385          1506  3
;  1386          1507  3              ! If the symid is zero, do not add it to the list.
;  1387          1508  3              !
;  1388          1509  3              IF .SYMID EQL 0 THEN RETURN;
;  1389          1510  3
;  1390          1511  3              ! First check whether the given symid is on the list already.
;  1391          1512  3              !
;  1392          1513  3              LINK_NODE = ..SYMID_LIST;
;  1393          1514  3              PREV_NODE = .SYMID_LIST;
;  1394          1515  3              WHILE .LINK_NODE NEQ 0 DO
;  1395          1516  4                  BEGIN
;  1396          1517  4                  IF .LINK_NODE [DBG$L_LINK_NODE_VALUE] EQL .SYMID
;  1397          1518  4                  THEN
;  1398          1519  4                      RETURN;
;  1399          1520  4                  PREV_NODE = .LINK_NODE;
;  1400          1521  4                  LINK_NODE = .LINK_NODE [DBG$L_LINK_NODE_LINK];
;  1401          1522  3                  END;
;  1402          1523  3
;  1403          1524  3
;  1404          1525  3              ! Allocate space for a new node and put it on the list.
;  1405          1526  3              !
;  1406          1527  3              LINK_NODE = DBG$GET_TEMPMEM (DBG$K_LINK_NODE_SIZE);
;  1407          1528  3              PREV_NODE [DBG$L_LINK_NODE_LINK] = .LINK_NODE;
```

```
; 1408          1529  3               LINK_NODE [DBG$L_LINK_NODE_VALUE] = .SYMID;
; 1409          1530  2               END;


                                                000C 00000 APPEND_TO_LIST:
                                                           .WORD   Save R2,R3                                  ; 1482
                                    52      04  AC  D0 00002        MOVL    SYMID, R2                          ; 1509
                                                2A  13 00006        BEQL    3$                                 ;
                                    50      08  BC  D0 00008        MOVL    @SYMID_LIST, LINK_NODE             ; 1513
                                    53      08  AC  D0 0000C        MOVL    SYMID_LIST, PREV_NODE              ; 1514
                                                50  D5 00010 1$:     TSTL    LINK_NODE                          ; 1515
                                                0E  13 00012        BEQL    2$                                 ;
                                    52      04  A0  D1 00014        CMPL    4(LINK_NODE), R2                   ; 1517
                                                18  13 00018        BEQL    3$                                 ;
                                    53          50  D0 0001A        MOVL    LINK_NODE, PREV_NODE               ; 1520
                                    50          60  D0 0001D        MOVL    (LINK_NODE), LINK_NODE             ; 1521
                                                EE  11 00020        BRB     1$                                 ; 1515
                                                02  DD 00022 2$:     PUSHL   #2                                 ; 1527
                          00000000G 00          01  FB 00024        CALLS   #1, DBG$GET_TEMPMEM                ;
                                    63          50  D0 0002B        MOVL    LINK_NODE, @PREV_NODE)             ; 1528
                              04    A0          52  D0 0002E        MOVL    R2, 4(LINK_NODE)                   ; 1529
                                                    04 00032 3$:     RET                                        ; 1530

; Routine Size:  51 bytes,    Routine Base:  DBG$CODE + 04C3

; 1410          1531  2
; 1411          1532  2       ! Initialize the pointer to the symid list.
; 1412          1533  2       !
; 1413          1534  2       SYMID_LIST = 0;
; 1414          1535  2
; 1415          1536  2       ! Handle value descriptors separately from primary descriptors.
; 1416          1537  2       !
; 1417          1538  2       SELECTONE .DESC [DBG$B_DHDR_TYPE] OF
; 1418          1539  2           SET
; 1419          1540  2
; 1420          1541  2           ! Ordinary value descriptors.
; 1421          1542  2           !
; 1422          1543  2           [DBG$K_VALUE_DESC, DBG$K_V_VALUE_DESC]:
; 1423          1544  3               BEGIN
; 1424          1545  3               MAP
; 1425          1546  3                   DESC: REF DBG$VALDESC;  ! Pointer to a new style value
; 1426          1547  3                                           !     descriptor
; 1427          1548  3               APPEND_TO_LIST (.DESC [DBG$L_DHDR_TYPEID], SYMID_LIST);
; 1428          1549  3               APPEND_TO_LIST (.DESC [DBG$L_DHDR_SYMIDO], SYMID_LIST);
; 1429          1550  2               END;
; 1430          1551  2
; 1431          1552  2           ! New style Primary Descriptors. Here we have to get symids from
; 1432          1553  2           ! the root node and all sub-nodes.
; 1433          1554  2           !
; 1434          1555  2           [DBG$K_PRIMARY_DESC]:
; 1435          1556  3               BEGIN
; 1436          1557  3
; 1437          1558  3               MAP
```

```
;  1438          1559  3          DESC: REF DBG$PRIMARY;                  ! Pointer to the Primary
;  1439          1560  3                                                  !     Descriptor for which
;  1440          1561  3                                                  !     a symid list is to
;  1441          1562  3                                                  !     be constructed.
;  1442          1563  3
;  1443          1564  3      LOCAL
;  1444          1565  3          SUBNODE: REF DBG$PRIM_NODE;             ! Pointer to a subnode.
;  1445          1566  3
;  1446          1567  3      ! Append the typeid and the symid from the root node.
;  1447          1568  3      !
;  1448          1569  3      APPEND_TO_LIST (.DESC [DBG$L_DHDR_TYPEID], SYMID_LIST);
;  1449          1570  3      APPEND_TO_LIST (.DESC [DBG$L_DHDR_SYMID0], SYMID_LIST);
;  1450          1571  3
;  1451          1572  3      ! Loop through each of the subnodes.
;  1452          1573  3      !
;  1453          1574  3      SUBNODE = .DESC [DBG$L_PRIM_FLINK];
;  1454          1575  3      WHILE .SUBNODE NEQ DESC[DBG$L_PRIM_FLINK] DO
;  1455          1576  4          BEGIN
;  1456          1577  4
;  1457          1578  4          ! All kinds of subnodes have typeids and symids
;  1458          1579  4          ! so we append these.
;  1459          1580  4          !
;  1460          1581  4          APPEND_TO_LIST (.SUBNODE [DBG$L_PNODE_TYPEID], SYMID_LIST);
;  1461          1582  4          APPEND_TO_LIST (.SUBNODE [DBG$L_PNODE_SYMID], SYMID_LIST);
;  1462          1583  4
;  1463          1584  4          ! If the subnode is an array node then it also
;  1464          1585  4          ! has typeids in the subscript vector.
;  1465          1586  4          !
;  1466          1587  4          IF .SUBNODE [DBG$B_PNODE_FCODE] EQL RST$K_TYPE_ARRAY
;  1467          1588  4          THEN
;  1468          1589  5              BEGIN
;  1469          1590  5              LOCAL
;  1470          1591  5                  SUBVECTOR: REF DBG$PRIM_NODE_SUBS;
;  1471          1592  5              APPEND_TO_LIST (.SUBNODE [DBG$L_PNARR_CELLTYPE],SYMID_LIST);
;  1472          1593  5              SUBVECTOR = SUBNODE [DBG$A_PNARR_SVECTOR];
;  1473          1594  5              ! Use whichever is larger, subcnt or dimcnt.
;  1474          1595  5              SUBCNT = .SUBNODE[DBG$B_PNARR_SUBCNT];
;  1475          1596  5              DIMCNT = .SUBNODE[DBG$B_PNARR_DIMCNT];
;  1476          1597  5              IF .SUBCNT GTR .DIMCNT
;  1477          1598  5              THEN
;  1478          1599  5                  NUMBLKS = .SUBCNT
;  1479          1600  5              ELSE
;  1480          1601  5                  NUMBLKS = .DIMCNT;
;  1481          1602  5              INCR I FROM 0 TO .NUMBLKS-1 DO
;  1482          1603  5                  APPEND_TO_LIST (.SUBVECTOR[.I, DBG$L_PNSUB_TYPEID],
;  1483          1604  5                                  SYMID_LIST);
;  1484          1605  5              END
;  1485          1606  4          ELSE IF .SUBNODE [DBG$B_PNODE_FCODE] EQL RST$K_TYPE_VARIANT
;  1486          1607  4          THEN
;  1487          1608  4              APPEND_TO_LIST(.SUBNODE[DBG$L_PNVAR_TAGID],SYMID_LIST);
;  1488          1609  4
;  1489          1610  4          SUBNODE = .SUBNODE [DBG$L_PNODE_FLINK];
;  1490          1611  3          END;
;  1491          1612  2      END;
;  1492          1613  2
;  1493          1614  2  ! At implementation level 3, we do not expect any other kind
;  1494          1615  2  ! of descriptor.
```

```
;  1495          1616  2          !
;  1496          1617  2          [OTHERWISE]:
;  1497          1618  2              $DBG_ERROR ('DBGLANVEC\DBG$NGET_SYMID',;
;  1498          1619  2
;  1499          1620  2          TES;
;  1500          1621  2
;  1501          1622  2      ! The symid list has been constructed. Return success.
;  1502          1623  2      !
;  1503          1624  2      .PARAM2 = .SYMID_LIST;
;  1504          1625  2      RETURN STS$K_SUCCESS;
;  1505          1626  1      END;


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

24 47 42 44 5C 43 45 56 4E 41 4C 47 42 44 18  00137 P.AAJ:  .ASCII  <24>\DBGLANVEC\<92>\DBG$NGET_SYMID\          ;
                  44 49 4D 59 53 5F 54 45 47 4E  00146                                                          ;


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

                                          03FC 00000            .ENTRY  DBG$NGET_SYMID, Save R2,R3,R4,R5,R6,R7,R8,-  ; 1415
                                                                R9                                                  ;
                              59      C8  AF  9E 00002          MOVAB   APPEND_TO_LIST, R9                          ;
                                      7E  D4 00006              CLRL    SYMID_LIST                                  ; 1534
                              55      04  AC  D0 00008          MOVL    DESC, R5                                    ; 1538
                        7A    8F      02  A5  91 0000C          CMPB    2(R5), #122                                 ; 1543
                                          07  13 00011          BEQL    1$                                          ;
                        83    8F      02  A5  91 00013          CMPB    2(R5), #131                                 ;
                                          13  12 00018          BNEQ    2$                                          ;
                                      5E  DD 0001A 1$:          PUSHL   SP                                          ; 1548
                                  08  A5  DD 0001C              PUSHL   8(R5)                                       ;
                              69          02  FB 0001F          CALLS   #2, APPEND_TO_LIST                          ;
                                      5E  DD 00022              PUSHL   SP                                          ; 1549
                                  0C  A5  DD 00024              PUSHL   12(R5)                                      ;
                              69          02  FB 00027          CALLS   #2, APPEND_TO_LIST                          ;
                                    00A0  31 0002A              BRW     12$                                         ; 1538
                        79    8F      02  A5  91 0002D 2$:      CMPB    2(R5), #121                                 ; 1555
                                          03  13 00032          BEQL    3$                                          ;
                                    0081  31 00034              BRW     11$                                         ;
                                      5E  DD 00037 3$:          PUSHL   SP                                          ; 1569
                                  08  A5  DD 00039              PUSHL   8(R5)                                       ;
                              69          02  FB 0003C          CALLS   #2, APPEND_TO_LIST                          ;
                                      5E  DD 0003F              PUSHL   SP                                          ; 1570
                                  0C  A5  DD 00041              PUSHL   12(R5)                                      ;
                              69          02  FB 00044          CALLS   #2, APPEND_TO_LIST                          ;
                              52      14  A5  D0 00047          MOVL    20(R5), SUBNODE                             ; 1574
                              50      14  A5  9E 0004B 4$:      MOVAB   20(R5), R0                                  ; 1575
                              50          52  D1 0004F          CMPL    SUBNODE, R0                                 ;
                                          79  13 00052          BEQL    12$                                         ;
                                      5E  DD 00054              PUSHL   SP                                          ; 1581
                                  0C  A2  DD 00056              PUSHL   12(SUBNODE)                                 ;
                              69          02  FB 00059          CALLS   #2, APPEND_TO_LIST                          ;
                                      5E  DD 0005C              PUSHL   SP                                          ; 1582
                                  10  A2  DD 0005E              PUSHL   16(SUBNODE)                                 ;
```

```
                69          02  FB 00061          CALLS   #2, APPEND_TO_LIST
                01    09    A2  91 00064          CMPB    9(SUBNODE), #1                          ; 1587
                            3B  12 00068          BNEQ    9$
                            5E  DD 0006A          PUSHL   SP                                      ; 1592
                      24    A2  DD 0006C          PUSHL   36(SUBNODE)
                19          02  FB 0006F          CALLS   #2, APPEND_TO_LIST
                53    28    A2  9E 00072          MOVAB   40(R2), SUBVECTOR                       ; 1593
                56    1F    A2  9A 00076          MOVZBL  31(SUBNODE), SUBCNT                     ; 1595
                58    1B    A2  9A 0007A          MOVZBL  27(SUBNODE), DIMCNT                     ; 1596
                58          56  D1 0007E          CMPL    SUBCNT, DIMCNT                          ; 1597
                            05  15 00081          BLEQ    5$
                57          56  D0 00083          MOVL    SUBCNT, NUMBLKS                         ; 1599
                            03  11 00086          BRB     6$
                57          58  D0 00088 5$:      MOVL    DIMCNT, NUMBLKS                         ; 1601
                54          01  CE 0008B 6$:      MNEGL   #1, I                                   ; 1603
                            0F  11 0008E          BRB     8$
                            5E  DD 00090 7$:      PUSHL   SP
          50    54          14  C5 00092          MULL3   #20, I, R0
                      10 A043   9F 00096          PUSHAB  16(R0)[SUBVECTOR]
                            9E  DD 0009A          PUSHL   @(SP)+
                69          02  FB 0009C          CALLS   #2, APPEND_TO_LIST
          ED    54          57  F2 0009F 8$:      AOBLSS  NUMBLKS, I, 7$
                            0E  11 000A3          BRB     10$                                     ; 1587
                13    09    A2  91 000A5 9$:      CMPB    9(SUBNODE), #19                         ; 1606
                            08  12 000A9          BNEQ    10$
                            5E  DD 000AB          PUSHL   SP                                      ; 1608
                      1C    A2  DD 000AD          PUSHL   28(SUBNODE)
                69          02  FB 000B0          CALLS   #2, APPEND_TO_LIST
                52          62  D0 000B3 10$:     MOVL    (SUBNODE), SUBNODE                      ; 1610
                            93  11 000B6          BRB     4$                                      ; 1575
                   00000000'    EF  9F 000B8 11$:     PUSHAB  P.AAJ                             ; 1618
                            01  DD 000BE          PUSHL   #1
                   00028362     8F  DD 000C0          PUSHL   #164706
          00000000G 00          03  FB 000C6          CALLS   #3, LIB$SIGNAL
                08    BC        6E  D0 000C9 12$:     MOVL    SYMID_LIST, @PARAM2               ; 1624
                50              01  D0 000D1          MOVL    #1, R0                            ; 1625
                                04  000D4          RET                                       ; 1626

; Routine Size:  213 bytes,    Routine Base:  DBG$CODE + 04F6
```

```
;1507          1627  1 GLOBAL ROUTINE DBG$NINITIALIZE : NOVALUE =
;1508          1628  1 !
;1509          1629  1 !  FUNCTION
;1510          1630  1 !        This routine calls language specific initialization routines.  This is
;1511          1631  1 !        done before each command is processed to garuantee the integrity of the
;1512          1632  1 !        language specific machinery.
;1513          1633  1 !
;1514          1634  1 !  FORMAL PARAMETERS:
;1515          1635  1 !
;1516          1636  1 !        NONE
;1517          1637  1 !
;1518          1638  1 !  IMPLICIT INPUTS:
;1519          1639  1 !
;1520          1640  1 !        NONE
;1521          1641  1 !
;1522          1642  1 !  IMPLICIT OUTPUTS:
;1523          1643  1 !
;1524          1644  1 !        NONE
;1525          1645  1 !
;1526          1646  1 !  ROUTINE VALUE:
;1527          1647  1 !
;1528          1648  1 !        NOVALUE
;1529          1649  1 !
;1530          1650  1 !  COMPLETION CODES:
;1531          1651  1 !
;1532          1652  1 !        NONE
;1533          1653  1 !
;1534          1654  1 !  SIDE EFFECTS:
;1535          1655  1 !
;1536          1656  1 !        NONE
;1537          1657  1 !
;1538          1658  2     BEGIN
;1539          1659  2     RETURN;
;1540          1660  1     END;


                                                0000 00000         .ENTRY  DBG$NINITIALIZE, Save nothing                ;1627
                                                  04 00002         RET                                                  ;1660

; Routine Size:  3 bytes,    Routine Base:  DBG$CODE + 05CB

;1541          1661  0 END ELUDOM


                                                                   .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY
;
;      Name                      Bytes                           Attributes
;
;  DBG$OWN                           4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
```

```
;  DBG$PLIT                    336  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                   1486  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          6          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        110          7         97        00:02.0
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                       418          3          0         31        00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1
;                                       386          2          0         22        00:00.3
```

```
;                                COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:DBGLANVEC/OBJ=OBJ$:DBGLANVEC MSRC$:DBGLANVEC/UPDATE=(ENH$:DBGLANVEC)

; Size:          1486 code + 340 data bytes
; Run Time:         00:32.0
; Elapsed Time:     01:49.9
; Lines/CPU Min:     3116
; Lexemes/CPU-Min: 11001
; Memory Used:  181 pages
; Compilation Complete
```

DBGIFTHEN
LIS

DBGLANVEC
LIS

DBGLANGOP
LIS

DBGGEN
LIS

DBGLEVEL1
LIS